Proyecto Educativo
de Maestría en Docencia Universitaria

Lenguaje de programación
Python para las Ciencias Económicas

(Una propuesta curricular innovadora)

Directora: Dra. Susana Marcipar Katz

Alumno: Sebastián Fumis

sfumis@gmail.com

Facultad de Humanidades y Ciencias, Universidad Nacional del Litoral

Índice

Agradecimientos La presentación de este Proyecto Educativo como corolario de una carrera de posgrado es un motivo de orgullo y satisfacción.

Sensaciones que son mas colectivas que propias, que se sustentan en el marco de oportunidades que nos brinda la Universidad Pública, en particular la prestigiosa Universidad Nacional del Litoral, principal impulsora de la formación continua de sus cuerpos docentes y también destinataria de los aportes que de este proyecto puedan surgir.

También extiendo los agradecimientos a mi Directora del proyecto, Dra. Susana Marcipar Katz, por ser soporte emocional en gran parte de este trayecto; su experiencia y motivación fueron fundamentales para el desarrollo de este trabajo de investigación.

A las autoridades de la Facultad de Ciencias Económicas, pasadas y presentes, por fomentar en su cuerpo docente un espíritu de mejora continua y apoyar las iniciativas que se les presentan.

En particular, dentro de la comunidad interna de la facultad, a los integrantes del equipo de investigación que se formó y que dio sustento a gran parte de este trabajo.

A los profesionales que encontré en este camino de investigación que hicieron crecer mi la curiosidad en un tema ajeno a mi formación profesional: a Federico Brun por encender la chispa de Python en mí.

A autores como Bota y Gosa por realizar un notable aporte bibliográfico del uso de Python orientado a la labor del profesional en Ciencias Económicas. Al anónimo divulgador que sostiene el canal de YouTube Pythonic Accountant por sus desinteresados y valiosísimos aportes.

A Al Sweigart por entusiasmar a Automatizar lo aburrido, que nuestra profesión lo tiene y mucho.

A los colegas docentes que conformamos la cohorte 2019 de esta Maestría en Docencia Universitaria con quienes compartimos un camino de intercambios entre profesores y profesoras provenientes de diferentes unidades académicas de la UNL, con quienes nos queda una relación de mutuo reconocimiento a la labor que desempeñamos. El intercambio de otras experiencias y el apoyo incondicional han sido motivadores durante este camino. Una permanente actitud proactiva y su compañía hicieron más llevadera esta etapa de mi formación académica.

A todos y cada uno de los maestros que dejaron marcada su impronta en mi formación les dedico las mismas palabras que tuviera A. Camus para con el querido Señor Germain:

Sin usted, la mano afectuosa que tendió al pobre niñito que era yo, sin su enseñanza y ejemplo, no hubiese sucedido nada de esto ... y le puedo asegurar que sus esfuerzos, su trabajo y el corazón generoso que usted puso continúan siempre vivos en uno de sus pequeños discípulos, que, a pesar de los años, no ha dejado de ser su alumno agradecido.

Finalmente, y por su mayor importancia hago extensivo mi agredecimiento a mi familia por su amor, comprensión y sacrificio. Su apoyo constante fue mi mayor motivación para perseguir mis metas y superar los desafíos que encontré en el camino.

Parte I

Conceptos, definiciones y alcances

1. Síntesis del proyecto educativo

.

En el acto inaugural de la Maestría en Docencia Universitaria en el año 2019 se presentó como novedad la posibilidad de presentar un proyecto educativo como documento final integrador de la misma, y en palabras de las autoridades de la Facultad de Humanidades y Ciencias (FHUC) esta modalidad debía presentarse en formato de tesis o Proyecto Educativo (PE).

El proyecto educativo «Lenguaje de programación Python para las Ciencias Económicas» como trabajo final de la Maestría en Docencia Universitaria de la Facultad de Humanidades y Ciencias de la Universidad Nacional del Litoral (UNL) pretende enriquecer el contenido curricular de las carreras de Contador Público (CP) y Licenciatura en Administración (LA), a través de la formalización de una propuesta ante el Consejo Directivo de la Facultad de Ciencias Económicas (FCE) de la UNL.

El proyecto ha sido pensado como una instancia propicia para la adopción de una metodología pedagógica basada en casos reales donde se logre la integración de experiencias laborales concretas dentro del aula mediante presentaciones narradas y contextualizadas.

A su vez, aprovechando la flexibilidad que dan las materias optativas al plan de estudios, se abre la puerta para el desafío de implementar un enfoque didáctico centrado en los estudiantes y sus intereses particulares. Debido a las múltiples soluciones posibles que los mismos presentan, se añade un desafío en el campo evaluativo para los docentes.

El abordaje de los casos en forma grupal, apunta a la generación de ámbitos colaborativos con el objetivo de ir generando una masa crítica que no sólo permita a la FCE-UNL ser pionera en esta temática a nivel país, sino también que sea semillero de una generación de contadores públicos con un mayor grado de alfabetización en lenguajes de programación que los estimule a incorporarse de una u otra forma dentro de las funciones sustantivas de la universidad pública: el «trípode docencia, extensión e investigación». (Baraldi, 2016:306-313)

En otro orden de importancia, es que la cuantiosa inversión en infraestructura informática que ha realizado la FCE-UNL en los últimos años, exige y justifica todas aquellas propuestas de aprovechamiento de las mismas, permitiendo sus aulas totalmente equipadas el uso de ordenadores portátiles individuales para cada alumno.

Específicamente, el presente proyecto tiene como objetivo diseñar instancias educativas que generen una masa crítica especializada en el uso de un lenguaje de programación para optimizar las tareas del profesional en ciencias económicas.

El trabajo de campo y sus hallazgos permiten encarar con optimismo el proceso de implementación de la idea fuerza que motivó esta investigación en el corto plazo.

Se ha detectado un favorable interés receptivo por parte de la comunidad educativa, y ya se ha logrado que el tema en estudio se materialice en otros proyectos y grupos de investigación dentro de la FCE.

Por otro lado, si bien los casos de estudio aquí presentados se basan en casos de aplicación práctica real, sólo dan una muestra de todo el potencial que encierra el desafío de la formación en lenguajes de programación a los profesionales en Ciencias Económicas.

La idea fuerza de la iniciativa se centra en la formalización de una asignatura optativa a partir del año 2024, para la carrera de CP y LA de la FCE-UNL.

2. La importancia del currículo abierto

La posibilidad de realizar un PE se incorporó por primera vez para la cohorte 2019, como una alternativa a las tradicionales tesis de maestrías, teniendo como objetivo generar propuestas que puedan lograr a futuro un impacto directo en las prácticas docentes de los maestrandos.

Los distintos seminarios dictados pusieron de relieve la importancia de brindarle a los maestrandos herramientas que permitan comprender su rol docente en la universidad argentina, incorporando aspectos pedagógicos, sociológicos, psicológicos, y de la relación docente-alumno.

El desafío por interrelacionar los contenidos brindados durante el cursado de los distintos seminarios con una propuesta educativa llama a reflexionar sobre las propias prácticas docentes, con una mirada introspectiva hacia dentro de la Unidad Académica donde se desarrollan y contextualizada en las necesidades que exige y exigirá el desarrollo profesional de los futuros graduados y graduadas.

El plan de estudios vigente para la carrera de Contador Público (CP) de la Facultad de Ciencias Económicas de la Universidad Nacional del Litoral (FCE-UNL) fue aprobado por el Consejo Superior en octubre de 2018 (Res. CS Nº 502/18).

El nuevo plan tuvo un proceso de análisis iniciado en el año 2016 cuando la conducción de la FCE-UNL elevó al Consejo Directivo un proyecto para la creación de la Comisión de Revisión Curricular.

Ésta se conformó con los Consejeros Directivos de la Comisión de Enseñanza, el Secretario Académico y los directores de los departamentos de la Facultad, iniciando sus actividades el 31/5/2016 y estableció para el año siguiente el desarrollo de diferentes instancias de consulta a toda la comunidad educativa y al medio, tratando de lograr la mayor participación posible de todos los actores involucrados, para poder establecer las líneas básicas y una propuesta de perfil del egresado. Los resultados de esta consulta se constituyen en los insumos básicos para diseñar un nuevo plan de estudios durante el año 2018 de la carrera de CP.

El cambio curricular hacia un enfoque más abierto y flexible en la educación responde directamente a las demandas y dinámicas del mercado laboral que se proyecta desafiante. En un entorno donde la tecnología y la globalización redefinen constantemente las habilidades y competencias requeridas, los futuros profesionales deben estar preparados para adaptarse a un panorama en constante evolución.

El enfoque basado en un currículo abierto en la formación en Ciencias Económicas no solo se alinea con las necesidades cambiantes del mercado laboral, sino que también pretende dotar a los profesionales con las competencias necesarias para sobresalir en un entorno dinámico,

exigente y competitivo.

El presente proyecto educativo se justifica por su pertinencia y oportunidad, en términos de Marcipar Katz y Zanabria:

> Cuando hablamos de 'cambio curricular' al interior de una institución educativa estamos hablando de cambios que deben afectar a: las prácticas docentes, la relación entre los integrantes de esa comunidad, la relación con la sociedad, la articulación entre ciencia y desarrollo, las concepciones sobre el rol educativo, las posturas epistemológicas, entre otras. En definitiva, cuando hoy hablamos de cambio curricular estamos hablando de cambios en la cultura del conocimiento. De no ser así, entendemos que los cambios sólo afectan al currículum explícito, pero éste no alcanza para dar cuentas de transformaciones al interior de una institución educativa. (2007:79-88.)

Es oportuno rescatar en este punto, los aportes logrados a partir de la instancia de «consulta externa» que fue diseñada e implementada por el Observatorio Social de la UNL y buscó conocer la opinión prospectiva que los contadores públicos tienen sobre las características y los requerimientos de su profesión para el año 2030.

De allí se desprende como relevante la percepción a futuro de un ámbito de trabajo profesional con mayor tecnificación de las tareas ligadas a la proliferación de las nuevas tecnologías de la información y la comunicación, situación que sobrelleva a que su labor se concentre en la compilación y recepción on-line de la información de los clientes o actores de las organizaciones. En ese contexto advierten la necesidad de actualización constante de su formación teórica y práctica. Su labor profesional, opinan, requerirá de una actitud proactiva hacia la actualización y aprendizaje en el puesto de trabajo y la ejecución de labores en equipos interdisciplinarios.

En una entrevista realizada en el diario El Litoral de Santa Fe el tecnólogo y emprendedor, Santiago Bilinkis nos deja una reflexión interpelativa:

> Si vos vas a una escuela de hace 100 años y vas a una escuela hoy, vas a ver muy poca diferencia. Y eso es muy preocupante porque la educación tiene que preparar a los chicos para lo que viene. Si no tenemos la pregunta respecto de cómo va a ser el mundo dentro de diez años y qué habilidades van a hacer falta en ese escenario, no estamos educando correctamente. (Muñoz (2022, 24 de noviembre), «Chat GPT, un "monstruo""" tecnológico que "se metió por la ventana" y revolucionó el mundo»)

Mediante la Resolución Nº 3400/2017 el Ministerio de Educación de la Nación quedaron establecidos los contenidos curriculares básicos para la carrera de CP. Los contenidos, agrupados por áreas temáticas son considerados obligatorios no así la estructura que se presenta, que no es

prescriptiva.

Dentro del área temática de Administración y Tecnologías de la Información, se ubican como contenidos los siguientes:

- Teoría general de los sistemas.
- Metodología de análisis, diseño e implementación de los sistemas de información.
- Aspectos tecnológicos de los medios de procesamiento y comunicaciones.
- Utilización de software de base, utilitarios y redes.
- Evaluación de sistemas aplicativos.
- Seguridad en los sistemas de información.

El plan de estudio actual exige un total de 270 horas en materias optativas que el alumno va eligiendo en su trayecto curricular.

El hecho de proponerse como una materia optativa, también abre la posibilidad de inducir hacia la construcción de conocimientos que sean del propio interés del educando por sobre la mera adquisición de los mismos y a la vez una oportunidad para introducir prácticas pedagógicas centradas en el alumno eficaces para lograr un proceso de aprendizaje transferible a la práctica profesional.

Se considera que un estudiante en una etapa avanzada de la carrera ya en el ciclo especializado, y con alguna práctica laboral en ciernes reunirá los mejores requisitos para nutrirse de los contenidos de la propuesta pero a la vez ser capaz de enriquecer la clase volcando y generando sus propias inquietudes laborales a un entorno donde la participación del estudiante es considerada central.

Entonces, resultó oportuno detectar un interés tanto de la comunidad vinculada al quehacer profesional como en los contenidos mínimos exigidos por la autoridad educativa por avanzar y profundizar en los conocimientos tecnológicos útiles para la profesión.

La enseñanza de algún lenguaje informático ampliará las capacidades de procesar la información que disponen los profesionales en ciencias económicas, en los sistemas informáticos con los que interactúa. Esto redundará en un mejor y mas eficiente desempeño de las tareas que realiza y una optimización del tiempo que requieren.

3. Habilidades y competencias del profesional en Ciencias Económicas.

Las primeras registraciones sobre los hechos económicos o patrimoniales datan del S. XV y desde entonces, tanto la ciencia que se dedica a su estudio como la denominación del profesional quedaron ligados al concepto medular sobre el cual se desarrollan: las cuentas. Para Pahlen Acuña et. al.: «las cuentas constituyen el eslabón entre las operaciones que realiza el ente como así también los hechos económicos que afectan su patrimonio ... y el inicio del proceso de registración de los datos en el Sistema de Información Contable». (2011:7)

Una visión moderna del rol del Contador Público lo ubica como un actor comprometido con el proceso decisorio y de control dentro del ente en el que desarrolla sus actividades, en este sentido adquiere un rol relevante en la administración del Sistema Integrado de Información.

La actividad profesional del contador/a permite interpretar el significado y poder comunicar a terceros de manera fehaciente, el conjunto de información procesada que es de utilidad para tomar decisiones -exante- u observar por el control de dichas decisiones y/o resultados -expost. (Ostengo, 2015:48)

La información contable es un insumo importante para la toma de decisiones dentro de las organizaciones. Esto se debe a las siguientes razones:

- Proporciona una visión general del rendimiento económico y financiero tanto para uso interno del ente, como para terceros interesados. Esto incluye datos sobre ingresos, gastos, activos, pasivos y patrimonio neto.

- Los Estados Contables ayudan a identificar las fortalezas y debilidades de una empresa o organización.

- La información contable orienta la toma de decisiones a largo plazo.

- La información contable es importante para cumplir con las obligaciones fiscales y regulatorias.

La multiplicidad de usuarios de la información contable y administrativa y sus propios diseños de los programas informáticos exigen habilidades para compatibilizar estas situaciones mediante la sistematización de la información que es de interés propio de cada organización.

A modo de ejemplo, la información que requiere la AFIP tiene un formato diferente a la información interna que procesa cualquier ente o unidad económica. Inclusive, los diferentes tipos de información contable interna tienen formatos distintos que se compatibilizan manualmente.

Las organizaciones ya han comenzado a demandar capacidades no incorporadas en los planes de estudio, basados en la tradición informática actual que centra sus competencias en el uso de software específico -profesional como usuario.

En términos generales se ha observado que, con archivos digitales de diferentes orígenes y sus distintos formatos, muchas veces resulta incompatible utilizar reportes propios del sistema de gestión de la entidad para ser utilizado en otras entidades.

Además, la necesidad de compatibilizar la estructura de los archivos entre distintas fuentes y destinos requiere de muchas horas de trabajo en la práctica profesional de los contadores públicos tanto a nivel de empresa, de organismos estatales como en estudio contable. Estas tareas son rutinarias y se realizan con la periodicidad que la información demandada sea requerida pudiendo ser diaria, semanal, mensual o anual.

La práctica profesional del contador público está íntimamente relacionada con procesos informatizados, y más allá del manejo de una gran cantidad de documentación respaldatoria en formato físico, la base de su trabajo está ligada a información contenida en soporte digital, «por lo que es posible realizar infinita cantidad de combinaciones con los datos que surgen de un movimiento contable»; siendo las planillas de cálculo la más utilizada diariamente. (Pahlen Acuña et al., 2011:22)

Las entidades que nuclean a los profesionales en Ciencias Económicas bregan por la formación permanente y continua de sus asociados, los cambios a los que se enfrenta la práctica profesional no son sólo normativos y en general suelen venir acompañados por una mayor carga de tareas y procedimientos que exigen una respuesta rápida en la actualización.

La formación universitaria, tanto de grado como de posgrado, se enfrentan con los desafíos que derivan de la factibilidad de inserción laboral de sus egresados.

En la actual configuración de los matriculados en las universidades públicas argentinas predominan mayoritariamente las carreras afines a las ciencias sociales.

«En muchos casos, la falta de conexión entre los programas educativos y las necesidades del mercado laboral hace que las grandes universidades estén produciendo legiones de profesionales desempleados» (Oppenheimer, 2013:345.

Los riesgos de una disociación entre contenidos brindados y mercado laboral hacen mas vulnerables a los profesionales frente al desempleo pero también a su reemplazo por la tecnología.

Si bien existe como contrapartida la creencia generalizada en círculos especializados de que la tecnología inexorablemente mejoraría el mundo, ésta exhibe como contrapartida una amenaza para una muy amplia gama de profesiones y oficios que van desde los taxistas que

serán sustituidos por vehículos autónomos como para los asistentes de abogados y contadores según estudios de la universidad de Oxford.

Cierto pesimismo se cierne, desde la perspectiva social, sobre el avance tecnológico por sus impactos en el mercado laboral y en lo que refiere a las Ciencias Económicas en particular queda bien expuesto en el documento "Tendencias contables en el mundo moderno":

> Con base en la revisión de los desarrollos actuales de la inteligencia artificial y las tecnologías de cadena de bloques en la profesión contable, centrándose en las cuatro grandes firmas contables, Zhang et al. (2020) sugirieron que no hay duda de que los robots financieros reemplazarán a los humanos en la realización de tareas contables básicas en el futuro, ya que ya son un componente importante del panorama contable. Como profesional financiero, es importante mejorar continuamente sus conocimientos y habilidades profesionales, incluida la experiencia en informática, para completar tareas más desafiantes (Melnyk et al.,2020)

En la actualidad, la nueva revolución es digital y no industrial. Su núcleo central es la información basada en el exponencial crecimiento y capacidad de almacenamiento de los datos. En la época del **Big Data** la forma de procesarlos y aplicarlos eficientemente dependen de la interacción entre humanos y ordenadores, donde la mayor parte de la tarea la realizarán ellos por nosotros.

Por otra parte además de la mirada propia de los egresados, también resulta pertinente incorporar al análisis las demandas que los propios estudiantes realizan en relación a lograr un mas rápido acercamiento dentro de la carrera universitaria, con los contenidos específicos del perfil profesional y de una mayor aplicabilidad de los contenidos dentro de las asignaturas diseñadas para la práctica profesional.

En síntesis, una asignatura que se presenta como innovadora y que se proyecte como iniciadora en la vinculación entre habilidades en programación aplicadas a las ciencias económicas deberá lograr formar profesionales con habilidades en:

- a. Análisis de datos: Resulta clave tener una buena comprensión de las técnicas y herramientas para el análisis de datos, su correcto procesamiento y el análisis estadístico y la visualización de datos.

- b. Procesamiento de grandes conjuntos de datos: En un mundo cada vez más digital, es importante tener una buena comprensión de cómo procesar y analizar grandes conjuntos de datos. Abarca el aprendizaje de herramientas y técnicas como el procesamiento distribuido, el aprendizaje automático y el procesamiento de datos en paralelo.

- c. Gestión de datos: Es de mucha utilidad tener una buena comprensión de cómo gestionar

> y almacenar datos de manera eficiente, sistémica y segura. Se incluye el aprendizaje de herramientas y técnicas como la gestión de bases de datos, la integración de datos y la gestión de datos en la nube.

Concretamente, para el manejo de datos en las ciencias económicas, es importante adquirir una formación en análisis de datos, modelización económica, procesamiento de grandes conjuntos de datos y gestión de datos. Esto puede incluir el aprendizaje de lenguajes de programación y herramientas de análisis de datos, así como el desarrollo de habilidades en técnicas y procesos especializados.

Nada mas desafiante que acercar al educando a la posibilidad de alcanzar una etapa superior de la formación profesional a la de un usuario experimentado en las planillas de cálculos de Microsoft Excel incorporando conocimientos de un lenguaje de programación como Python queda expuesta en palabras de Félix Zumstein:

> Si Ud. usa Python con Excel, usted estará capacitado para usar un lenguaje de programación que es bueno para todos los aspectos de esta historia, ya sea la automatización de Excel, el acceso y preparación de conjuntos de datos, o el desarrollo de análisis de datos o tareas de visualización, Mas importante, es que Ud. puede reutilizar sus habilidades en Python por fuera de Excel: Si necesita escalar el poder computacional, Ud. podrá fácilmente mover su modelo cuantitativo, de simulación o aplicación de machine learning a la nube, donde lo estarán esperando recursos informáticos prácticamente ilimitados. (2021:8)

4. Propósito central y objetivos

Es necesario entonces, perseguir objetivos de corto y mediano plazo que motiven en los más jóvenes el interés por esta temática, con el convencimiento que la potencialidad que encierra adquirir conocimientos básicos de programación les servirá en toda su carrera profesional con la sola limitante que su propia dedicación les determine. En esta línea se persiguen los objetivos:

- Diseñar instancias educativas que generen en la FCE-UNL masa crítica especializada en el uso de un lenguaje de programación para optimizar eficazmente tareas del profesional en ciencias económicas.

- Desarrollar en lenguaje de programación Python procesos, procedimientos y/o instrucciones de análisis de datos en tareas profesionales de las ciencias económicas de ámbito público o privado de manera tal que sean insumos para el aprendizaje basado en problemas y/o estudio de casos.

- Diseñar un plan de capacitación dirigido a estudiantes avanzados de ciencias económicas para las etapas iniciales en la formación de Python como lenguaje soporte de las tareas de la práctica profesional, que pueda materializarse como una asignatura optativa dentro del plan de estudios de la carrera de CP y LA.

Los resultados esperados se circunscriben a lograr fomentar en los estudiantes el interés temprano en la programación y demostrar la relevancia de la programación en la carrera profesional de ciencias económicas.

Generarles conciencia en la autodeterminación y el compromiso personal para explotar al máximo sus potencialidades en un contexto actual que exige el perfeccionamiento continuo

Orientarlos en los beneficios de integración interdisciplinaria, la complementariedad en las relaciones laborales y los escenarios de trabajos colaborativos y en redes.

5. Etapas y estrategias para su implementación

Resultó muy motivador a mediados de 2022 que la FCE-UNL aprobara por Resolución C.D. Nº 386/2022 una convocatoria dirigida a investigadores cofinanciada por la Secretaría de Políticas Universitaria del Ministerio de Educación de la Nación.

Esta instancia posibilitó articular las tareas previstas en el plan de trabajo de este proyecto educativo con dicha convocatoria denominada «Proyecto Específico de Fortalecimiento para la Investigación en Ciencias Económicas (PROMCE)».

El esquema planteado para los equipos de investigación también permitió darle cabida a la recomendación de "Fomentar la participación de los estudiantes en los proyectos de investigación" que fijara el Comité Evaluador de la CONEAU al momento de acreditar la carrera de CP por la CONEAU en Diciembre de 2020 por el plazo máximo de 6 años.

Participar de esta convocatoria permitió materializar muchas tareas y acciones que forman parte del presente proyecto educativo.

Para el logro de los objetivos propuestos se realizaron actividades secuenciadas y en algunos casos simultáneas. Las mismas pueden agruparse en tres momentos o etapas, a saber:

- Etapa 1: indagatoria/exploratoria

  - Diagnóstico del nivel y alcance de conocimientos sobre lenguajes de programación que circula en el ámbito profesional del Contador Público en ciudad de Santa Fe. Consulta a informantes claves de ámbito público y privado.

  - Diagnóstico sobre grado de interés y motivación de estudiantes avanzados de la carrera CP de la FCE-UNL para aprender lenguajes de programación.

  - Entrevistar a experto en lenguajes de programación acerca de su experiencia en aplicaciones orientada a la actividad profesional en ciencias económicas.

- Etapa 2: Identificación de casos Actividades dentro de la práctica profesional propiamente dicha cuyos resultados puedan ser extrapolados a diferentes situaciones de la actividad del Contador Público.

  - D) Analizar los diferentes reportes de salida que utiliza el sistema de gestión de una organización en cuanto a su estructura de presentación.

  - E) Identificar aquellas prácticas que se realizan mediante planillas de cálculos y que tengan mayores riesgos de errores producto de la manipulación de los datos por acción

humana para programarlas en lenguaje Python.

- F) Identificar aquellos datos de mayor utilidad en la exposición de la información.

- G) Relacionar los reportes del sistema que sirven de base de datos para la confección de otros reportes a utilizar en sistemas externos.

- H) Compatibilizar archivos propios de origen con archivos externos de otras entidades como la Administración Federal de impuestos, bancos, etc. mediante el uso del lenguaje de programación Python.

- I) Redacción de problemas y casos a los fines de presentarlos en las instancias educativas.

- Etapa 3: Prueba Piloto de Instancia educativa en formato de taller.

  - J) Selección de tres estudiantes y tres graduados recientes voluntarios para aprender Python aplicado a situaciones profesionales en ciencias económicas.

  - K) Definición y descripción de: contenidos disciplinares, objetivos; cronograma; método de trabajo y evaluación de los aprendizajes de la instancia educativa.

  - L) Evaluación de la prueba piloto del Taller. Contendrá opiniones de los asistentes respecto a la metodología aplicada y se les solicitará un auto informe de sus aprendizajes. Análisis de logros.

  - M) Informe final del proyecto educativo.

6. Metodología

Se trata de una investigación empírica puesto que entre sus finalidades se encuentra ir conformando una cantidad mínima de personas «masa crítica» capacitadas en lenguaje de programación Python de manera tal que desarrollen procesos, procedimientos y/o análisis de datos del ámbito de las ciencias económicas con eficiencia y confiabilidad.

Por otra parte, en un comienzo el nivel de conocimiento es exploratorio en cuanto se requiere determinar el estado actual de conocimiento sobre lenguaje de programación entre los profesionales de dichas ciencias. Esto implicó indagar entre los profesionales informáticos la experiencia y requerimientos demandados por parte de contadores o unidades de gestión administrativas contables. Además se hizo un trabajo exploratorio entre estudiantes de la carrera de CP interesados en participar voluntariamente en el proyecto de investigación.

En esta primera etapa se aplicaron las siguientes técnicas de observación:

- Revisión de fuentes secundarias (documentales o estadísticas).
- Entrevistas semi-estructuradas dirigidas a Contadores y Profesionales informáticos seleccionados como informantes claves.
- Cuestionarios con preguntas abiertas y cerradas dirigidas a estudiantes (Google Forms).
- Técnicas de toma de decisión.

La segunda etapa de la investigación se desarrolló con técnicas de análisis de información contable.

Dado el alcance temporal del presente proyecto, se consideró pertinente centrar el análisis dentro de un límite espacial de casos o situaciones obtenidas de entidades locales, y en relación con sus propios reportes internos, con las entidades bancarias con las que trabaja, y la Administración Federal de Impuestos.

En este contexto se desarrollaron las actividades E), F), G), H), I) señaladas en la Etapa 2: analítica-descriptiva con las que se pretendió lograr con un selección puntual de casos de procesamiento de datos del ente y/o procesos seleccionados.

Además, de ser un insumo pedagógico para la instancia educativa de capacitación en lenguaje Python, dicha acción reviste potencial para su transferencia al medio.

Teniendo en cuenta los resultados esperados se entiende que este proyecto tiende a promover el mejoramiento de la calidad de la formación en la carrera de Contador Público y

contribuye con los procesos formativos de docentes, estudiantes y graduados a través de las estrategias de enseñanza y aprendizaje aquí descriptas.

La decisión de centrarse en casos específicos dentro de un ámbito local proporciona un enfoque concreto y aplicable, lo que facilita la implementación de las estrategias propuestas y la transferencia inmediata de conocimientos a otros contextos similares, en los que el futuro profesional deba desempeñarse.

La perspectiva integral e integradora de este proyecto apunta a combinar tanto habilidades técnicas derivadas de la programación como conocimientos específicos del campo contable potenciando la capacidad de utilizar herramientas tecnológicas modernas, en contexto de grandes volúmenes de información, cada vez más frecuentes en las ciencias económicas.

7. Dimensiones de análisis y actores involucrados

El proyecto educativo correlaciona las situaciones problemáticas señaladas con sus posibilidades de impacto favorable en las dimensiones tecnológica, económica, educativa y social dentro de la comunidad de profesionales en ciencias económicas.

El vertiginoso mundo en el que nos toca desenvolvernos, tanto como docentes como profesionales independientes tiene un constante y acelerado ritmo evolutivo impactando en múltiples dimensiones que dan forma a esta propuesta educativa: laborales, tecnológicas, económicas y sociales.

Sobrevuela como amenaza para el mercado laboral, el impacto que genera el avance tecnológico. Este acecho es mas o menos inminente de acuerdo al tipo de actividad humana que se realice pero genera incertidumbre en una amplia gama de profesiones y oficios.

> Quizá la revolución tecnológica eche pronto del mercado de trabajo a miles de millones de humanos y cree una nueva y enorme clase inútil, que lleve a revueltas sociales y políticas que ninguna ideología existente sabrá cómo manejar. Todos los debates sobre tecnología e ideología pueden parecer muy abstractos y lejanos, pero la perspectiva muy real del desempleo masivo (o del desempleo personal) no deja indiferente a nadie. (Harari, 2018:27)

Asumir como posible un escenario de crisis en la demanda laboral, debe servir para comenzar lo antes posible, para poder moldear el perfil de aquellos profesionales que el mercado va a requerir en los próximos años.

Mas allá de los debates estériles que se puedan suscitar, no se pueden esperar a que los acontecimientos se sucedan para reaccionar. Es pertinente que la Universidad adopte una actitud proactiva al respecto.

Por un lado ya se ha mencionado que de ser necesario se pueden implementar gradualmente nuevas opciones en los programas de estudio, que habiliten la incorporación de nuevas tecnologías y metodologías de enseñanza.

Por otro lado, se requiere identificar situaciones y soluciones prácticas y efectivas que puedan implementarse en el corto y mediano plazo. Para ello, es necesario analizar el trabajo de profesionales de otras disciplinas que ayuden a detectar las áreas de oportunidades y con los de las ciencias económicas aquellas necesidades emergentes en términos de habilidades y competencias profesionales.

Los actores con los que se tuvo un contacto estrecho durante esta etapa fueron especialistas en el área de la programación, graduados en ciencias económicas con diversos grados

de antigüedad en la matrícula, como también los estudiantes de la FCE-UNL quienes fueron fundamentales para el análisis de la potencialidad de implementar los talleres de formación.

Desde un principio se planteó como tarea generar una entrevista a un profesional informático, pero debido al interés detectado fue posible lograr un total de cuatro entrevistados. En particular se destaca la colaboración de Ing. Federico Brun.

Con respecto a profesionales graduados se realizaron 7 entrevistas las cuales se exponen, en modo resumido, a partir de la página 61.

A la vez se tuvieron contacto con jóvenes estudiantes que resultaron encuestados en un total de 207 cuyas conclusiones se exponen en desde la página 68.

Se pudo realizar una instancia de intercambio, donde fue posible compartir esta experiencia educativa dentro del seno de los Colegios Profesionales en Ciencias Económicas, por considerarlo un actor clave que promueve la capacitación permanente de sus asociados.

La dimensión laboral de la presente propuesta se vislumbra a partir del impacto que tienen las nuevas tecnologías aplicadas a las tareas rutinarias de la vida profesional, en cuanto a poder reducir las cargas sobre recursos humanos que puedan ser reemplazadas por procedimientos informatizados más rápidos y confiables por la reducción de los márgenes de error.

El manejo de lenguajes de programación es una habilidad valiosa y cada vez más demandada en el mundo laboral. Aprender a programar puede ayudar a los profesionales de ciencias económicas a analizar y visualizar grandes cantidades de datos y a desarrollar herramientas y aplicaciones que les ayuden a realizar sus tareas de un forma eficiente.

La dimensión tecnológica de la propuesta está dada por valorizar la posibilidad de achicar brechas de conocimientos entre los profesionales de ciencias económicas y los lenguajes de programación que puedan hacer de su labor cotidiana una tarea más sencilla y confortable, permitiendo poner en diálogo sus necesidades en el manejo de la información con especialistas de otras disciplinas informáticas, estadísticas, etc. a quienes lo contable les resulta un tanto lejano.

La dimensión económica apunta a reconocer qué impacto tendrá para un profesional capacitado la posibilidad de realizar tareas por sí mismo que antes estuvieran reservadas exclusivamente a expertos en informática, o bien por aquellas otras que permitan incorporar mayores prestaciones a sus servicios, o simultáneamente ganar en eficiencia y productividad con los mismos recursos.

La dimensión educativa, al interior de la FCE-UNL, están estrechamente vinculada con las asignaturas del departamento contable: Contabilidad I a V, Auditoría y Teoría y Técnica Impositiva I y II. El lenguaje de programación Python permite articular diferentes bancos de

datos que corresponden a fuentes contables y/o administrativas, permitiendo encontrar soluciones tanto de proceso como de funcionamiento de un ente y asimismo dejar abierto espacios para resolver nuevos interrogantes.

Desde el punto de vista social, la propuesta de la enseñanza de un lenguaje de programación y el impacto que generará en la relación entre profesionales abre el camino para articular espacios para compartir conocimientos y trabajos que beneficien a toda la comunidad profesional.

Es decir, el desarrollo o configuración de una solución que se obtenga para una organización o estudio contable será fácilmente adaptable a requerimientos que presenten entidades de similares características u otro colega profesional. Por tanto, la posibilidad de abrir espacios de intercambio de estas nuevas herramientas invirtiendo capacitación y tiempo por única vez se verán recompensadas por las externalidades que en el tiempo se obtengan de procesos informatizados más rápidos, confiables y seguros. Estos espacios colaborativos son muy comunes y en el campo de la programación funcionan con una llamativa generosidad interpares.

De todas formas, si bien es cierto que el escenario probable se presenta con dificultades, mucho dependerá de quienes vean en él nuevas oportunidades, que surgen del mismo proceso innovador que acecha, que el mismo autor reconoce.

> Los empleos que requieran especialización en una estrecha gama de actividades rutinizadas se automatizarán. Pero será mucho más difícil sustituir a los humanos por máquinas en tareas menos rutinarias que exijan el uso simultáneo de un amplio espectro de habilidades, y que impliquen tener que afrontar situaciones imprevistas.(Harari, 2018:32)

8. Indicadores para la evaluación de la implementación del proyecto

En consonancia con lo propuesto para la Etapa 3 «Prueba Piloto de Instancia Educativa en formato de Taller» se realizaron una serie de seis encuentros tipo clases piloto entre un grupo de docentes y otros integrantes del equipo de investigación.

De dicha instancia se analizaron como factible de incorporar los siguientes indicadores cuantitativos y cualitativos:

- Cantidad de alumnos dispuestos a participar en cada apertura de la asignatura optativa.
- Porcentaje de participación en la captación de estudiante en relación con otras alternativas optativas similares dentro de la FCE/UNL.
- Cantidad y calidad de los proyectos finales que presente cada cohorte de alumnos de la cátedra optativa, para su promoción.
- Resolución de casos: cantidad y calidad resolutiva.
- Impacto en otras cátedras del área contable (medido por incorporación de Python; incorporación de los casos; cambios metodológicos en las asignaturas; etc.)
- Encuestas de satisfacción por parte de los alumnos participantes.
- Cantidad de servicios a terceros que puedan asumir los capacitados en Python.
- Demanda en I+D para expertos en Python.

Parte II

Sustento teórico-pedagógico

Los compiladores G. Ciofalo y S. Leonzi titulan su trabajo con "Homo Communicans", pero con un agregado en forma de subtítulo con una dualidad muy sugerente: Una especie de/en evolución.

Para los autores, la combinación entre evolución antropológica y tecnológica se presenta con claras evidencias a cada paso en la vida cotidiana, dándole a la comunicación una centralidad en la sociedad contemporánea y al surgimiento de un nuevo individuo: el Homo Communicans.

La comunicación mediada por la tecnología permite nuevas formas de conectividad, de acceder a insumos pedagógicos, de interactuar con ellos y a la vez van reconfigurando las formas de aprender, incluso los lugares de aprendizaje -el propio hogar-.

Los nativos digitales ya no se anotan en listas de esperas para retirar libros de las bibliotecas, acceden desde sus camas a información inmediata a través de dispositivos cada vez mas pequeños cuyo esfuerzo puede consistir en mover algunos de sus diez dedos, e incluso la forma de interactuar que en algunos casos sólo les exige usar algunos de sus dos pulgares.

Sutil y cariñosamente bautizados por M. Serres las "Pulgarcitas" y los "Pulgarcitos" de esta era es un ser distinto quien ya no tiene «la misma esperanza de vida, ya no se comunica de la misma manera, ya no percibe el mismo mundo, ya no vive en la misma naturaleza» (2014:21) y se anima a arriesgar que «nuestros sucesores podrían encontrarse mañana tan separados de nuestra lengua como lo estamos hoy del francés antiguo practicado por Chrétieen e Troyes o Joinville» (22-23).

Las próximas generaciones que acudan a las aulas serán verdaderos "nacidos y criados digitales", con mayor cantidad de exposición y apropiación tecnológica que la de sus futuros docentes; y será responsabilidad de éstos acompañarlos a desafiar los límites de las fronteras del conocimiento de lo cual dependerá su desarrollo profesional.

Acaso no resulta imperioso que el docente ingrese al aula, ya sabiendo el significado de vocablos inexistentes pocos años atrás: **App**, **Big Data**, **ChatGPT**, **Ciberbullying**, **Gamer**, o **Podcast**, **etc.** para poder establecer relaciones fluidas con los educandos de hoy en día.

La capacidad evolutiva de la especie humana, a diferencia de la especie animal, se basa en su capacidad de actuar inteligentemente, de crear sus propias herramientas y de valerse de ellas mismas para continuar en una espiral de evolucionismo continuo.

El lenguaje o los lenguajes –orales y escritos– como medios de comunicación, son una de las mas poderosas de esas herramientas para el intercambio y desarrollo intra e intercultural.

Actualmente se ha comenzado a introducir la enseñanza de programación y de robótica en la escuela primaria, acompañando el proceso de transformación tecnológica impulsado por el advenimiento digital. Así es como se observa, cada vez a más temprana edad, cómo repercuten las nuevas tecnologías en la mutación de conductas y de la mente –de los y las Pulgarcitas en el simbolismo de Serres–, que nadie puede predecir certeramente cómo terminará. Los jóvenes que accederán en los próximos años a la universidad tendrán una mayor cercanía a la programación debido a su incorporación en los trayectos curriculares de la educación primaria y secundaria por lo que la continuidad de esta formación en la universidad será no sólo algo más accesible sino también una verdadera exigencia para la formación profesional de alumnos y docentes.

La educación universitaria debe brindar el herramental necesario para que los jóvenes desarrollen mas y nuevas habilidades tecnológicas pero también debe fomentar un pensamiento crítico y una comprensión profunda de la tecnología como una herramienta creada por la mente humana al servicio de su bienestar. Debe acompañar a los estudiantes para ser tanto usuarios competentes como creadores responsables en un mundo cada vez más digitalizado.

Por tanto, la discusión acerca de la utilidad en la disruptiva Inteligencia Artificial no debe hacer creer que la revolución mental es un efecto de la evolución tecnológica, cuando en realidad deberíamos entender que es lo contrario: «No os preguntéis qué clase de mente puede generar el uso de Google; preguntaos qué clase de mente ha generado una herramienta como Google». (Lion, 2020:22)

El ser humano es un ser creativo e ingenioso, capaz de crear herramientas y artefactos para mejorar su calidad de vida y expandir sus capacidades.

En la actualidad, la importancia otrora tuvieron la rueda o el arco y flecha, la adquieren los innumerables instrumentos digitales que nos rodean.

Estos avances han permitido una interrelación muy profunda entre el hombre y las máquinas que él mismo crea. Y fue posible que esta relación tan estrecha se genere por las formas de comunicación entre el hombre y las máquinas. A través de interfaces de usuario cada vez mas

intuitivas, comandos de voz, gestos táctiles y otros medios, los seres humanos pueden interactuar de manera eficiente y efectiva con la tecnología que han creado, siendo posible gracias a otra creación humana: los lenguajes de programación.

9. Lenguajes: su importancia en el individuo y en el entramado social

La pedagogía encuentra un aliado estratégico en la fuerza que el lenguaje le brinda al ser humano desde sus mas tempranas edades para desarrollar sus capacidades psicológicas y sociales.

Es necesario darle un valor propio al lenguaje, superador al hecho de materializar el pensamiento –en forma oral o escrita– asignándole la capacidad social de intercerder en las ideas y sentimientos del otro.

La capacidad de abstracción humana le ha permitido incorporar "los lenguajes" como ejes vertebradores de la comunicación y por ende del vínculo social.

Los lenguajes no son sólo el medio entre los emisores y receptores de la comunicación. Dewey resalta entre sus credos acerca de las materias de enseñanza el potencial sociabilizador del lenguaje en el ámbito educativo: «Cuando es tratado simplemente como un medio de adquirir información individual o como un medio de mostrar lo que ha aprendido, pierde su motivo y finalidad social» (Dewey:12). En la actualidad se debe ampliar el horizonte y asignarle al lenguaje el mismo potencial cuando esos otros actores comunicacionales, no son seres de la misma especie.

En el contexto de análisis de la educación formal a fines del siglo XIX sostenía que «En los libros de pedagogía casi siempre se trata el lenguaje simplemente como la expresión del pensamiento. Es cierto que el lenguaje es un instrumento lógico, pero es fundamental y principalmente un instrumento social»(Dewey:12).

En la era de la Inteligencia Artificial es necesario reconocerle al lenguaje su verdadero potencial sociabilizador en la relación hombre-máquina. No es sólo el dispositivo sobre el que se asienta la comunicación; es la herramienta que fortalece el vínculo entre emisor y receptor y esto está avanzando muy velozmente tanto desde la imaginación del hombre por diseñar lenguajes que las máquinas entiendan como por la capacidad autónoma de las máquinas de procesar el lenguaje humano.

Se podrá discutir si existe vinculación "social" entre el ser humano y la enorme cantidad de dispositivos electrónicos con los que interactúa a diario pero lo que es innegable aunque pueda pasar inadvertido es que si esa relación existe lo es gracias al lenguaje. «Cuando se lo trata simplemente como una forma de obtener información individual, o como un medio para mostrar lo que uno ha aprendido, pierde su motivo y fin social»(Dewey:12).

Actualmente los niños a muy corta edad, comienzan a tener acceso a dispositivos móviles

que les permiten reconocer mediante las pantallas relaciones unívocas entre el querer hacer y un estímulo recibido, y de manera imperceptible allí también existe un diálogo mediado por un lenguaje.

En 2010, la Universidad de Murcia concedió a Jesús García Molina, la responsabilidad del Laudatio in honorem del doctor Alan C. Kay, a quien presentó como un investigador comprometido en que los avances tecnológicos permitan facilitar el aprendizaje y mejorar la creatividad en los niños desde muy temprana edad. A finales de la década de 1960 Alan Kay merodeaba la idea de ordenadores pequeños, dinámicos y portátiles similares a una libreta pero con capacidad de almacenar una gran biblioteca y conectados en red a los que denominó Dynabook.

Alan Kay ha estudiado la forma en la que los niños aprenden y sostiene que «...las sociedades alfabetizadas no son "sociedades orales con sistemas de escritura" sino sociedades que piensan de un modo cualitativamente diferente de las sociedades orales. En otras palabras, la escritura no es tan sólo un registro del discurso oral sino aprender y producir lectura y escritura provoca cambios grandes en el propio proceso de pensamiento».

Por ello es necesario generar espacios educativos que otorguen facilidades a las personas desde sus primeros años, para comunicarse y aprender y adquirir rápidamente un cúmulo en su vocabulario que lo fortalezca en su ida y vuelta comunicacional.

Pero aún mas allá esa capacidad de sociabilización, abarcativa de la integración y coordinación entre seres de su misma especie, la humanidad la ha logrado con entes electrónicos que fueron diseñados y creados por el mismo ser humano.

Esa capacidad de analizar un problema-necesidad, de imaginarse herramientas y construirlas, de aplicar leyes y procedimientos a seguir, han valido tanto para el desplazamiento físico con el invento de la rueda, como para la vinculación entre emisor, receptor, mensaje y medio de comunicación.

> La escritura nació como la criada de la conciencia humana, pero cada vez más se está convirtiendo en su dueña y señora. Nuestros ordenadores tienen dificultades para comprender cómo Homo Sapiens habla, siente y sueña. De manera que enseñamos a Homo Sapiens a hablar, sentir y soñar en el idioma de los números, que los ordenadores puedan comprender. (Harari, 2014: 103).

Resulta cotidiano, desde el rol de usuarios informáticos, distinguir la diferencia entre hardware y software, sin embargo existe todo un proceso de comunicación entre los componentes involucrados que suele pasar inadvertido.

En el desafío autoimpuesto, Kay puso en marcha también la creación de un lenguaje de programación: «Pero Smalltalk no era sólo un lenguaje, era también un entorno de programación y su creación exigió una importante investigación en el terreno de la interacción hombre-ordenador, ya que se pretendía conseguir un entorno amigable y confortable en el que se combinase texto con gráficos y animación».

Reconociéndose influenciado por el enfoque de aprendizaje activo de la pedagoga italiana María Montessori la investigación de Kay se centró en descubrir cómo los niños aprenden, «ya que se pretendía usar Smalltalk para enseñarles a pensar y a adquirir destrezas como la lectura, aprender música o adquirir conocimientos de física, química o matemática».

Por ello, la tarea por delante que tiene este proyecto educativo, es presentarse como una gran oportunidad para los educandos universitarios que estén dispuestos a ampliar sus conocimientos en un entorno dominado por el «aprender haciendo» y que seguramente les deparará grandes satisfacciones.

Como propone Sweigart en su muy didáctico libro para principiantes Automatizar las cosas aburridas con Python:

> Para la mayoría de las personas, su computadora es solo un dispositivo en lugar de una herramienta. Pero al aprender a programar, obtendrás acceso a una de las herramientas más poderosas del mundo moderno y te divertirás en el camino. (Sweigart:10).

Retomando el reconocimiento que le hiciera la Universidad de Murcia en palabras de García Molina:

> Alan Kay fue un visionario que se planteó construir máquinas pequeñas, baratas, de uso personal y fáciles de usar para todo el mundo, incluidos los niños.
> Esta concepción del ordenador era sustentada por su visión como una máquina capaz de simular o representar cualquier sistema del mundo real o imaginario, siempre que su comportamiento pudiese ser descrito en términos de un lenguaje ejecutable.
> Alan Kay diseñó Smalltalk en torno a una idea fascinante: la metáfora de objetos que intercambian mensajes con el propósito de colaborar entre ellos en el cumplimiento de una operación. Unos pocos conceptos se combinan de forma armoniosa para crear un lenguaje elegante y simple pero con una gran potencia expresiva, que lo convierte en una verdadera obra de arte, una de las construcciones más bellas de la Informática.
>
> Del Laudatio

El camino iniciado por Kay abrió las puertas para que hoy en día la mayoría de los

programadores realicen sus tareas con la lógica ideada en los sesentas. También surgen de allí el origen de lo que es el trabajo con ventanas superpuestas o el uso de un “mouse” para dar instrucciones sobre una lista desplegable de opciones en la pantalla.

La potencia comunicacional de los lenguajes no ha quedado circunscripta sólo a la relación entre las personas sino que la inteligencia humana ha llegado en un «corto» tiempo evolutivo a crearse herramientas lingüísticas aptas para el intercambio entre seres humanos y computadoras hasta tal punto que éstas ya tienen habilidades y capacidades que superan las de cualquier ser vivo.

En estas capacidades, dónde las máquinas realizan las tareas de forma mas precisa y rápida, es donde aparecen oportunidades para que la carga humana tenga lugar a un relax. Hacia esto se apunta con ampliar las capacidades y competencias del profesional en Ciencias Económicas.

9.1. La comunicación entre seres humanos y ordenadores en la era de la inteligencia artificial.

Se puede generalizar que un lenguaje de programación es una estructura de reglas que se utiliza para escribir y ejecutar código informático. Estos lenguajes son utilizados para dar instrucciones a la computadora sobre qué debe realizar.

Un conjunto de líneas de código permiten la creación de programas y aplicaciones para ser utilizada en ordenadores y otros dispositivos.

Tanto los lenguajes de comunicación entre personas como los lenguajes de programación tienen un conjunto de reglas y sintaxis específicas que deben seguirse al utilizarlos.

Sin embargo, los lenguajes entre personas mas allá de su correcta sintaxis gramatical son mucho mas flexibles. Por ejemplo, una persona podrá interpretar indistintamente el significado de las siguientes frases «al lado de la casa blanca», «al lado de la blanca casa», e incluso «al lado de la Casablanca».

Por el contrario los lenguajes de programación tienen una mayor rigidez y menor margen de error por lo que hay que seguir una estricta serie de reglas sobre cómo se deben escribir las instrucciones y cómo deben estructurarse.

También en otras áreas de aplicación es posible encontrar ejemplos donde surge la necesidad de pensar en otro tipo de lenguajes que el ser humano es capaz de interpretar.

Como sostiene la doctora Alicia Dickenstein, en una reciente entrevista brindada al

periódico Infobae:

> La matemática también tiene un lenguaje propio. Y si uno no entiende el lenguaje de la matemática, eso se convierte en una obstrucción para poder pensar. Entiendo que quizás es más difícil que otras disciplinas, pero siempre insisto con la idea de que la matemática está grabada en nuestro cerebro. Habrá quienes tengan más o menos capacidades, más o menos interés, pero la matemática es universal y es tan inherente al ser humano como hablar o escribir. (Blanco, 2023:párr. 13)

Si se le propone a una persona que piense en todos los números naturales posibles entre el número mil y el diez mil seguramente no podrá retenerlos en su cabeza pero es posible ayudarse del poder de la abstracción para descomponer sus partes y pensar en el todo, pero el gran facilitador para representar su respuesta y poder transmitirla sin tener que escribir todos y cada uno de los elementos comprendidos, es el lenguaje con el que pueda expresarla.

Es exigible para un ingresante universitario estar en condiciones de decodificar lo siguiente y evaluar la validez de la siguiente expresión como posible respuesta:

$$\{ \, n \in \mathbb{N} \mid 1.000 < n < 10.000 \, \}$$

Se trata de poner en contexto la capacidad intrínseca de la humanidad de transmitir información acerca de cosas inmateriales, que no pueden ser vistas o tocadas. Como sostiene Harari: «Hasta donde sabemos, solo los sapiens pueden hablar acerca de tipos enteros de entidades que nunca han visto, ni tocado ni olido. Esta capacidad de hablar sobre ficciones es la característica más singular del lenguaje de los sapiens». (2014:26).

A la vez, hay que hacer una pequeña introducción –producto de la era digital– en la cual se luce el lenguaje binario como la base sobre la cual se construye la programación, siendo esencial para el desarrollo y la ejecución de software en los sistemas informáticos modernos.

Por mas sencillo que parezca hoy en día enviar una fotografía desde un teléfono móvil a otro, sin soporte físico alguno de por medio, esta acción encierra todo un desarrollo de innovaciones en campos como la miniaturización de componentes electrónicos, el desarrollo de redes de comunicación inalámbricas, y la evolución de algoritmos de compresión de datos.

Desde el nivel más básico de la representación binaria de datos hasta la complejidad de los algoritmos de compresión que permiten la transferencia eficiente de archivos, el lenguaje binario se erige como el fundamento subyacente que posibilita la transmisión y recepción de información digital, delineando así la trayectoria misma de la revolución tecnológica que vivimos.

> El cambio de era que estamos viviendo está relacionado con la nueva capacidad de los humanos de expresarse con unos y ceros. No importa si se trata de un texto, un sonido, o una imagen, estática o en movimiento, sea lo que sea lo podemos codificar en lenguaje binario y transmitirlo a cualquier parte del mundo casi al instante. (Roca, 2015:13-14)

No se trata de cargar sobre el currículum universitario de la FCE-UNL con conocimientos específicos propios de otras especialidades, sino sólo dejar sentado los mecanismos de intercomunicación entre seres humanos y ordenadores.

La reciente irrupción de **ChatGPT** también merece un exhaustivo análisis acerca de su utilidad en el campo educativo, inexistente al momento de iniciar el proyecto educativo. Rápidamente logro instalarse en los medios de comunicación y entre la comunidad educativa.

Siendo una herramienta de uso libre y gratuito –al menos por ahora– el veredicto final lo tendrán los millones de usuarios que se animen a interactuar con la «inteligencia artificial» para sus propios fines.

Es posible anticipar, que no recibirá por parte de los usuarios una calificación única, unívoca y coincidente. Por un lado hay que advertir que aún presenta fallas en sus respuestas a temas puntuales, pero por otro lado no se puede dejar de destacar el gran trabajo de búsqueda y síntesis que realiza sobre una inabarcable fuente de información.

La sigla GPT deriva del inglés “Generative Pre-Trained Transformer” lo que pudiera llevarse al castellano como Transformador Generativo Preentrenado. Con sólo usarlo es posible ir aceptando su definición dentro de un canal de diálogo de ida y vuelta: el ya mas conocido “Chat”.

Las advertencias sobre las posibles consecuencias del uso de la inteligencia artificial ya habían llegado años atrás desde Hollywood, así que tampoco demoraron en llegar a los tribunales de los EE.UU. los pedidos para que se suspenda la aplicación de esta nueva herramienta de la vida real.

> El campo de la inteligencia artificial busca crear un nuevo tipo de inteligencia basada únicamente en la escritura binaria de los ordenadores. Filmes de ciencia ficción como Matrix y Terminator hablan de un día en el que la escritura binaria se libera del yugo de la humanidad. Cuando los humanos intentan recuperar el control de la escritura rebelde, esta responde intentando eliminar a la raza humana.(Harari, 2014:103)

A modo de ejemplo se transcribe un diálogo con ChatGPT a partir de una simple consulta utilizándolo como herramienta de ayuda para manipular archivos de origen Microsoft

Excel®mediante el lenguaje de programación Python.

Consulta:

Como se puede importar un archivo de Excel llamado “Ejemplo.xls” ubicado en la carpeta “C:Documentos” para convertirlo en una base de datos utilizando Python y mostrar su contenido en pantalla?

Respuesta:

Puedes hacerlo utilizando pandas para leer el archivo Excel y mostrar su contenido en pantalla

import pandas as pd

Ruta completa al archivo Excel, incluida la carpeta:

ruta_excel = r’C:\Documentos\Ejemplo.xls

Leer el archivo Excel y cargarlo en un DataFrame de pandas

dataframe = pd.read_excel(ruta_excel)

Mostrar el contenido del DataFrame en pantalla

print(“Contenido del DataFrame:”)

print(dataframe)

Cabe resaltar que la demora en la respuesta fue menor a los dos segundos.

Detrás de este boom disruptivo que se ha generado con una rápida expansión de la inteligencia artificial existen empresas a nivel global que van cobrando notoriedad. Uno de los mas importantes ejemplos es lo que viene sucediendo con NVIDIA, una firma que originalmente comenzó siendo líder en tecnología de procesamiento gráfico (GPU) y unidades de procesamiento acelerado (APU). Actualmente desarrolla y fabrica hardware y software para gráficos de alta definición, gaming, y está haciendo una importante apuesta en torno a la inteligencia artificial (IA), computación de alto rendimiento (HPC) y visualización profesional.

Estas reorientaciones son detectadas por el mercado, y a comienzos de 2024 ha experimentado un muy fuerte aumento en su cotización bursátil debido a su expansión en el mercado de la inteligencia artificial y centros de datos, su continua innovación tecnológica y sus

adquisiciones estratégicas.

En una reciente entrevista brindada en la Cumbre Mundial de Gobiernos, su CEO Jensen Huang explicaba con firmeza:

> Quiero decir algo, y va a sonar completamente opuesto a lo que siente la gente. En el transcurso de los últimos 10 o 15 años, casi todos los que se sientan en un escenario como este dirían que es vital que sus hijos aprendan informática: todos deberían aprender a programar. Y, de hecho, es casi exactamente lo contrario. Nuestro trabajo es crear tecnología informática de modo que nadie tenga que programar y que el lenguaje de programación sea humano. Todo el mundo en el mundo es ahora programador. Éste es el milagro, éste es el milagro de la inteligencia artificial. Por primera vez hemos cerrado la brecha: la brecha tecnológica se ha cerrado por completo. Y esta es la razón por la que tanta gente puede involucrarse con la inteligencia artificial, es la razón por la que cada gobierno, cada conferencia industrial, cada empresa habla hoy de inteligencia artificial porque por primera vez puedes imaginar que todos en tu empresa sean como tecnólogo, este es un momento tremendo para que todos ustedes se den cuenta de que la brecha tecnológica se ha cerrado o, de otra manera, el liderazgo tecnológico de otros países ahora se ha restablecido. Los países, las personas que entienden cómo resolver el problema del dominio en biología digital o en la educación de los jóvenes o en la manufactura o en la agricultura, aquellas personas que entienden la experiencia en el dominio. Ahora puede utilizar la tecnología que ahora está disponible para usted. Ahora tiene una computadora que hará lo que usted le diga que haga para ayudarlo a automatizar su trabajo, amplificar su productividad y hacerlo más eficiente, por lo que creo que este es un momento tremendo, el impacto, por supuesto, es grande y es imperativo Activar y aprovechar la tecnología es absolutamente inmediato y además darse cuenta de que utilizar la Inteligencia Artificial es mucho más fácil ahora que en cualquier otro momento de la historia de la informática. Es vital que mejoremos las habilidades de todos y creo que el proceso de mejora de habilidades será delicioso y sorprendente darse cuenta de que esta computadora puede realizar todas estas cosas que usted le está indicando que haga y hacerlo con tanta facilidad. Summit«A Conversation with the Founder of NVIDIA: Who Will Shape the Future of AI» https://youtu.be/8Pm2xEViNIo?t=1109

Este pensamiento plantea una perspectiva interesante sobre el papel de la informática y la programación en la era de la inteligencia artificial (IA), soslayando la necesidad de que todos aprendan a programar. Existen notables avances en la creación de tecnología informática que disminuye la necesidad de que las personas programen directamente, debido a la accesibilidad tecnológica aun para quienes no tienen habilidades técnicas avanzadas en programación. Sin embargo, el potencial transformador de la inteligencia artificial al cerrar la brecha tecnológica y permitir que personas de diversos ámbitos puedan utilizarla para mejorar su productividad y eficiencia en diferentes sectores, como la educación, permite ampliar las oportunidades para las personas en todo el mundo. Es por ello, que lo que en el fondo se puede sostener sin lugar a

dudas es la idea que la Inteligencia Artificial vino para ayudar en el uso de las herramientas informáticas, lo que ofrece oportunidades emocionantes para aquellos que estén dispuestos a explorar y aprender.

10. El enfoque pedagógico.

La incorporación curricular de un lenguaje de programación en las carreras relacionadas con las Ciencias Económicas permitirá a los estudiantes asumir un rol activo, toda vez que en una instancia inicial, se propone como una materia optativa dentro de la amplia oferta académica.

Se resalta nuevamente el carácter optativo e introductorio con el que se puede materializar inicialmente este proyecto educativo.

Aprender a programar también puede ayudar a los profesionales de ciencias económicas a desarrollar habilidades de resolución de problemas y pensamiento lógico. Estas habilidades son importantes para muchos trabajos en el campo de las ciencias económicas y pueden ser útiles en muchas otras áreas.

Una aproximación a la programación puede ser un experiencia desafiante para algunos estudiantes, especialmente si no tienen conocimientos previos en la programación. Es importante tener en cuenta esto y ofrecer apoyo y recursos para que los estudiantes puedan tener éxito en el curso.

Finalmente, es importante elegir el lenguaje de programación adecuado para incluir en la currícula. Algunos lenguajes son más adecuados para determinados propósitos que otros, por lo que es importante investigar y elegir el lenguaje que mejor se adapte a las necesidades de los estudiantes y a los objetivos del curso.

Ya se ha caracterizado a los jóvenes y a sus formas de relacionarse. Los estudiantes universitarios también pretenden que el proceso de enseñanza les sea accesible, inmediato y de utilidad.

No obstante son atravesados por las dudas que les impone la «modernidad líquida». Este concepto ideado por Zygmunt Bauman describe la transición de una sociedad sólida y estructurada hacia una sociedad caracterizada por la fluidez, la incertidumbre y la falta de estabilidad en múltiples aspectos de la vida.

Bauman argumenta que esta característica de la sociedad contemporánea tiene profundas implicaciones en la forma en que vivimos nuestras vidas y nos relacionamos con los demás.

En este contexto, la comunicación es esencial para adaptarse a un entorno en constante cambio. Los jóvenes se ven obligados a estar conectados de manera continua para mantenerse actualizados, incorporarse a redes, tomar decisiones rápidas y navegar en un mundo caracterizado por la volatilidad, la proliferación y creación de contenidos.

Para tener éxito en la incorporación de lenguajes de programación en la currícula

universitaria del profesional de ciencias económicas, y tomando como una oportunidad lo expresado en el párrafo anterior, es importante poner a disposición de los estudiante los siguientes apoyos y recursos:

- Recursos de aprendizaje: Es importante proporcionar recursos de aprendizaje adecuados, como tutoriales en línea, libros y otros materiales de referencia. Estos recursos deben estar diseñados de manera que sean accesibles y comprensibles para estudiantes sin experiencia previa en la programación.

- Asesoramiento y apoyo individualizado: Para aquellos estudiantes que encuentren especialmente difícil aprender a programar, es importante proporcionar asesoramiento y apoyo individualizado. Esto puede incluir sesiones de tutoría o apoyo de un profesor o tutor experto en la programación.

- Ejemplos y proyectos prácticos: Proporcionar ejemplos y proyectos prácticos puede ayudar a los estudiantes a aplicar lo que han aprendido y a desarrollar habilidades prácticas. Estos proyectos deben estar diseñados de manera que sean relevantes y desafiantes para los estudiantes.

- Oportunidades de colaboración y discusión: Fomentar la colaboración y la discusión entre los estudiantes puede ser una manera efectiva de ayudar a los estudiantes a comprender mejor la programación y a resolver problemas. Esto puede incluir grupos de estudio o discusiones en línea.

Este esquema de recursos, fue probado durante la realización del proyecto de investigación, y permite plantearles a los estudiantes una forma de aprender en un ambiente innovador, de cambio constante, cooperativo y globalizado.

La acción pedagógica y la metodología del aprendizaje que se propone busca mostrar de una manera optimista la aparición de nuevas exigencias en la formación profesional.

Si por un lado la modernidad líquida fomenta el individualismo y la fragmentación de las identidades, la aparición de entornos colaborativos, abiertos y gratuitos en el mundo de los programadores permite advertir que gran parte de las soluciones dependen de la capacidad de búsqueda.

Si la configuración de las redes sociales en connivencia con las adicciones derivadas de lo que se exhiben en las pantallas hacen presumir de una pérdida de tiempo, éstas también pueden ser fuente de información valiosa en tiempo real.

Martin Hilbert es un experto alemán en redes digitales y doctor en comunicación, conocido también como “el gurú del Big Data” afirma que desde el 2014 hasta 2017, se creo

tanta información como desde la Prehistoria hasta el 2014, y que la tasa de duplicación de la información es de dos años y medio.

Está claro que esta otra característica de la sociedad del conocimiento actual donde la calidad de la comunicación queda afectada por la veracidad, velocidad y la cantidad de datos que se comparten, cae como una pesada mochila para los jóvenes estudiantes.

Esta cantidad enorme de informaciones lo que se conoce como Big Data (macrodatos, o datos masivos o a gran escala), no son abarcables por los procesadores tradicionales de datos, ya que, por su número presentan problemas de gestión, almacenamiento, análisis, búsqueda y visualización, entre otros. Y es este "mundo de datos" el que nos rodea día a día.

En la era del Big Data, la cantidad de información disponible es abrumadora. El "homo communicans" debe lidiar con grandes volúmenes de datos para tomar decisiones informadas. La habilidad para procesar, analizar y comprender esta información se vuelve esencial para navegar en un mundo de datos masivos.

Hay un punto de conexión entre el "homo communicans" y "la modernidad líquida" en la era del Big Data, ya que ambos reflejan la importancia de la comunicación en un mundo caracterizado por la fluidez, la incertidumbre y una creciente dependencia de la tecnología de la información para la toma de decisiones eficientes. El desafío que presentan es el de saber adaptarse y aprender en forma permanente.

Este horizonte es el que debe abrir la puerta para que la programación entre en la currícula de las Ciencias Económicas.

10.1. Los orígenes del Aprendizaje basado en problemas

El aprendizaje basado en problemas (ABP o también PBL, por sus siglas en inglés) es un enfoque de enseñanza y aprendizaje que se centra en el estudiante y en el proceso de resolución de problemas. El ABP se originó en la década de 1960 en la Universidad de Stanford, cuando un grupo de profesores liderados por David Ausubel y John Dewey comenzaron a explorar nuevas formas de enseñanza que fueran más significativas y retadoras para los estudiantes.

La Facultad de Ciencias Médicas de la Universidad Nacional del Litoral (FCM-UNL) es un claro ejemplo en la utilización de esta metodología, adoptando estándares pedagógicos de otras Facultades de medicina del mundo. Sin embargo, dentro de la FCE-UNL es una modalidad cuasi reservada para las asignaturas Taller de Práctica Integradora II y Prácticas Profesional Supervisada del Ciclo de Formación Especializada (4º y 5º año).

Larisa Carrera, ex decana de la FCM-UNL y actual vicerrectora de la UNL, sintetiza

el rol de los docentes bajo esta modalidad educativa, lo cual constituye una oportunidad para explorar esta metodología tan desafiante a nivel docente:

> En este modelo el docente asume el rol estimulador, guía y orientador del aprendizaje. El punto de partida del aprendizaje es la indagación y la identificación de los problemas de la práctica médica habitual. La participación activa del sujeto en la construcción del conocimiento le permite desarrollar capacidad de deducir y relacionar, de lograr articular los conceptos provenientes de diferentes disciplinas en el proceso de formulación de hipótesis y elaboración de síntesis, esta metodología tiene el objetivo de estimular la discusión, el diálogo, la reflexión y la participación de todos los estudiantes. (2013:160)

El profesor chileno Juan Iglesias señala que el ABP surgido y difundido a partir de la experiencia de la Universidad de Mac Master (Canadá), consta de dos fases:

> En la primera, se trata de revitalizar el proceso de enseñanza-aprendizaje en el aula, a fin de que los estudiantes puedan efectuar una mayor cantidad de estudios independientes. En la segunda fase se organizan sesiones basadas en problemas en pequeños grupos, en los que los estudiantes analizan problemas simulados de la vida real que se programan semanalmente. Todo ello facilita el aprendizaje de las disciplinas, con unas pocas lecciones magistrales y un mayor número de estudios optativos y autodirigidos. (2002:83)

Los especialistas señalan cierta diferenciación teórico-práctica entre el aprendizaje basado en problemas (ABP) y el método de estudio de casos (MEC). Sin embargo un análisis superficial permite asignarle a estas dos metodologías pedagógicas fortalezas suficientes como para constituirse en ejes vertebradores de la propuesta. Es posible anticipar que ninguna prevalece notorariamente sobre la otra y que en algunos aspectos ambas coinciden, y en otros una prevalecerá conceptualmente sobre la otra de manera muy sutil cuando se trata de aplicarlas a una instancia educativa de un lenguaje de programación en las carreras de Ciencias Económicas.

El «problema» es el elemento básico para generar el estímulo del aprendizaje. El aprendizaje basado en problemas supone el reconocimiento implícito de que los estudiantes pueden aprender por sí mismos con la guía del tutor.

Algunos expertos sostienen que el MEC es un complemento al ABP. Según la pedagoga y escritora estadounidense Selma Wassermann, una reconocida experta en la metodología, en este tipo de aprendizaje «los casos son instrumentos educativos complejos que revisten la forma de narrativas». (19)

Dichas narrativas se generan en torno a problemas o situaciones que ocurren en la vida real –aunque también pueden ser ficticios–. De este modo, los casos se centran en distintas materias y asignaturas y cada uno de ellos incluye información y datos que activa la enseñanza a través de la investigación y la discusión del problema o caso.

Sin embargo, como ya se ha expuesto, existen similitudes entre el ABP y el estudio de casos; en ambos se fomenta el aprendizaje colaborativo y es necesario un rol docente que oriente a los estudiantes en la autodeterminación del proceso de enseñanza-aprendizaje, y en ambos predomina el modelo constructivista.

Por su lado, aunque no sean sustanciales se advierten también aspectos diferenciales que se sintetizan en el siguiente cuadro:

Aspectos a comparar	Estudio del Caso	Aprendizaje Basado en Problemas
Situación descripta	Auténtica y Concreta	Real o ficticia
Interrogantes a responder	El por qué y el cómo se ha dado el caso	Si tiene solución, cuál podría aplicarse. Cómo y en qué se aplica lo aprendido
Análisis de la misma	Desde el análisis individual al grupal	En grupo con diferenciación individual
Características de la resolución	Múltiples soluciones	Múltiples soluciones
Rol del docente	Encaminar la investigación en la dirección correcta. Imparte el marco teórico y orienta las hipótesis de investigación	Tutoriza la búsqueda de la información y orienta el proceso de solución
Interacción con el Alumno	Toda la clase trabaja separada. Primero, individualmente y, a continuación, en grupos	Se tutoriza a cada uno de los grupos por separado
Lugar de trabajo	Normalmente en el aula y en horas lectivas.	Mixta, dentro y fuera del aula
Duración de las clases	Puede trabajarse en una sola sesión o en varias	Más de una sesión de clase y más de una tutoría
Información brindada	Se brinda casi toda la información necesaria	Generalmente los alumnos tienen que ampliar la información

Sintéticamente, estos modelos se diferencian en que mientras en el MEC los alumnos requieren de una formación previa para poder dar opinión fundada sobre el tema que se les presenta, en el ABP los alumnos están recién siendo alfabetizados en cierto tema.

Para el especialista Peter Bouhuijs el ABP requiere de una «presión exterior y de un fuerte liderazgo interno». Tomar la recomendación surgida de la consulta a la comunidad profesional en tiempos de análisis y evaluación del plan de estudios y generar masa crítica dentro de la comunidad de la FCE en el marco del Proyecto de Investigación mencionado conjugan con estos dos requisitos.

«El común denominador en estos intentos es el énfasis en un aprendizaje activo usando un problema como estimulo y punto de partida para el proceso de aprendizaje». (Bouhuijs, 2011:18)

Es decir, la propuesta se inserta en el contexto de una metodología didáctica ABP, metodología que hace énfasis en que el alumno pasa a ser el auténtico eje de la educación universitaria, trabaja en grupo y en sesiones de tutoría con problemas diseñados especialmente para lograr los objetivos propuestos.

Requiere que los estudiantes se involucren en una formación auto-dirigida en la que deben tomar ellos mismos la iniciativa para identificar los elementos necesarios para entender mejor el problema y detectar el lugar donde hallar la información necesaria. Es el propio grupo de alumnos quien analiza el problema planteado, establece sus propios objetivos de aprendizaje, realiza las búsquedas bibliográficas pertinentes y las utiliza para formalizar sus respuestas.

De esta manera el ABP proporciona las habilidades para un aprendizaje a lo largo de toda la vida. Mientras tanto el profesor deja de ser la fuente del conocimiento y abandona su rol de transmisor para convertirse en un guía y facilitador del proceso de aprendizaje. Esta metodología se orienta a: mejorar el rendimiento académico de los alumnos; propender hacia el aprendizaje autónomo; favorecer el mayor protagonismo por parte de los alumnos y el mejoramiento de sus hábitos de estudio en el trabajo intelectual.

El ABP debe capacitar a los alumnos a poder aprender por sí mismos, desafiándolos a alcanzar niveles cognitivos más altos de comprensión.

Claro está que esta situación exige desafíos particulares, como la cooperación entre disciplinas desde el punto de vista de la articulación curricular en pos de establecer objetivos y metas comunes, lo que significa un cambio en la cultura académica en si mismo.

Insiste Bouhuijs en el liderazgo docente, y en que es necesario un tiempo prudencial para su instauración. Los resultados del proyecto de investigación «Sinergia entre la práctica

profesional y Python como lenguaje de programación» presentados en mayo de 2023 dan cuenta de la formación de cierta masa crítica entre los integrantes del mismo como para pensar en un futuro cuerpo docente capacitado en esta metodología didáctica.

Los casos propuestos en cada uno de los talleres exigen que el docente se presente –en términos de Bouhuijs– como un promotor del aprendizaje mas que un presentador de conocimientos.

Quizás el planteo hacia lograr una materia optativa no sea lo suficiente como para instaurar un nuevo enfoque metodológico pero podrá ser un primer intento de articulación con otras cátedras, incluso con docentes de la Facultad de Ciencias Hídricas de la UNL donde se ofrece la carrera de Ingeniería Informática.

Esta forma de presentación del problema a resolver se citan al comienzo de cada uno de los ejercicios prácticos propuestos para las clases como se puede observar a partir de la página 122 en la sección correspondiente a la acción propuesta de Redacción de problemas y casos a los fines de presentarlos en las instancias educativas.

10.2. Justificación del enfoque para el Proyecto Educativo.

El surgimiento del Aprendizaje Basado en Problemas (ABP) en la enseñanza universitaria se encuentra enraizado en una serie de desarrollos pedagógicos y filosóficos a lo largo del tiempo.

El ABP se originó en la Facultad de Medicina de la Universidad de McMaster en Canadá en la década de 1960 y tuvo un éxito notable en la formación de médicos.

A medida que los resultados exitosos del ABP se hicieron evidentes en la educación médica, esta metodología se extendió a otras disciplinas universitarias. Los educadores reconocieron que el enfoque en la resolución de problemas se podía aplicar de manera efectiva a una variedad de campos, desde las ciencias sociales hasta las ciencias de la computación.

El ABP se encuentra fuertemente influenciado por las teorías del aprendizaje constructivista, que destacan la importancia de que los estudiantes construyan su conocimiento activamente a través de la exploración y la resolución de problemas.

La metodología se alinea con filósofos como Jean Piaget y Lev Vygotsky, cuyos enfoques pedagógicos enfatizan la construcción del conocimiento por parte del estudiante.

A medida que el mundo se volvió más orientado hacia la tecnología y la globalización, las universidades se dieron cuenta de la importancia de enseñar a los estudiantes habilidades como la resolución de problemas, el pensamiento crítico, la comunicación efectiva y la colaboración.

El ABP se alinea naturalmente con la necesidad de desarrollar estas habilidades esenciales para el siglo XXI.

El ABP se ha adaptado para acomodar una amplia gama de estilos de aprendizaje y niveles de habilidad. Los estudiantes pueden abordar los problemas a su propio ritmo y profundidad, lo que facilita la inclusión de diferentes tipos de estudiantes en el proceso de aprendizaje.

Se ha demostrado que los estudiantes que se benefician de esta metodología tienden a retener mejor el conocimiento, a desarrollar habilidades más sólidas y a estar más comprometidos en su aprendizaje.

Esta metodología ha evolucionado con los avances tecnológicos, y es potenciada por la creación de entornos virtuales de aprendizaje que facilitan su implementación. La tecnología también ha ampliado el acceso a una variedad de recursos educativos en línea, enriqueciendo la experiencia de aprendizaje basado en problemas.

Si bien en sus comienzos el ABP en la enseñanza universitaria surgió como una respuesta efectiva a la necesidad de formar a los estudiantes en habilidades críticas, promoviendo el aprendizaje activo y contextualizado en la medicina; en la actualidad su posterior expansión a otras disciplinas reflejan su versatilidad y su capacidad para adaptarse a las necesidades cambiantes de la educación superior en el siglo XXI.

10.3. El ABP y la educación informática

Los resultados del proyecto de investigación permiten extrapolar una experiencia de este tipo para una instancia educativa para el aprendizaje de un lenguaje de Programación (Python) en la FCE-UNL.

La enseñanza de un lenguaje de programación con la metodología de aprendizaje basado en problemas ofrece una serie de ventajas pedagógicas significativas que benefician tanto a los educadores como a los estudiantes. Esta metodología encuentra soporte en la gran cantidad de información disponible en distintos formatos.

Esta característica de un aprendizaje activo y contextualizado, se sostiene en la existencia de gran cantidad de fuentes de información virtuales no tradicionales donde cobra relevancia la capacidad de búsqueda y selección por parte alumno.

En su mayoría en soporte virtual, coexisten numerosos divulgadores –reconocidos y anónimos–, material audiovisual en diferentes idiomas, blogs de preguntas y respuestas colaborativos hasta páginas en internet con desafíos de programación abiertos. Discernir acerca de la utilidad entre tanta información se constituye en una habilidad a desarrollar en sí misma.

Por lo tanto este es un marco propicio para proponer en forma introductoria problemas de la vida real contextualizados y habilitar el aprendizaje autónomo aplicando lo que han descubierto y desarrollar habilidades de pensamiento crítico y resolución de problemas.

Existiendo múltiples posibilidades de resolución de los problemas, cada alumno podrá plantear caminos diferentes lo que también enriquece el proceso de enseñanza-aprendizaje del conjunto.

Enseñar un lenguaje de programación con la metodología de aprendizaje basado en problemas (ABP) es una estrategia pedagógica altamente efectiva que ofrece numerosas ventajas tanto para los educadores como para los estudiantes. Las razones más importantes para justificar esta elección:

El ABP permite a los estudiantes aprender programación en un contexto relevante y práctico. Al abordar problemas reales o simulados, los estudiantes pueden ver la aplicación directa de las habilidades que están adquiriendo. Esto les proporciona un sentido claro de propósito y motivación, lo que aumenta su compromiso y retención de conocimiento.

La programación es, en esencia, la resolución de problemas algorítmicos. Mediante el ABP, los estudiantes aprenden a identificar, descomponer y resolver problemas de manera efectiva, habilidades que son valiosas en una amplia gama de disciplinas y en la vida cotidiana.

El ABP fomenta un enfoque activo en el proceso de aprendizaje. Los estudiantes se involucran en la búsqueda de soluciones, lo que les permite construir su comprensión de los conceptos de programación de manera más significativa. Esto contrasta con los métodos de enseñanza pasivos que a menudo resultan en una mera memorización sin comprensión real.

La programación en el mundo real generalmente implica trabajo en equipo y virtualizado. La metodología ABP promueve la colaboración entre estudiantes al enfrentar problemas complejos. Esto refleja la realidad laboral en el campo de la tecnología, donde la colaboración y la comunicación son esenciales.

El ABP permite a los educadores adaptar el contenido de acuerdo con las necesidades y el nivel de los estudiantes. Pueden seleccionar problemas de diferentes niveles de dificultad y ajustar el enfoque según el progreso de la clase. Esto facilita la diferenciación y la personalización

del aprendizaje y la escalabilidad.

La programación requiere pensamiento lógico, análisis y creatividad. Al enfrentar problemas con múltiples soluciones posibles, los estudiantes desarrollan estas habilidades, lo que les ayuda a abordar desafíos de manera innovadora y efectiva.

La programación es una habilidad fundamental en la sociedad actual. El ABP prepara a los estudiantes para el mundo digital y les brinda una ventaja competitiva en el mercado laboral, donde la demanda de habilidades en programación sigue creciendo.

El ABP facilita la evaluación auténtica de las habilidades de programación de los estudiantes, ya que se basa en la resolución de problemas reales en lugar de exámenes teóricos. Esto permite una evaluación más precisa y significativa del conocimiento y las habilidades adquiridas.

Los estudiantes suelen encontrar la resolución de problemas prácticos y desafiantes inherentemente motivadora. El ABP despierta la curiosidad y el interés de los estudiantes, lo que fomenta una motivación intrínseca para aprender y mejorar.

Los procesos cognitivos son las capacidades mentales que nos permiten adquirir, procesar, almacenar y utilizar información. Estos procesos son esenciales para aprender cualquier cosa, incluyendo un lenguaje de programación.

Aprender un lenguaje de programación requiere una serie de procesos cognitivos, como la atención, la memoria, el razonamiento y la resolución de problemas. La atención es la capacidad de de enfocarse en una tarea específica y bloquear distracciones. La memoria es la capacidad de almacenar y recuperar información. El razonamiento es la capacidad de usar la lógica para resolver problemas y tomar decisiones. La resolución de problemas es la capacidad de encontrar soluciones a problemas complejos.

Para aprender un lenguaje de programación, es importante tener una buena atención y memoria a corto plazo. La atención es esencial para seguir las instrucciones y entender los conceptos, mientras que la memoria a corto plazo es necesaria para recordar lo que se ha aprendido. El razonamiento y la resolución de problemas son también habilidades importantes, ya que la programación a menudo involucra la resolución de problemas y la toma de decisiones sobre cómo resolverlos.

Además, el aprendizaje de un lenguaje de programación también requiere la habilidad de abstracción, es decir, la capacidad de entender conceptos abstractos y aplicarlos a situaciones concretas. Esta habilidad es esencial para entender cómo funcionan los lenguajes de programación y cómo escribir código que haga lo que se desea.

En síntesis, aprender un lenguaje de programación requiere una serie de procesos cognitivos, incluyendo la atención, la memoria, el razonamiento y la resolución de problemas. Tener una buena comprensión de estos procesos puede ayudar a una persona a aprender un lenguaje de programación de manera más efectiva.

11. Las herramientas pedagógicas: definiciones y conceptos

Mediante una propuesta educativa se pretende desafiar el proceso de construcción personal que significa el aprendizaje en los estudiantes de ciencias económicas, teniendo en cuenta sus conocimientos previos y la interacción colaborativa con y de otros individuos, para rediseñar su campo de acción profesional enriqueciendo su estructura cognitiva, a través del fascinante mundo de la programación.

Ampliar los conocimientos informáticos en torno a un lenguaje de programación le abre a los futuros profesionales un camino lleno de posibilidades, sobreponiéndose al rol de usuario básico logrando un mayor aprovechamiento del potencial de las computadoras que dispone en su hogar u oficina para lograr en cuestión de segundos realizar tareas que tardarían decenas de horas humanas, con el sólo hecho de programar algunas pocas líneas de código o algoritmos específicos.

La elección de «un lenguaje de programación» debe ser acertada, de tal manera que permita un acceso fácil y económico para los usuarios, también que sus fuentes de consultas y difusión se encuentren lo suficientemente desarrolladas para que los estudiantes de ciencias económicas –en este caso– encuentren rápidamente motivación suficiente para entrar y permanecer atrapados por una propuesta educativa innovadora.

> Python es un lenguaje de programación. Un lenguaje de programación es un conjunto de reglas para escribir texto de manera que su computadora pueda entender lo que significa ese texto. Cuando instala Python en su computadora, instala un programa (al igual que instala Excel) que sabe cómo convertir texto (es decir, código Python) en instrucciones para que las ejecute el procesador de su computadora.(Bota y Gosa, 2021:4)

La programación es algo amplio y por eso tiende a atrapar a quienes se animan a atravesar sus fronteras. “Atrapar” en el sentido trágico de quién sufre la privación de sus libertades, pero también en el sentido más benévolo de quien lograr sentir una atracción hacia una inmensidad que va corriendo los límites cuál movediza utopía.

No está demás reiterar que se pretende alivianar de los profundos fundamentos de la programación a los estudiantes de ciencias económicas. No obstante, a los efectos de fundamentar la propuesta educativa, es oportuno dotar de contexto el surgimiento y estado de situación de Python en la comunidad especializada.

Nacido en Países Bajos en 1956, Guido van Rossum se formó en matemáticas y ciencias

de la computación en la Universidad de Ámsterdam. Desde su graduación en 1982 su aporte mas destacado al mundo de las ciencias de la computación ha sido la creación de Python, un lenguaje de programación de alto nivel que se caracteriza por su sintaxis clara y legible.

Guido van Rossum comenzó a trabajar en Python en la década de 1980 y lanzó su primera versión en 1991, desempeñado un papel fundamental en el desarrollo y evolución de Python durante muchos años.

Existen ciertos conceptos básicos que deben transmitirse a toda persona interesada en comenzar a aprender programación, confiando que una vez emprendido el camino existen grandes chances de mantener alta la motivación, buscando sus propios intereses, y fundamentalmente aprendiendo sobre los propios pasos –con aciertos y errores–.

Un concepto fundamental, sobre el que gira un programa orientado a objetos son los datos. En Python, todo es finalmente un objeto.

Los objetos se caracterizan por contener datos y funcionalidades –también llamados atributos y métodos–. Estos objetos surgen o se agrupan dentro de una determinada Clase. Las clases son estructuras –o plantillas– que configuran las características de los objetos y permite la creación o instanciación de nuevos objetos.

La «clase» CLIENTES, permite la creación de objetos dónde cada uno de ellos tendrá sus propios “atributos” y “métodos”.

También se podrían mencionar otras características muy técnicas que tiene Python con el objetivo de acercarnos a la forma de pensar que debería adquirir un programador profesional: Abstracción, Herencia, Encapsulación y Polimorfismo.

Sin embargo, este nivel de profundidad de análisis parece impropio para los destinatarios finales de esta propuesta educativa y las mismas pueden adquirirse con la práctica.

Entonces, el esfuerzo debe estar dado por transmitir desde un comienzo aquellos conceptos y herramientas que permitan un «despertar» rápido del estudiante.

Por ejemplo, conocer de la existencias de una innumerable cantidad de «librerías» en Python puede constituirse en un verdadero descubrimiento.

Una librería –para algunos también conocidas como bibliotecas– son funciones predefinidas que se pueden utilizar en un programa de Python, que ya fueron desarrolladas por otros programadores y pueden ser reutilizadas por otros. Estas bibliotecas proporcionan funcionalidades específicas que facilitan el desarrollo de software al ofrecer implementaciones reutilizables de código.

Por citar sólo algunas que se consideran de mucha utilidad para el profesional en ciencias económicas se tienen:

- Pandas: Para análisis y manipulación de datos.
- NumPy: Para operaciones numéricas y manipulación de matrices.
- Matplotlib: Para visualización de datos.

Además de la fortaleza de las librerías, Python presenta una simpleza en su sintaxis capaz de permitir la creación de los objetos e identificar el tipo al que se refiere desde su creación:

Con solo tipear:

cantidad = 100

y

mensaje = “Su saldo es: ”

es suficiente para que Python identifique la existencia de una variable denominada «cantidad» y su tipo es un “número entero” y la existencia de una variable denominada «mensaje» cuyo contenido es de tipo “cadena de texto” y su valor: “es Su saldo es: ”.

Se puede dar la instrucción que se muestren estas dos variables en una misma línea, mediante:

print (mensaje + cantidad)

y el resultado visible en la pantalla será:

Su saldo es: 100

Esto conlleva a la idea de palabras reservadas en los lenguajes de programación.

Por ejemplo «print» tiene un solo significado para el lenguaje y responde a la función de imprimir por pantalla algo.

Sin embargo si uno quisiera que se imprima la palabra print, no lo podrá hacer haciendo «print print». Habrá que seguir una restricción gramatical que consta de primero escribir la función print y a continuación, lo contenido entre comillas simples y encerrado entre paréntesis.

La solución es: print (“print”) siendo el objeto impreso una cadena de texto.

También existen otros objetos fácilmente relacionables con elementos de la vida real como las “listas”, los “diccionarios”, las “tuplas” y los indispensables “dataframes” que en la jerga del profesional de ciencias económicas no es mas que un cuadro de celdas con datos de doble entrada, asimilables con un hoja de cálculo de Microsoft Excel®.

Dejando por un momento estas menciones introductorias relacionadas a estas nuevas entidades dentro de Python, con denominaciones bastante familiares, pretendiendo sean un agradable despertar, donde el dominio de idioma inglés adquiere protagonismo, se retoman en profundidad en la resolución de los ejercicios prácticos seleccionados.

Retomando las investigaciones de los años sesenta de Alan Kay puede afirmarse que Python es una derivación de nuevos paradigmas de interrelación entre los datos y los algoritmos, presentad en sociedad como código abierto y gratuito lo que permitió su rápida expansión entre los programadores.

Su lógica está basada en el paradigma orientado a objetos (POO), que tiene como virtud una estructura similar a cómo piensa el ser humano y cuenta con una sintaxis accesible para las personas con un nivel de “alfabetización” básico en lenguajes de programación.

Un lenguaje de programación orientado a objetos es un paradigma de programación que se basa en la representación de conceptos del mundo real como “objeto” en el código. Estos objetos tienen propiedades (también llamadas atributos) y comportamientos (también llamados métodos) y se pueden interactuar entre sí enviándose mensajes y solicitando servicios.

En un lenguaje de programación orientado a objetos, se crean clases para definir los objetos y sus propiedades y comportamientos. Luego, se pueden crear instancias de estas clases, que son objetos individuales con sus propias características y comportamientos.

Uno de los beneficios de utilizar un lenguaje de programación orientado a objetos es que permite una mayor modularidad y reutilización de código. Las clases pueden ser reutilizadas para crear diferentes objetos y pueden ser modificadas o extendidas para adaptarse a diferentes necesidades sin tener que reescribir todo el código. Además, el enfoque orientado a objetos también puede hacer que el código sea más fácil de entender y mantener, ya que divide el problema en partes más pequeñas y manejables.

Es necesario pensar en objetos que se interrelacionan, comunicándose entre ellos. Cada objeto posee datos (también llamados atributos) y funcionalidades (métodos). Para facilitar la creación en forma reiterada de nuevos objetos se utilizan las plantillas y moldes que tendrán las estructuras mínimas requeridas para cada tipo de objeto que se denominan CLASES.

El proceso a partir del cual se utiliza una clase para crear un objeto, se llama INSTANCIAR Esto permite escribir el código por única vez para crear la clase con sus datos y funcionalidades y luego cada vez que se cree un objeto de esa clase no haga falta tener que repetir código. Por ejemplo la clase CLIENTE tiene como datos Razon Social y Cuit y como funcionalidades Liquidar IVA, Liquidar Ganancias. A partir de estas estructura luego es fácil reproducir clientes, cada uno de los cuales será un OBJETO. La acción mediante la utilización de una clase para la creación de un objeto se llama INSTANCIAR.

12. Los entornos de desarrollo: Jupyter Notebook y Google Colab

La enseñanza de programación en el ámbito universitario se ha vuelto fundamental para preparar a los estudiantes en habilidades tecnológicas relevantes. En este contexto, Python ha surgido como un lenguaje de programación popular para introducir a los estudiantes a los conceptos fundamentales de la programación. Sin embargo, la forma en que se imparte esta enseñanza es un aspecto crítico que puede influir en el proceso de aprendizaje y comprensión de los estudiantes.

La experiencia práctica durante las pruebas piloto han confirmado las ventajas pedagógicas de utilizar Jupyter Notebooks como herramienta en la enseñanza de Python, destacando su capacidad para fomentar la interactividad, experimentación, documentación, visualización de datos, reutilización y colaboración, así como su flexibilidad y adaptabilidad a diferentes estilos de enseñanza.

Si bien durante todo el proceso de investigación se utilizo este entorno, el proyecto Google Colab también tiene similares características y permite el trabajo en la nube y de manera colaborativa con otros usuarios/programadores por lo que se la considera otra herramienta de gran potencial para la enseñanza.

Sus fortalezas son:

- Interactividad y experimentación: Jupyter Notebooks permite a los estudiantes interactuar directamente con el código de Python, lo que fomenta la experimentación y la exploración activa. Los estudiantes pueden ejecutar fragmentos de código de manera incremental y observar los resultados de inmediato. Esta capacidad de experimentar directamente con el código en un entorno interactivo ayuda a reforzar la comprensión de los conceptos y facilita el proceso de aprendizaje. Además, la retroalimentación inmediata proporcionada por Jupyter Notebooks permite a los estudiantes corregir errores de forma rápida y eficiente.

- Documentación y narrativa: Una de las características distintivas de Jupyter Notebooks es su capacidad para combinar código ejecutable con texto enriquecido, como explicaciones, ejemplos y gráficos. Esta funcionalidad permite a los educadores crear una narrativa coherente alrededor de los conceptos de programación, proporcionando explicaciones claras y concisas junto con ejemplos de código concretos. Los estudiantes pueden acceder a una documentación contextual y detallada que los guía a lo largo del proceso de aprendizaje, lo que facilita la comprensión y la asimilación de los conceptos. La capacidad de agregar explicaciones en formato de texto también brinda a los estudiantes diferentes formas de aprender, adaptándose a sus preferencias de aprendizaje.

- Visualización de datos: Python es ampliamente utilizado en la visualización de datos, y Jupyter Notebooks facilita la creación y visualización de gráficos y figuras. Los estudiantes pueden generar visualizaciones interactivas directamente en el mismo entorno donde escriben su código, lo que les permite explorar y comprender mejor los datos. Esta capacidad de visualización en tiempo real brinda a los estudiantes una experiencia de aprendizaje más inmersiva y estimulante. La visualización de datos también ayuda a los estudiantes a comprender conceptos abstractos y a desarrollar habilidades de análisis crítico.

- Reutilización y colaboración: Jupyter Notebooks promueve la reutilización y la colaboración, lo que es especialmente valioso en un entorno educativo. Los estudiantes pueden compartir sus notebooks con sus compañeros y profesores, lo que facilita la colaboración y el intercambio de conocimientos. Además, los notebooks pueden servir como recursos de aprendizaje a largo plazo, permitiendo a los estudiantes revisar y practicar conceptos en cualquier momento. La capacidad de compartir y colaborar en tiempo real también fomenta la participación activa de los estudiantes y promueve un ambiente de aprendizaje colaborativo.

- Flexibilidad y adaptabilidad: Jupyter Notebooks es una herramienta flexible que se adapta a diferentes estilos de enseñanza y aprendizaje. Los educadores pueden estructurar los notebooks de acuerdo con sus propias necesidades y preferencias, organizando el contenido de manera coherente y secuencial. Además, los estudiantes pueden avanzar a su propio ritmo y revisar conceptos previos en cualquier momento, lo que fomenta un aprendizaje autónomo y personalizado. La flexibilidad de Jupyter Notebooks también permite a los educadores adaptar los materiales de enseñanza a diferentes niveles de dificultad, adaptándose a las necesidades específicas de los estudiantes.

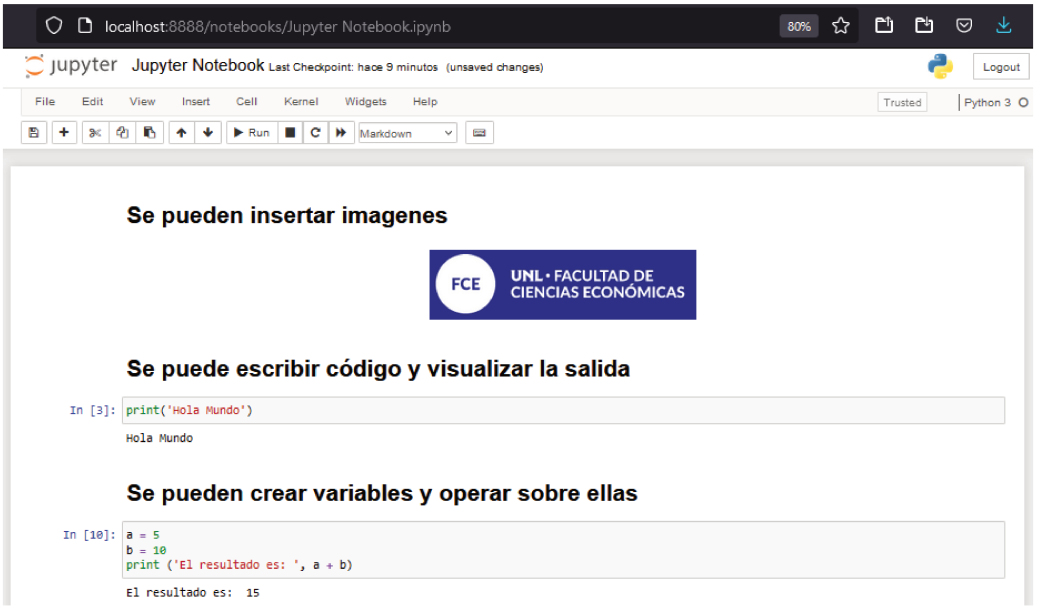


Figura 1: Jupyter

La utilización de Jupyter Notebooks en la enseñanza de Python ofrece ventajas pedagógicas significativas. Su capacidad para fomentar la interactividad, experimentación, documentación, visualización de datos, reutilización y colaboración, así como su flexibilidad y adaptabilidad, proporcionan a los educadores y estudiantes un entorno de aprendizaje enriquecedor. La combinación de código ejecutable, explicaciones claras y ejemplos concretos permite a los estudiantes desarrollar una comprensión profunda de los conceptos de programación. Al implementar Jupyter Notebooks como herramienta educativa, los educadores pueden mejorar la experiencia de aprendizaje de los estudiantes, fomentando su participación activa y promoviendo un aprendizaje significativo y autónomo en el contexto de la programación con Python.

13. La opción por Python: Características y potencialidad

Un análisis inicial lleva a pensar que los principales beneficios de utilizar Python en las ciencias económicas es la gran cantidad de bibliotecas y herramientas disponibles para el análisis de datos y la modelización económica. Estas bibliotecas son las que proporcionan una gran cantidad de funcionalidades para el manejo y procesamiento de datos, incluyendo la limpieza de datos, el análisis estadístico y la visualización de datos.

Además, Python también tiene una amplia comunidad de usuarios y desarrolladores que han creado una gran cantidad de herramientas y paquetes adicionales –creadores de bibliotecas–. Esto hace que sea relativamente fácil encontrar soluciones a problemas específicos y obtener ayuda en línea. Otro beneficio de utilizar Python en las ciencias económicas es que es un lenguaje de programación de alto nivel que es fácil de aprender y utilizar. Esto lo hace ideal para aquellos que no tienen experiencia previa en programación o para aquellos que quieren utilizar Python como una herramienta adicional en su arsenal de análisis de datos.

«Una de las razones por las que me gusta Python es por que proporciona un equilibrio muy bueno entre lo práctico y lo conceptual. Puesto que Python es un lenguaje interpretado, los principiantes pueden tomar el lenguaje y empezar a hacer cosas interesantes casi inmediato, sin perderse el los problemas de compilación y enlazado. Además, Python viene con una gran biblioteca de módulos que se pueden usar para hacer toda clase de tareas que abarcan desde programación para web a gráficos. Este enfoque práctico es una buena manera de enganchar a estudiantes y permite que completen proyectos significativos. Sin embargo, Python también puede servir como una base excelente para introducir conceptos importantes de informática.». (Downey et al.: p vi)

Es difícil predecir con certeza cuáles serán los lenguajes de programación con más potencial futuro, ya que esto depende en gran medida de las tendencias y necesidades del mercado y de la industria en general. Sin embargo, hay algunos lenguajes de programación que han demostrado ser populares y versátiles y que, por lo tanto, podrían tener un potencial futuro fuerte.

- 1. Python es un lenguaje de programación de alto nivel y fácil de usar que se utiliza en una amplia variedad de campos, incluyendo el análisis de datos, el **machine learning**, la ciencia de la computación y el desarrollo web. Tiene una gran cantidad de bibliotecas y herramientas disponibles y una amplia comunidad de usuarios y desarrolladores, lo que lo hace muy atractivo para muchos profesionales.

- Java es un lenguaje de programación orientado a objetos que se utiliza ampliamente en la industria y tiene una gran cantidad de herramientas y bibliotecas disponibles. Es especialmente popular en el desarrollo de aplicaciones empresariales y móviles.

- C++ es un lenguaje de programación de bajo nivel y de alto rendimiento que se utiliza en una amplia variedad de campos, incluyendo el desarrollo de sistemas operativos, juegos y aplicaciones empresariales. Aunque es un lenguaje más difícil de aprender que algunos otros, su rendimiento y versatilidad lo hacen atractivo para muchos profesionales.

- JavaScript es un lenguaje de programación de alto nivel que se utiliza principalmente en el desarrollo web y en la creación de aplicaciones web y móviles. Es un lenguaje muy popular y versátil, y se espera que siga siendo una elección popular para el desarrollo web en el futuro.

En una nota periodística de la sección Tecnología del Diario Clarín se analiza el perfil laboral demandado por algunos de los denominados «unicornios» en la Argentina, entre los que se citan a Mercado Libre, Despegar, Globant, Olx, Auth0, Ualá, Vercel, Aleph, Tiendanube, Mural y Bitfarms. Si bien se advierte que es difícil decir cuál es «el» lenguaje de programación más usado, sobre todo porque varía según los usos, cuando describe la potencial de cada uno los autores refieren que Python es «es el lenguaje de programación más popular del mundo y esto no es casual: ha sido reconocido en todo el mundo como el más accesible, fácil y simple de aprender».

> «Tiene una sintáxis muy intuitiva que ayuda a los que están empezando pero, además, es trasladable a otros lenguajes, lo cual es una gran herramienta. También corre con la ventaja de que todo lo que se aprende sirve para desarrollos más complejos, pero con un plus: al ser el más usado del mundo, no solo tiene presente sino también futuro en cuanto a usabilidad. Una gran parte de las empresas argentinas de tecnología usan Python».(«Los lenguajes de programación más buscados por los unicornios argentinos», Diario Clarín: pag....)

Otro plano de análisis, que no se puede soslayar es si resulta necesario ampliar los conocimientos por encima de las utilidades que hoy brinda el software de mayor difusión entre los profesionales de ciencias económicas que si dudase Microsoft Excel®.

El terreno que muy bien se ha ganado Microsoft Excel®en la práctica profesional se debe a lo útil, eficiente y fácil de usar en la manipulación de datos en contextos de mediana complejidad de cálculos, especialmente amigable para quienes no tienen experiencia en programación.

Python y Microsoft Excel®pueden ser consideradas como dos herramientas muy diferentes que se utilizan en diferentes contextos y para diferentes propósitos. Sin embargo, recientemente han aparecido funcionalidades combinadas producto de alianzas estratégicas que

permiten mantenerse ambas vigentes y renovadas en la consideración de los usuarios.

Por tanto, no es de extrañar que sean cada vez mas frecuente la necesidad de aplicar nociones básicas en Python, incluso para aplicarlas dentro de las planillas de cálculo de Microsoft Excel®.

Mas allá de complementariedad o competencia, existen puntos comparativos donde la utilización de un lenguaje con fines específicos generará un mejor rendimiento que las funcionalidades previstas por un software sea libre o licenciado.

- Python es un lenguaje de programación de alto rendimiento que puede manejar grandes conjuntos de datos con facilidad, mientras que Microsoft Excel®tiene una capacidad limitada en la cantidad de los registros a procesar y a la velocidad de respuesta.

- Python es un lenguaje de programación muy versátil que se puede utilizar para realizar una amplia variedad de tareas de manipulación y análisis de datos. Puede utilizar bibliotecas y herramientas especializadas para realizar tareas específicas y personalizadas. En cambio Microsoft Excel®ofrece menos flexibilidad y no es tan fácil de personalizar.

- Con Python, es posible escribir scripts y programas para automatizar tareas repetitivas de manipulación y análisis de datos. Esto puede ahorrar tiempo y esfuerzo y garantizar la precisión de los resultados. Por otro lado, si bien esto es posible con las macros de Microsoft Excel®no es tan fácil de automatizar y requiere más intervención manual.

- Python se puede integrar con otras herramientas y sistemas de manera relativamente fácil. Esto significa que puede utilizar Python para manipular y analizar datos provenientes de otras fuentes y utilizar los resultados en otras aplicaciones. Por otro lado Microsoft Excel®presenta limitaciones en la integración con otras herramientas y sistemas.

Adquirir habilidades y competencias en programación le brindan al profesional en ciencias económicas la posibilidad de generar, por sí, soluciones muy valoradas en procesos rutinarios y con cargas de datos repetitivas.

Python es un lenguaje de programación muy versátil y puede ser utilizado para automatizar una amplia variedad de tareas desde compilación entre documentos hasta envío de correos electrónicos.

Al escribir scripts o programas en Python, es posible automatizar tareas que de otra manera serían realizadas manualmente, lo que puede ahorrar tiempo, disminuir el uso de recursos y esfuerzos y garantizar la precisión de los resultados.

Con muy pocas líneas de código es posible:

- Leer y procesar grandes conjuntos de datos, eliminar errores y duplicados, descartar datos irrelevantes y preparar los datos para su análisis.
- Realizar análisis estadísticos y visualizar los resultados de manera automática.
- Recopilar y procesar datos de diferentes fuentes y generar informes automatizados.
- Enviar correos electrónicos de manera automática a partir de datos estructurados en otras bases de datos.

Parte III

Desarrollo de las etapas y hallazgos obtenidos

Para el logro de los Objetivos propuestos se realizan actividades secuenciadas y en algunos casos simultáneas. Las mismas pueden agruparse en tres momentos o etapas, a saber:

En la etapa indagatoria - exploratoria se tuvo como objetivo comenzar a indagar acerca del estado de conocimiento de Python dentro de la comunidad vinculada a la comunidad profesional en la ciudad de Santa Fe.

Se mantuvieron entrevistas con profesionales y se realizaron encuestas entre los estudiantes de las carreras afines den la FCE-UNL.

Asimismo dentro de esta etapa, se llevaron adelante entrevistas personales con otros profesionales en informática.

En la segunda etapa se produjo la identificación de los casos de estudio, circunscribiéndose en su mayoría a partir de información brindada por una empresa hotelera de la ciudad de Santa Fe. De allí se relevaron fuentes de información, sistemas contable, y necesidades propias para poder presentarse como ejercicios prácticos basados en casos reales.

Finalmente, se llevaron a cabo en formato de taller las actividades previstas en la tercer etapa, surgiendo de allí los detalles finales para delinear una propuesta didáctica. En esta etapa se consolidó la idea validándose los supuestos iniciales siendo una instancia muy valiosa de retroalimentación al cabo de seis encuentros al cabo de dos meses de trabajo.

14. Diagnóstico del nivel y alcance del conocimiento sobre Python entre profesionales

Se realizaron un total de siete encuestas personalizadas, seleccionando profesionales de la matrícula del Colegio de Profesionales de Ciencias Económicas con asiento en la ciudad de Santa Fe, con diferentes años de antigüedad en el ejercicio profesional.

1- Edad:

A partir de ciertas relaciones profesionales previas y por contactos personales con algunos de ellos se pretendió alcanzar un nivel de respuesta en tres rangos etarios diferentes.

Se recibieron respuestas positivas a realizar la entrevista sobre el tema por parte de 3 profesionales de entre 30 y 40 años de edad, 2 de entre 40 y 50 años y un solo profesional mayor a los 50 años.

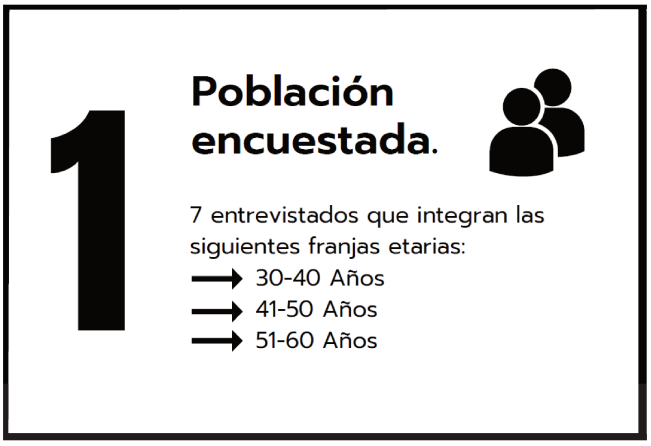
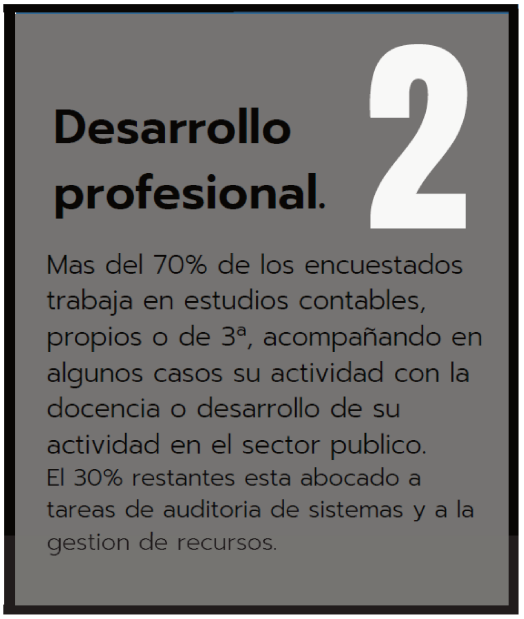


Figura 2: Respuesta a pregunta 1

2- Desarrollo profesional:

De los profesionales entrevistados cinco ejercen la profesión en estudios contables atendiendo distintos requerimientos de tipo impositivo, laboral, societario y auditorías.

Dos de los profesionales entrevistados también desarrollan tareas laborales en la docencia universitaria y otros dos complementan sus actividades en relación de dependencia en el sector público provincial.



III. Teniendo en cuenta la potencialidad de Python para la automatización de tareas del quehacer profesional se indagó acerca de la ponderación que se tienen sobre las tareas rutinarias y repetitivas en la práctica profesional.

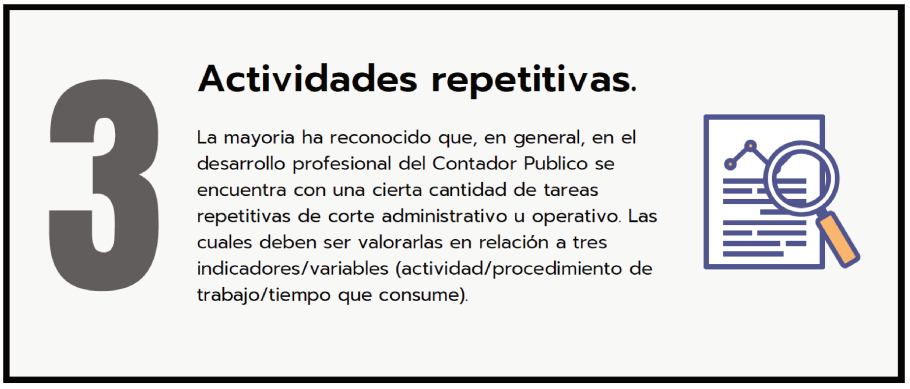


Figura 3: Respuesta a pregunta 3

IV. Para profundizar en aquellas tareas reconocidas como repetitivas se solicitó a los entrevistados que mencionen algunos ejemplos, donde se destacan aquellas vinculadas a la manipulación de información mediante archivos digitales mediante diferentes sistemas de información o programas.

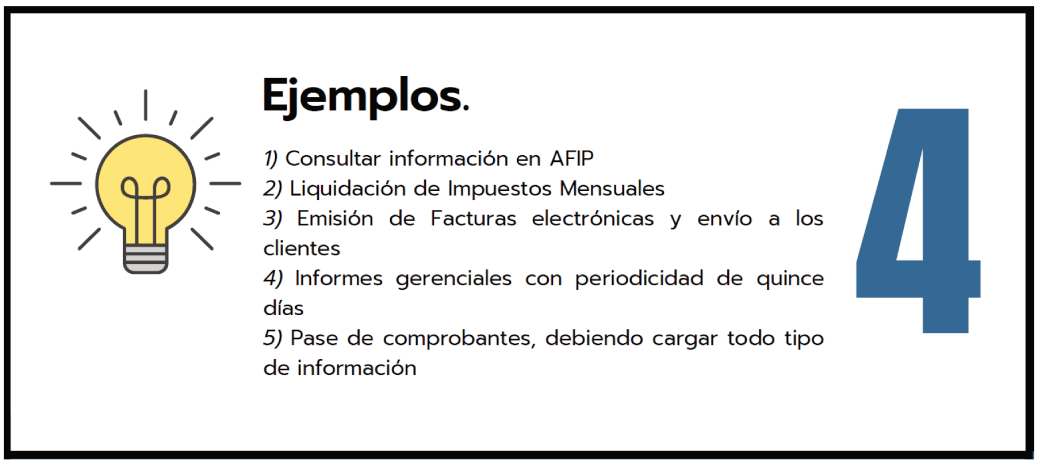


Figura 4: Respuesta a pregunta 4

V. Para validar cierta bibliografía analizada en la etapa indagatoria se consultó acerca del grado de utilización del programa Microsoft Excel en la práctica profesional, corroborándose su difundida utilización en todos los entrevistados.

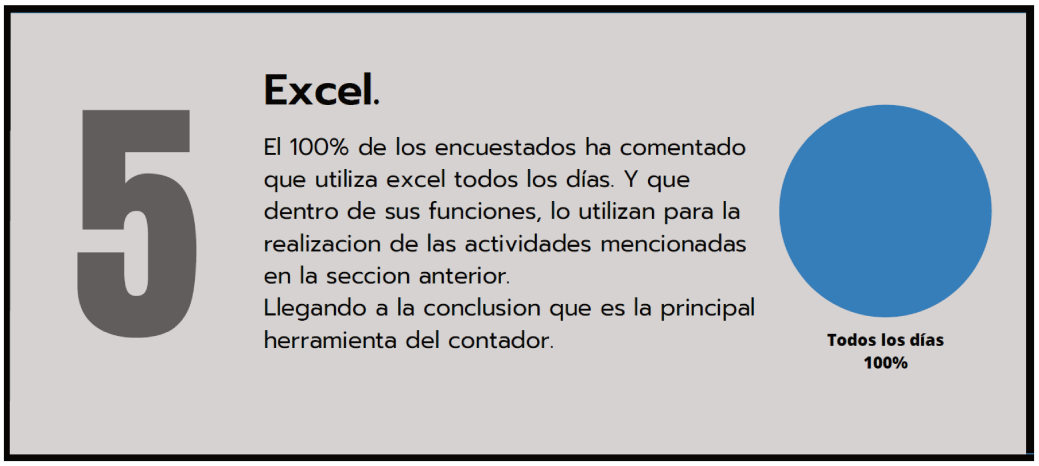


Figura 5: Respuesta a pregunta 5

Con respecto al nivel de conocimientos previos en algún lenguaje de programación, los entrevistados que respondieron afirmativamente corresponden a la franja etaria entre los 30 y 40 años.

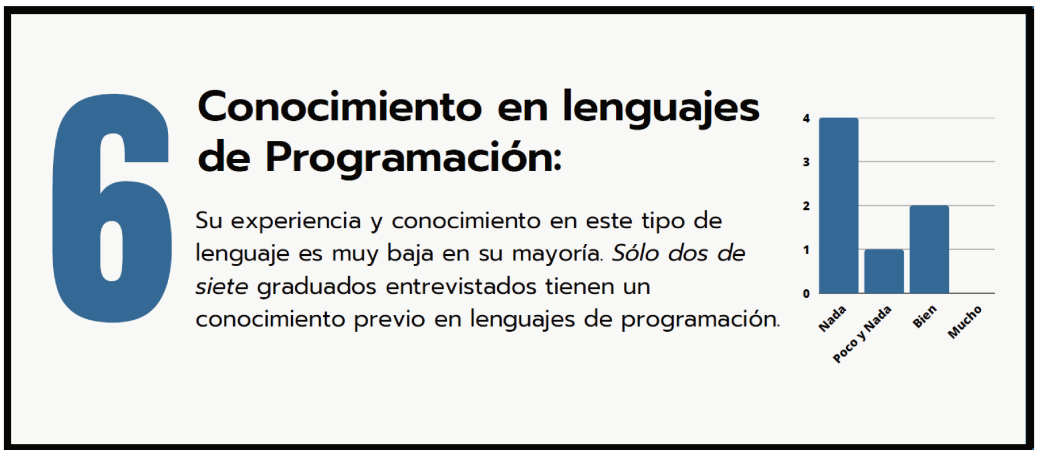


Figura 6: Respuesta a pregunta 6

Sobre la respuesta anterior se solicitó se nombrasen aquellos lenguajes que conocieran o supieran de su utilización en las Ciencias Económicas.

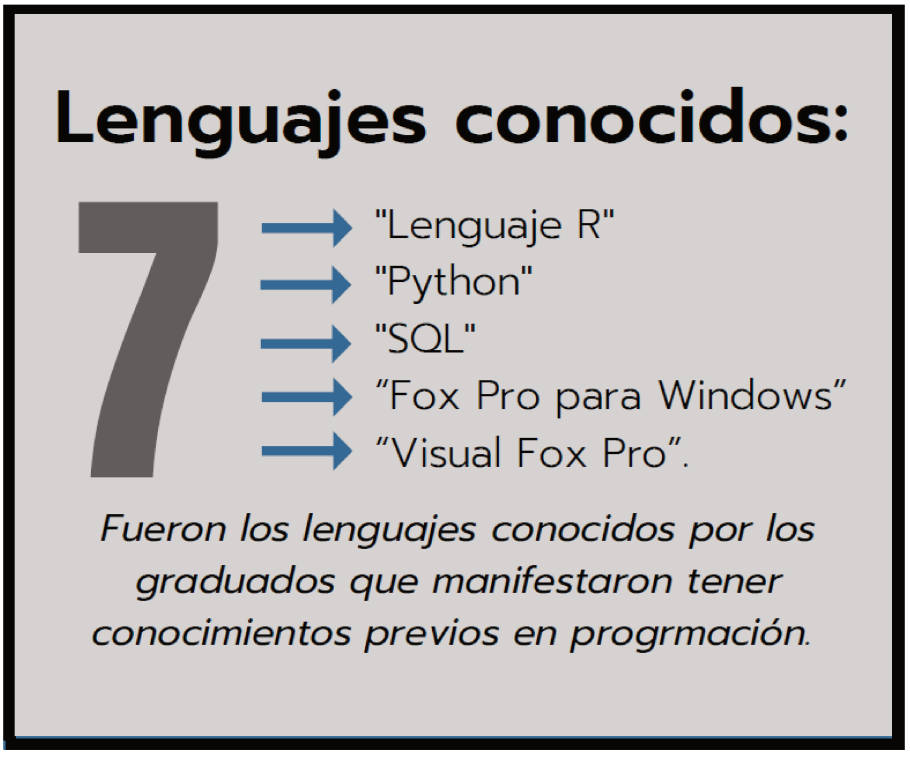


Figura 7: Respuesta a pregunta 7

En general la consideración acerca de una posible tendencia sobre la demanda del

mercado laboral, los entrevistados coincidieron en que la misma estará determinada por lo que suceda en el mercado laboral internacional.

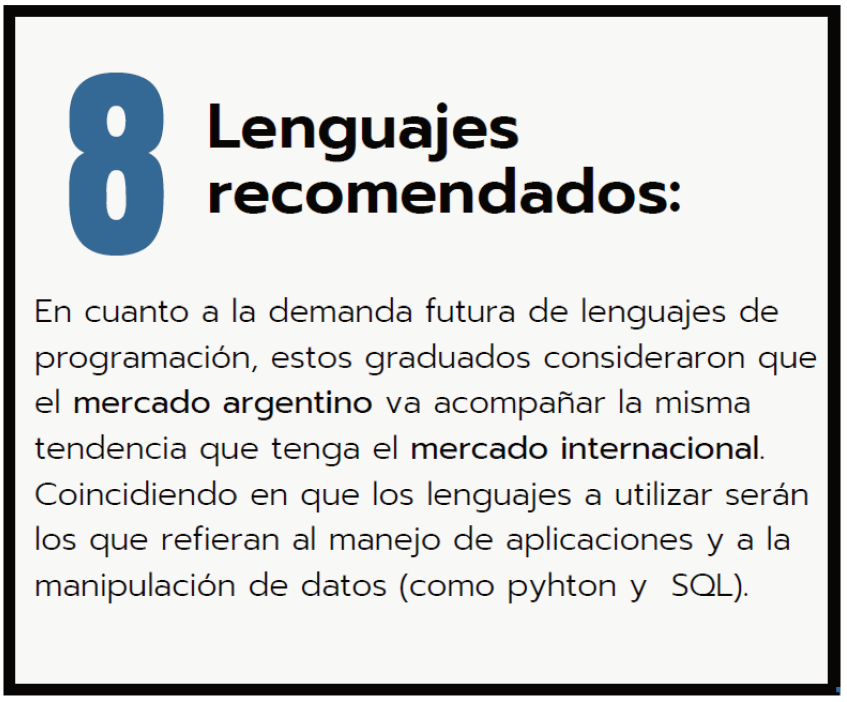


Figura 8: Respuesta a pregunta 8

¿Conoce el lenguaje Python? ¿Podría definirlo? Utilice si es posible adjetivos para calificarlo.

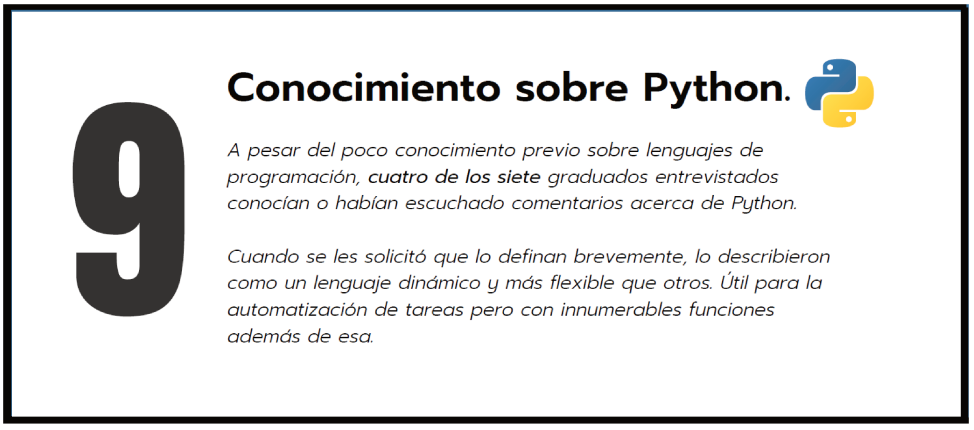


Figura 9: Respuesta a pregunta 9

X. ¿considera a Python como un lenguaje apropiado para dar soporte al desarrollo

profesional, de actividades vinculadas a las ciencias económicas?

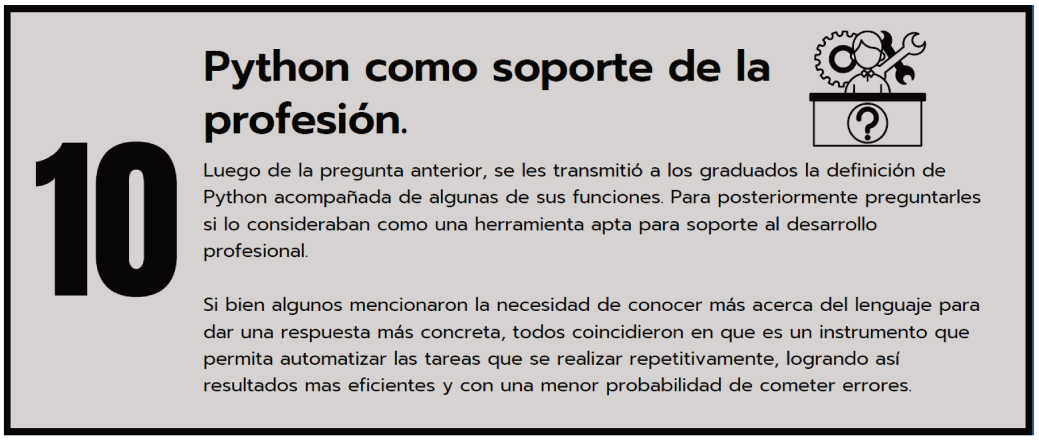


Figura 10: Respuesta a pregunta 10

XI. ¿Considera que la utilización de la programación para potenciar actividades de la profesión es un tema que actualmente se encuentra en discusión?

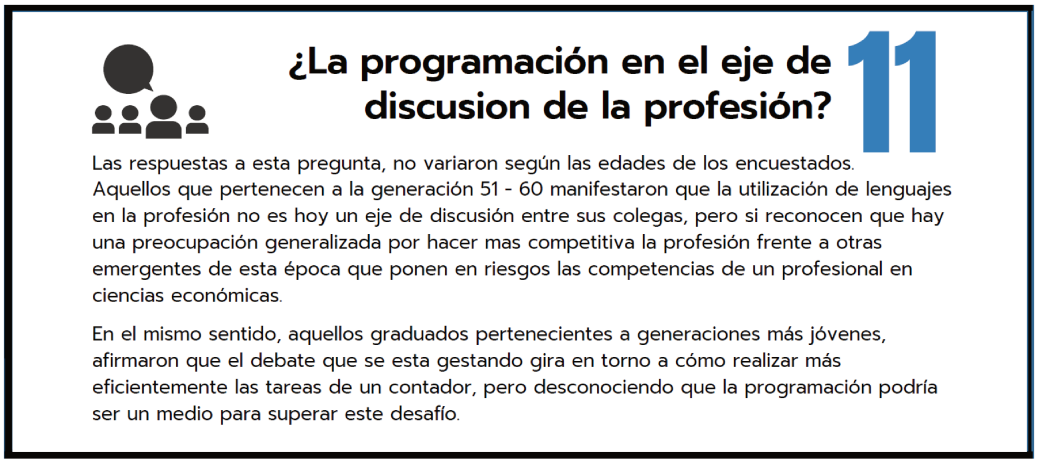


Figura 11: Respuesta a pregunta 11

XII. ¿Consideras necesario que los futuros profesionales nos capacitemos durante nuestra formación universitaria en la temática?

XIII. Consideraciones finales - Resumen.

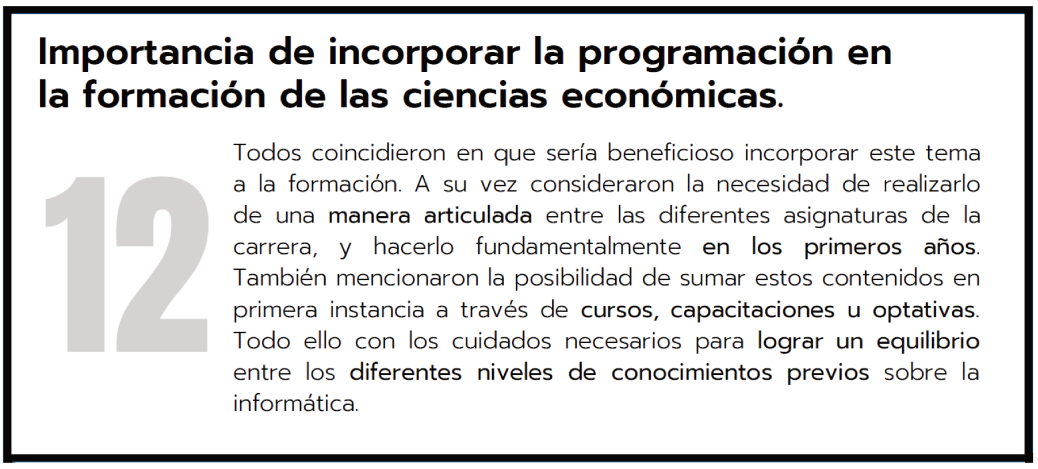


Figura 12: Respuesta a pregunta 12

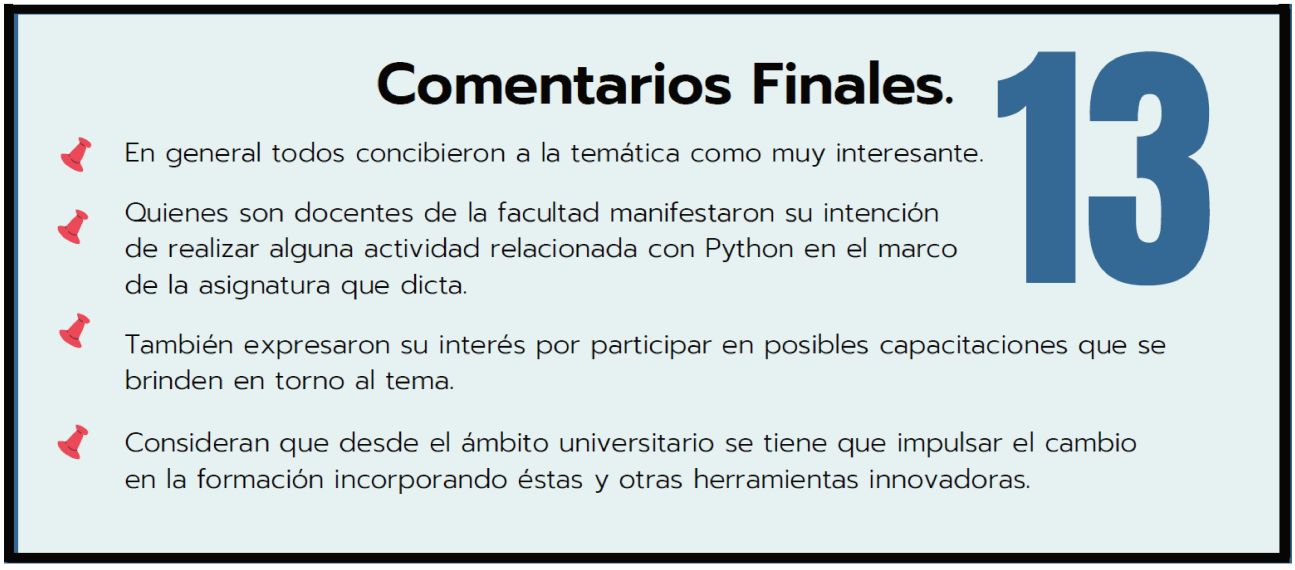


Figura 13: Respuesta a pregunta 13

15. Diagnóstico sobre el grado de interés sobe Python entre estudiantes

Para realizar una encuesta representativa entre los actuales estudiantes de la FCE-UNL se puso a disposición por 25 días un Formulario de Google, y dándose difusión a través de las redes sociales del Centro de Estudiantes, lográndose un total de 205 respuestas.

La participación por género logró una representación muy similar a la matrícula existente.

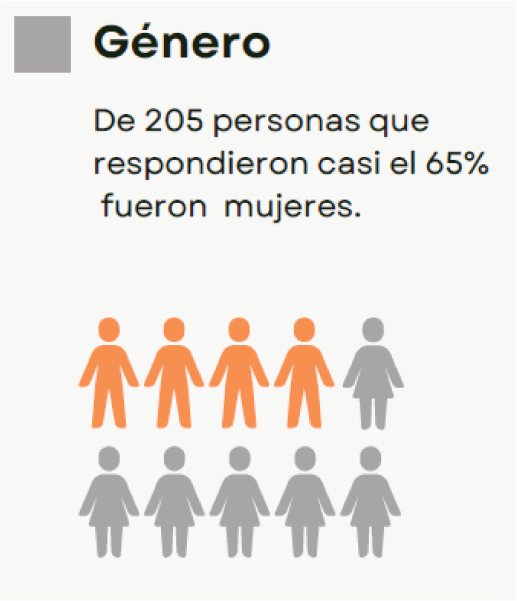


Figura 14: Respuesta a pregunta 1

La participación por edades también mantuvo una estructura similar a la de los estudiantes en general.

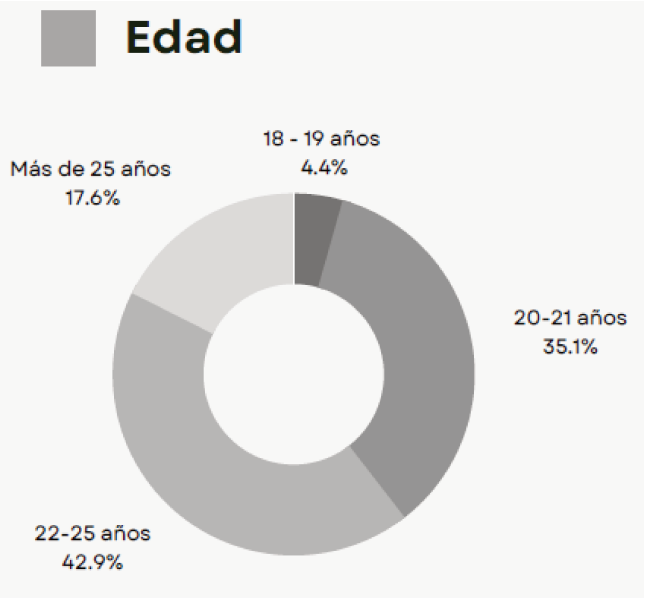


Figura 15: Respuesta a pregunta 2

Se presentó un menor interés en la participación de los ingresantes y se mantuvo mas estable en los años posteriores, siendo la mayor participación relativa la de quienes se encuentran cursando el quinto o mas años desde sus respectivos ingresos.

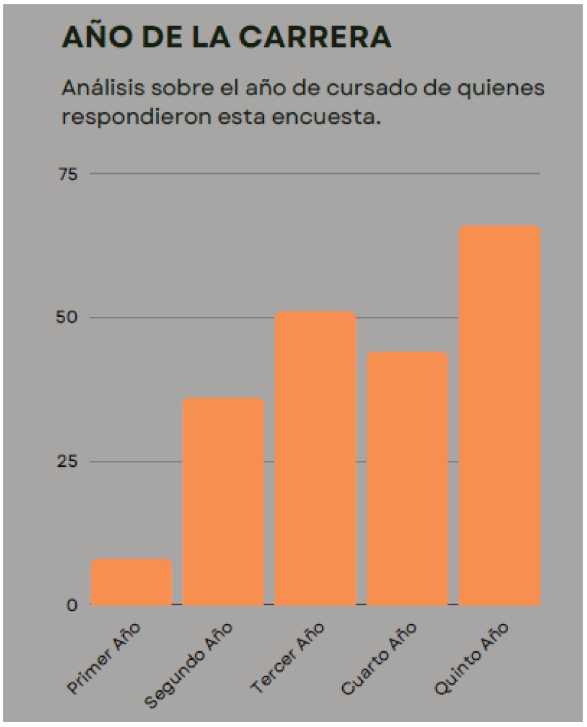


Figura 16: Respuesta a pregunta 3

Se consultó acerca del grado de conocimiento previo de Python, dándoles a quienes respondieron afirmativamente una opción posterior para su identificación entre las siguientes opciones:

- Un Software de gestión contable.
- Un lenguaje de programación de usos múltiples.
- Una herramienta contable, de uso obligatorio impuesto por la AFIP.
- Una teoría contable, impulsada por Andrew Python.

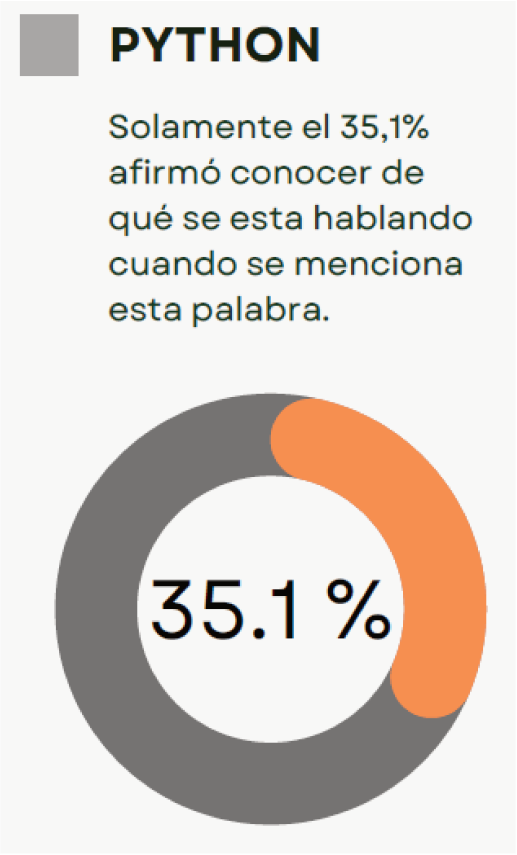


Figura 17: Respuesta a pregunta 4

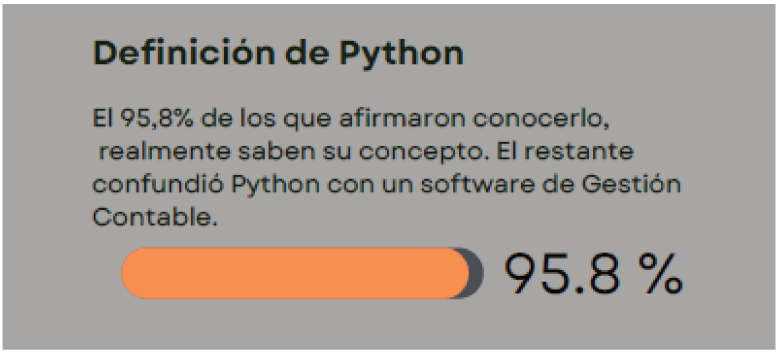


Figura 18: Respuesta a pregunta 5

Para permitir continuar con la encuestas, en particular a quienes no presentaron un conocimiento previo sobre Python se les facilitó la siguiente cita "Python es uno de los lenguajes de programación más conocidos del mundo. Ha creado de todo, desde el algoritmo de recomendaciones de Netflix hasta el software que controla los vehículos autónomos. Se trata de

un lenguaje de uso general, lo que significa que está diseñado para usarse en una gran variedad de aplicaciones. Cuenta con una sintaxis accesible para las personas con un nivel de 'alfabetización' básico en lenguajes de programación lo que conlleva a que Python sea uno de los lenguajes más requeridos en el mundo por encima de las alternativas que existen en el mercado actualmente.

Menos del 20 % de los encuestados manifestó tener conocimientos suficientes de informática.

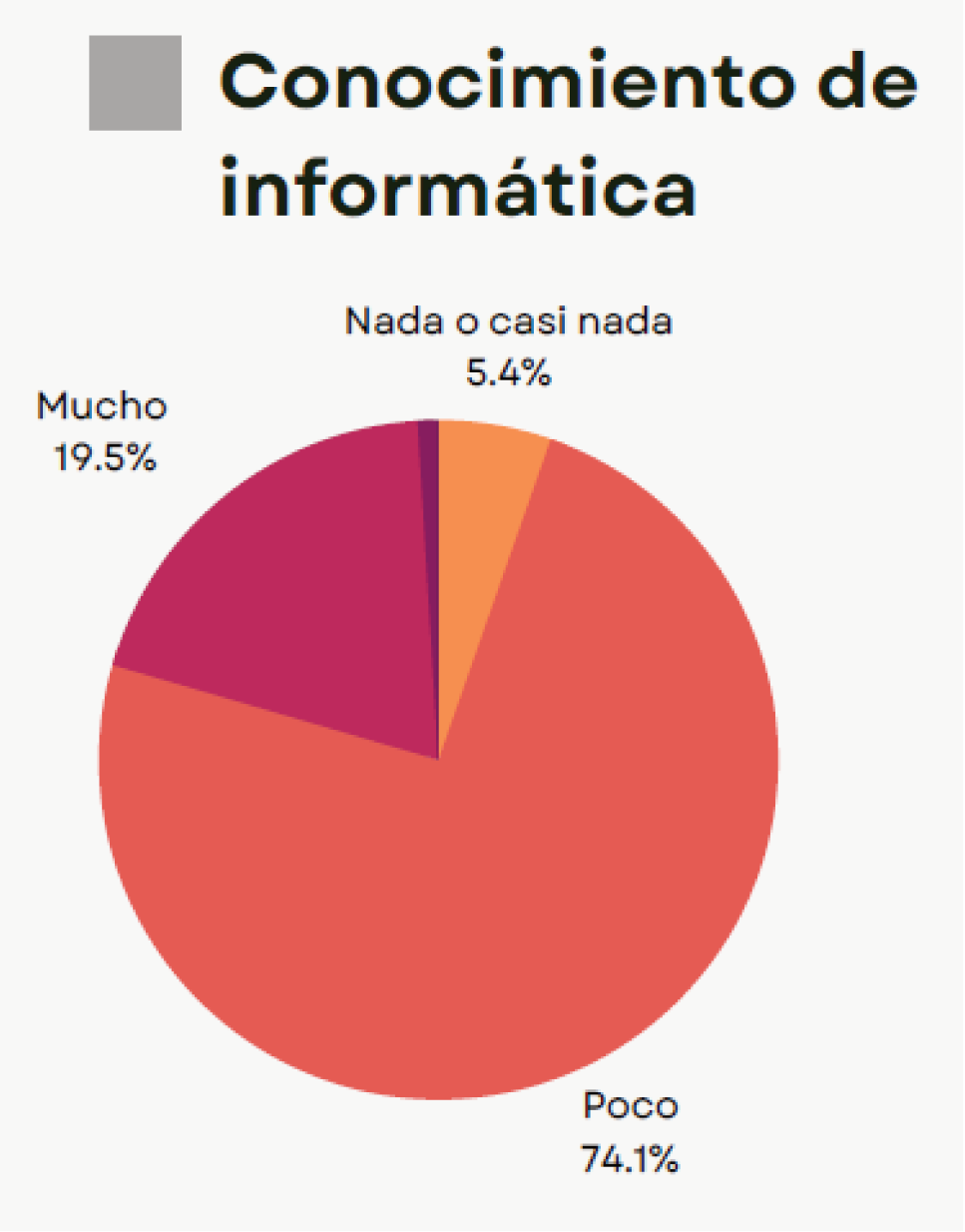


Figura 19: Respuesta a pregunta 6

Y menos del 25 % por ciento de los encuestados manifestó tener algo de conocimiento de programación.

Figura 20: Respuesta a pregunta 7

A partir de la definición brindada se solicitó si pensaban que podría existir alguna potencial relación entre las Ciencias Económicas y el lenguaje de programación Python.

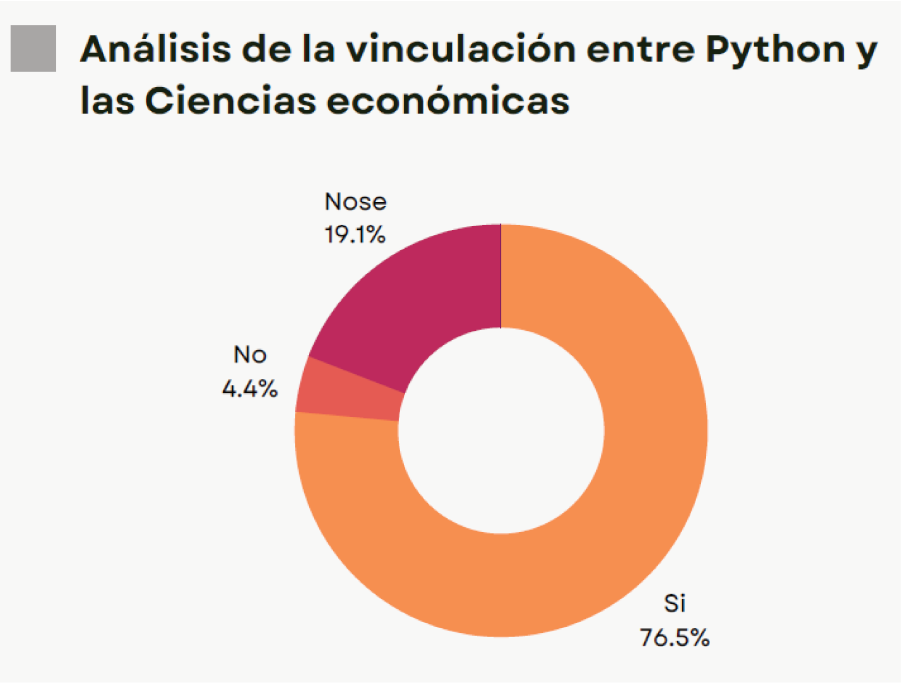


Figura 21: Respuesta a pregunta 8

9- A partir de la definición que te brindamos, ¿podrías relacionar a Python con las ciencias económicas? (marcar opción correcta)

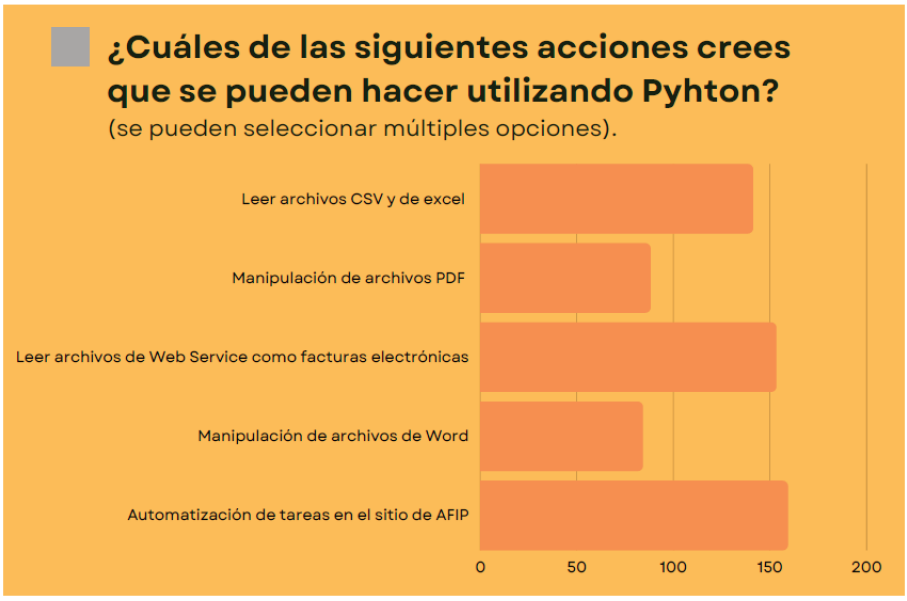


Figura 22: Respuesta a pregunta 9

10- ¿Cuál los siguientes acciones crees que se pueden hacer utilizando PYTHON? (marque todas las que crea correcta)

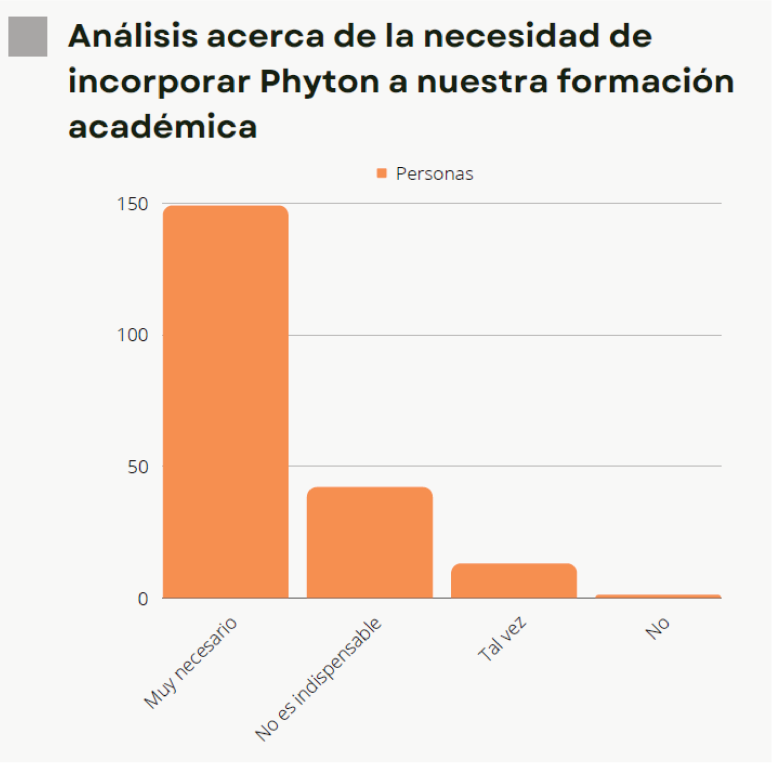


Figura 23: Respuesta a pregunta 10

11- ¿Sabías que todas las opciones anteriores son correctas? A partir de esto: ¿te parece necesario que incorporemos conocimientos de PYTHON en nuestra formación académica con el objetivo de hacer más eficientes las tareas realizadas por un contador?

12- ¿Estarías interesado/a en capacitarte acerca de PYTHON? (marcar opción correcta)

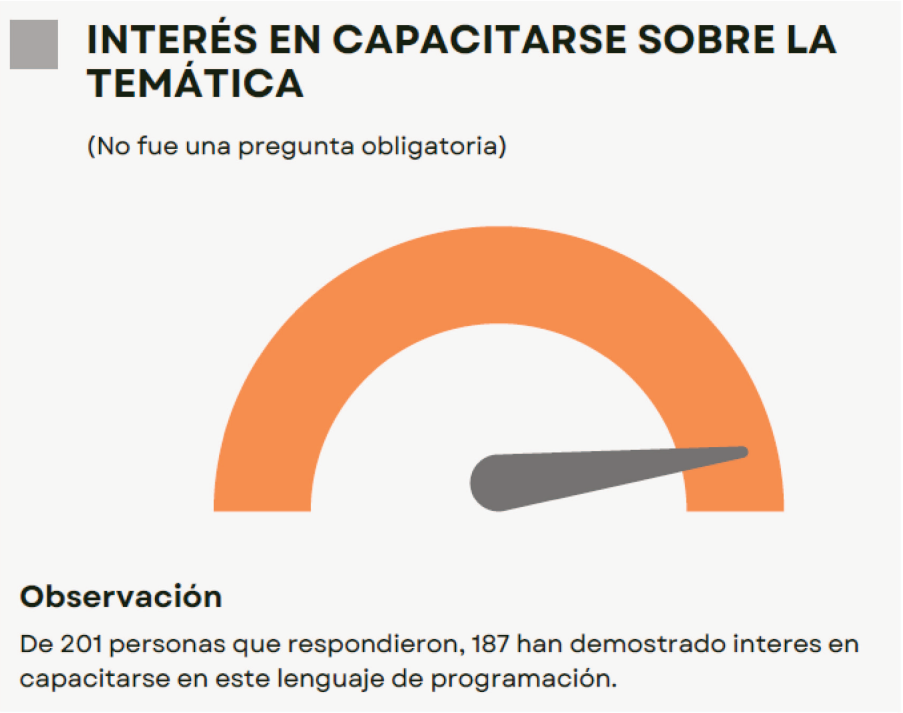


Figura 24: Respuesta a pregunta 11

¿Querés dejarnos algún comentario? (respuesta breve)

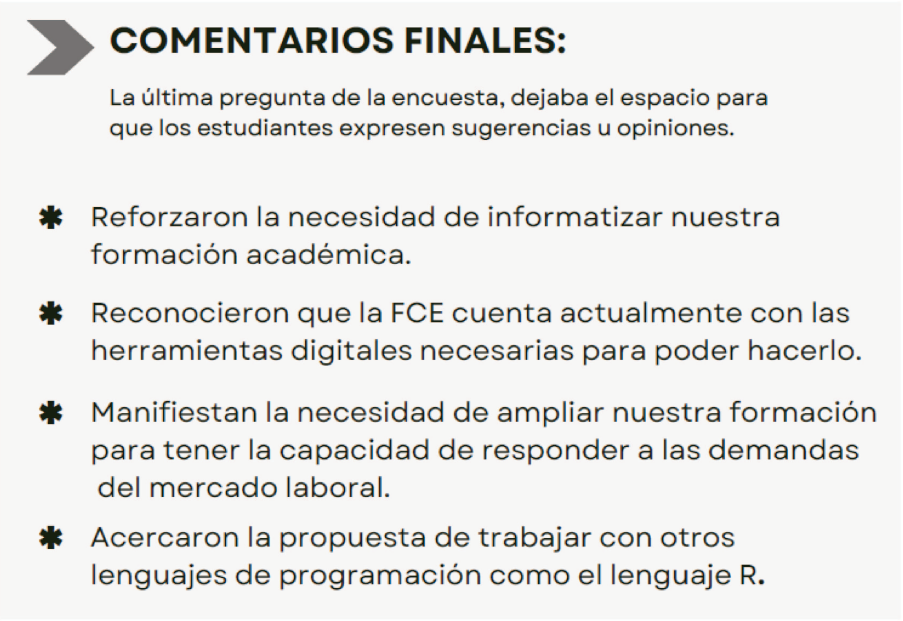


Figura 25: Respuesta a pregunta 12

16. La posición entre especialistas en lenguajes de programación acerca de su experiencia en aplicaciones orientada a la actividad profesional de las ciencias económicas

El día 21 de octubre de 2022 se llevó a cabo una entrevista a Federico Brun, analista en sistemas, quien se desempeña como full developer en una reconocida empresa de finanzas y ha dictado cursos de posgrado en la Universidad Católica de Córdoba. Se deja expresa constancia que el entrevistado ha prestado su conformidad para ser mencionado en el presente trabajo y se agradece por la desinteresada y valiosa colaboración brindada.

El profesional entrevistado dejó interesantes puntos de vistas para reforzar la línea de investigación de este proyecto entre las que destacamos que "Python es un lenguaje de Programación con Orientación a Objetos, fácil de aprender, con una curva de aprendizaje amigable para personas sin experiencia previa". En particular, respecto a su potencialidad enfocada a las tareas profesionales sostiene que "parece muy interesante detectar cómo diferentes profesionales con quienes he trabajado (Ingenieros, Contadores, Administradores, Economistas, Analistas de BI), pudieron ver un incremento en su productividad luego de aprender a programar en Python, y desarrollado herramientas propias para dar soporte a sus tareas".

2-Edad:

26 años 3- Género: Masculino 4- Formación alcanzada. Analista de Sistemas. 5- Ocupación laboral actual. Fullstack Developer en Ripio.com 6- Experiencia docente. Instructor del programa "Introducción a Python para Finanzas y Análisis de Datos" de ICDA.

7- ¿Qué lenguajes de programación considera que el mercado laboral argentino demandará en los próximos años?

Python y JavaScript.

8- ¿Por qué razón aconsejaría aprender un lenguaje de programación a personas sin experiencia previa.

Porque ya se ha convertido en una herramienta indispensable para que el profesional pueda aportar valor con sus tareas y diferenciarse en el ámbito de sus competencias, reduciendo costos, tiempos y mejorando la performance.

¿Cómo podría definir a PYTHON como lenguaje de programación?

Lenguaje de Programación con Orientación a Objetos, fácil de aprender, con una curva de aprendizaje amigable para personas sin experiencia previa.

10- Utilice 8 adjetivos para calificarlo y desarrolle brevemente 3 de ellos.

- a. Potente
- b. Rápido
- c. Sencillo
- d. Amigable
- e. Versátil

11- Que ventajas y desventajas mencionarías en relación a otros lenguajes de programación que conozcas.

Según mi experiencia lo considero mucho mas fácil de aprender respecto a otros lenguajes, ya que desde su concepción es muy intuitivo. Como desventaja diaria que no es aconsejable para desarrollos para plataformas mobile, como si lo es por ejemplo JavaScript, lo que hace que sea todavía más versátil.

12- En particular que grado de dificultad le asignarías al aprendizaje de Python para un principiante en la programación.

Considerando una escala de 1 a 10, diría que para un principiante sería un 4 en complejidad. Siempre pensando en los fundamentos y conceptos básicos.

13- Considera a Python como un lenguaje apropiado para dar soporte al desarrollo profesional, de actividades no directamente vinculadas a las ciencias de la computación.

Absolutamente. Evidencia suficiente de esto es el interés por capacitaciones afines de un grupo demográfico muy amplio en términos de edad y profesión.

14- Tiene ejemplos para dar que conozca o haya desarrollado. ¿En qué prácticas profesionales considera de mayor utilidad su aplicación?

Dejando de lado mi campo que es el desarrollo de software, me parece muy interesante detectar cómo diferentes profesionales con quienes he trabajado (Ingenieros, Contadores, Administradores, Economistas, Analistas de BI), pudieron ver un incremento en su productividad luego de aprender a programar en Python, y desarrollado herramientas propias para dar soporte a sus tareas.

15- En particular aplicado a las Ciencias Económicas, en la labor del Contador Publico como profesional independiente, ¿qué tareas o acciones le parece que puedan ser potenciadas con el uso de un programa como Python?

Generación de reportes, trámites de AFIP, análisis de balances y cashflow, controles de stock, análisis de tendencias y proyecciones.

16- ¿Ha recibido consultas o intereses de profesionales en Ciencias Económicas para el desarrollo o aplicación de Python en su practica profesional. Podría mencionar algún caso?

Si. Un promotor de seguros y ejecutivo de cuenta de inversiones me pidió que desarrollemos un software a medida para automatizar sus flujos de trabajo y poder realizar diferentes análisis de datos.

17- De las siguientes tareas que pueden ser automatizadas con Python que reflexión breve le merece cada una Leer, compatibilizar y procesar automáticamente archivos CSV y de Excel. Reduce tiempos, costos, y errores. Manipular el contenido de archivos PDF. Permite generar reportes de alta calidad técnica. Generar acciones y procesos rutinarios. Reduce tiempos, costos, y errores. Automatización de tareas en el Sitio de AFIP (Descargar, y procesar archivos por ej. facturas electrónicas). Reduce tiempos, costos, y errores. Podría citar otras...

18- De 1 (mín.) a 5 (máx.) valorice cada lenguaje como óptimo para su introducción en la currícula de las carreras universitarias de Ciencias Económicas.

Lenguaje	Calificación				
	1	2	3	4	5
Java	■				
Visual Basic	■				
C		■			
Ruby	■				
C#		■			
Python					■
C++			■		
Otro					

19- De los mejores puntuados anteriormente en que etapa y forma considera mas conveniente su incorporación

- En la carrera de grado como
- En la carrera de grado como obligatorio. ✓
- Como curso específico de capacitación externo
- Como curso dentro de una carrera de posgrado
- En la carrera de grado como optativo
- En la carrera de grado como obligatorio.
- Como curso específico de capacitación externo
- Como curso dentro de una carrera de posgrado

20- Una con flechas los lenguajes con las ventajas o atributos que mejor lo caractericen. Puede haber mas de una.

21- De 1 (min) a 5 (max) valorice cada una de las siguientes afirmaciones relacionadas a Python.

Lenguaje	Ventajas
Java	Eficiencia del Lenguaje
Visual Basic	Existencia de Desarrolladores
C	Velocidad del desarrollo
Ruby	Mano de obra calificada
C#	Demanda del mercado
Python	Tendencia del mercado
C++	Bajo costo/Software libre

Afirmación	Calificación				
	1	2	3	4	5
Para alguien que trabaja en contabilidad en este momento, que tiene que obtener datos de varios lugares (por ejemplo, SAP o alguna otra base de datos corporativa) y unirlos en Excel, aprender Python debería ser extremadamente convincente. Simplemente copiar y pegar datos de varios archivos grandes de Excel en una nueva hoja de cálculo generará mucha frustración. Y si necesita repetir esta operación cada pocos días, su única opción es hacerlo todo de nuevo, manualmente. En Python, puede hacer esto en unas pocas líneas de código y, una vez escrito, puede reutilizar el código repetidamente					✓
Para la mayoría de las personas, su computadora es solo un dispositivo en lugar de una herramienta. Pero al aprender a programar, obtendrás acceso a una de las herramientas más poderosas del mundo moderno y te divertirás en el camino				✓	
Las bibliotecas de Python son colecciones de código que le permiten ejecutar diferentes operaciones en sus datos sin escribir todo el código para esas operaciones. Casi siempre son de uso gratuito y de código abierto (generalmente son desarrollados por comunidades de programadores, ingenieros, científicos o entusiastas que contribuyen con su tiempo y experiencia porque beneficia a todos (a ellos mismos y a la comunidad en general). Usted también puede contribuir.					✓

Si Ud. usa Python con Excel, usted estará capacitado para usar un lenguaje de programación que es bueno para todos los aspectos de esta historia, ya sea la automatización de Excel, el acceso y preparación de conjuntos de datos, o el desarrollo de análisis de datos o tareas de visualización, Mas importante, es que Ud. puede reutilizar sus habilidades en Python por fuera de Excel: Si necesita escalar el poder computacional, Ud. podrá fácilmente mover su modelo cuantitativo, de simulación o aplicación de machine learning a la nube, donde lo estarán esperando recursos informáticos prácticamente ilimitados					✓
El primer inconveniente es que el trabajo que haces en Excel no es fácil reproducibles. Con Excel, no tiene un registro de todos los pasos que tomó en su análisis, por lo que no puede volver a ejecutar esos pasos si algo sale mal o si obtiene un nuevo conjunto de datos que necesita el mismo tipo de análisis Repetir las mismas acciones una y otra vez, cada vez los cambios de datos o los bloqueos de Excel no solo consumen mucho tiempo y molesto, pero extremadamente propenso a errores.		✓			
Python te permite redefinir tu trabajo. Pasas de ser un "mono que tipea y clickea.ª un desarrollador que usa su cerebro para decirle a la computadora lo que tiene que tipear y/o clickear				✓	

22- ¿En su ámbito laboral se está pensando en utilizar un nuevo lenguaje de programación en la organización en los próximos años? ¿Cuál?

No

23- ¿Que bibliografía recomendaría para su estudio en general y en particular en casos de utilidad para las profesiones de las ciencias economías.

- Python for Finance, Yves Hilpisch.
- Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners, Al Sweigart.

24- Si fuera de su interés, puedes dejar algún comentario u opinión libre que te genere haber participado de estas encuesta.

Como desarrollador, me parece muy interesante ver cómo aumenta la demanda de forma natural y orgánica, en otros ámbitos no relacionados a la tecnología y ciencias informáticas.

16.1. Síntesis de opiniones

La vinculación con otros especialistas en programación de la ciudad de Santa Fe permitió la realización de otras tres entrevistas personales, basadas en el mismo cuestionario anterior.

Las similitudes de respuestas permite ponerlas en una apretada síntesis visual como lo permite la técnica de nube de palabras, la cual fue realizada también utilizando algunas líneas de código utilizando Python.

Figura 26: Nube de palabras con otras tres entrevistas

16.2. ¿Que dice la Inteligencia artificial?

Pregunta: ¿Cómo te presentarías?

ChatGPT: ¡Hola! Soy ChatGPT, un modelo de lenguaje desarrollado por OpenAI lla-

mado GPT-3.5. Estoy aquí para ayudarte proporcionándote información, respondiendo preguntas y asistiéndole en lo que necesites. ¿En qué puedo ayudarte hoy?

Pregunta: ¿Qué lenguajes de programación considera que el mercado laboral argentino demandará en los próximos años?

ChatGPT: Aunque no tengo información actualizada después de enero de 2022, puedo proporcionarte algunas tendencias generales que eran relevantes hasta ese momento. Sin embargo, ten en cuenta que la demanda de lenguajes de programación puede variar según la industria y las necesidades específicas de cada empresa.

Hasta 2022, algunos de los lenguajes de programación más demandados incluían:

JavaScript: Ampliamente utilizado para el desarrollo web, tanto en el frontend como en el backend (Node.js).

Python: Versátil y utilizado en una variedad de aplicaciones, como desarrollo web, inteligencia artificial, análisis de datos y automatización.

Java: Aún es una opción sólida para el desarrollo empresarial, especialmente en grandes sistemas.

PHP: Aunque ha perdido popularidad en comparación con años anteriores, sigue siendo utilizado en el desarrollo web, especialmente en la creación de sitios basados en WordPress.

C#: Principalmente asociado con el desarrollo en el entorno Microsoft, es utilizado para aplicaciones de escritorio, desarrollo web (a través de ASP.NET), y en el desarrollo de juegos con Unity.

Swift y Kotlin: Para desarrollo de aplicaciones móviles en iOS y Android, respectivamente.

SQL: No es un lenguaje de programación en el sentido tradicional, pero las habilidades en el manejo de bases de datos son muy demandadas.

Es importante mantenerse actualizado con las tendencias del mercado laboral y las nuevas tecnologías. Además, ten en cuenta que las habilidades blandas, como la resolución de problemas, la comunicación y la capacidad de trabajar en equipo, también son muy valoradas en cualquier industria.

Pregunta: ¿Por qué razón aconsejaría aprender un lenguaje de programación a personas sin experiencia previa?

ChatGPT: Existen varias razones sólidas por las cuales aconsejaría aprender un lenguaje de programación a personas sin experiencia previa:

Habilidades del Siglo XXI: En la era digital actual, la capacidad de programar es una habilidad valiosa y demandada en muchos campos. Incluso si no planeas convertirte en un desarrollador profesional, comprender los conceptos básicos de la programación puede ser beneficioso en diversas áreas.

Resolución de Problemas: La programación fomenta el pensamiento lógico y la resolución de problemas. Aprender a descomponer problemas complejos en pasos más pequeños y abordarlos de manera sistemática es una habilidad transferible a muchas situaciones de la vida cotidiana.

Creatividad e Innovación: Programar permite crear cosas nuevas, desde simples scripts hasta aplicaciones complejas. Es una forma de expresar creatividad y explorar ideas innovadoras. Aprender a programar te brinda las herramientas para convertir tus conceptos en realidad digital.

Entender la Tecnología: Vivimos en un mundo cada vez más tecnológico, y entender cómo funcionan las tecnologías que utilizamos a diario puede empoderarnos. Conocer un lenguaje de programación te da una comprensión más profunda de los sistemas informáticos y de la tecnología en general.

Flexibilidad Laboral: Las habilidades de programación son útiles en una variedad de campos y roles laborales. Muchas empresas buscan personas con conocimientos básicos de programación incluso en roles que no son exclusivamente de desarrollo de software, como en marketing, finanzas, y análisis de datos.

Empleabilidad y Oportunidades Laborales: La demanda de profesionales con habilidades en programación sigue siendo alta en muchos sectores. Aprender a programar puede mejorar significativamente tus perspectivas de empleo y abrir nuevas oportunidades laborales.

Autonomía y Autodidactismo: La programación fomenta el hábito del aprendizaje continuo y la capacidad de autodidactismo. La tecnología evoluciona rápidamente, y los programadores a menudo tienen que aprender nuevas tecnologías y enfoques por sí mismos.

Comprender el Mundo Digital: La programación te proporciona una comprensión más profunda de cómo funcionan las aplicaciones y servicios digitales que utilizamos a diario. Esto puede hacer que te sientas más capacitado y confiado en un mundo cada vez más digitalizado.

En resumen, aprender a programar no solo es beneficioso para aquellos que buscan carreras en tecnología, sino que también proporciona habilidades valiosas que son aplicables en diversos aspectos de la vida personal y profesional.

Pregunta: ¿Cómo podría definir a PYTHON como lenguaje de programación?

ChatGPT: Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. Aquí hay algunas características y aspectos clave que definen a Python:

Legibilidad y Simplicidad: Python se destaca por su sintaxis clara y legible. El diseño del lenguaje favorece la legibilidad del código, lo que facilita a los programadores escribir y mantener código.

Interpretado y de Alto Nivel: Python es un lenguaje interpretado, lo que significa que no se compila directamente a código máquina, sino que se ejecuta línea por línea. Además, al ser de alto nivel, proporciona abstracciones poderosas que simplifican la programación.

Multiparadigma: Python soporta múltiples estilos de programación, incluyendo programación orientada a objetos, programación imperativa y programación funcional. Esto brinda flexibilidad a los desarrolladores para elegir el enfoque que mejor se adapte a sus necesidades.

Amplia Biblioteca Estándar: Python incluye una biblioteca estándar extensa que abarca desde operaciones básicas hasta funcionalidades avanzadas. Esto facilita el desarrollo de aplicaciones sin tener que escribir código desde cero para muchas tareas comunes.

Portabilidad: Python es compatible con varias plataformas, lo que significa que el código escrito en Python puede ejecutarse en diferentes sistemas operativos sin modificaciones significativas.

Dinámicamente Tipado: Python es un lenguaje de tipado dinámico, lo que significa que no es necesario declarar el tipo de una variable antes de usarla. Esto proporciona flexibilidad, pero también requiere una gestión cuidadosa de los tipos de datos.

Comunidad Activa y Ecosistema Vibrante: Python cuenta con una comunidad activa de desarrolladores y una amplia variedad de bibliotecas y frameworks que facilitan el desarrollo de aplicaciones en diversos dominios, como desarrollo web, ciencia de datos, inteligencia artificial, entre otros.

Desarrollo Rápido de Prototipos y Productividad: Gracias a su sintaxis clara y a las características que fomentan la legibilidad, Python es adecuado para el desarrollo rápido de prototipos. Además, su énfasis en la productividad hace que sea popular en proyectos donde la velocidad de desarrollo es crucial.

En resumen, Python es un lenguaje versátil que ha ganado popularidad debido a su legibilidad, simplicidad, flexibilidad y amplio soporte en diversos campos de la informática.

Pregunta: Utilice 8 adjetivos para calificarlo y desarrolle brevemente 3 de ello.

ChatGPT: Adjetivos para Calificar a Python:

Versátil: Python es versátil ya que se adapta a una variedad de aplicaciones y entornos, desde desarrollo web y scripting hasta inteligencia artificial y análisis de datos.

Legible: La sintaxis clara y legible de Python facilita la comprensión del código, lo que contribuye a un desarrollo y mantenimiento más sencillos.

Potente: A pesar de su simplicidad aparente, Python es un lenguaje potente que permite a los desarrolladores abordar problemas complejos y realizar tareas avanzadas.

Amigable para Principiantes: Python es conocido por ser amigable para principiantes, lo que lo convierte en un lenguaje ideal para aquellos que se inician en la programación.

Productivo: La sintaxis concisa y las estructuras de datos integradas hacen que el desarrollo en Python sea eficiente y productivo, permitiendo a los programadores lograr más con menos código.

Escalable: Python es escalable, lo que significa que se puede utilizar en proyectos pequeños y simples, así como en aplicaciones grandes y complejas.

Comunidad Activa: La comunidad activa de Python contribuye al desarrollo constante del lenguaje, brindando soporte, actualizaciones y una amplia variedad de recursos.

Innovador: Python se utiliza en tecnologías innovadoras como inteligencia artificial, aprendizaje profundo y desarrollo de aplicaciones web modernas, lo que lo posiciona como un lenguaje innovador en el mundo de la programación.

Desarrollo Breve de Tres Adjetivos:

Versátil: Python destaca por su versatilidad, ya que es aplicable en una amplia gama de dominios. Desde la automatización de tareas hasta el desarrollo de aplicaciones complejas, Python es una herramienta versátil que satisface diversas necesidades.

Legible: La legibilidad del código en Python es fundamental. La sintaxis clara y estructurada facilita la colaboración entre desarrolladores y la comprensión del código incluso para aquellos que no lo han escrito originalmente. Esta legibilidad contribuye a la mantenibilidad a largo plazo de los proyectos.

Potente: A pesar de su simplicidad aparente, Python es un lenguaje potente. Es capaz

de manejar problemas complejos y está equipado con bibliotecas y frameworks que permiten a los desarrolladores abordar una amplia variedad de desafíos, desde análisis de datos hasta desarrollo de inteligencia artificial. La potencia de Python reside en su capacidad para ofrecer soluciones robustas y eficientes.

Pregunta: ¿Qué ventajas y desventajas mencionarías en relación a otros lenguajes de programación que conozcas?

ChatGPT: Ventajas de Python:

Sintaxis Clara y Legible: La sintaxis de Python es intuitiva y fácil de leer, lo que facilita el aprendizaje y la comprensión del código.

Desarrollo Rápido: La productividad es alta en Python debido a su sintaxis concisa y a la amplia variedad de bibliotecas disponibles, lo que acelera el desarrollo de prototipos y aplicaciones.

Amplia Comunidad y Ecosistema: Python tiene una comunidad activa y un ecosistema vibrante con una gran cantidad de bibliotecas y frameworks, lo que facilita el acceso a soluciones y recursos.

Versatilidad: Python se utiliza en diversos campos, desde desarrollo web hasta inteligencia artificial y análisis de datos, lo que lo convierte en un lenguaje versátil.

Multiplataforma: Python es compatible con múltiples plataformas, lo que significa que el código puede ejecutarse en diferentes sistemas operativos sin modificaciones significativas.

Desventajas de Python:

Rendimiento Relativo: En comparación con lenguajes de bajo nivel como C++ o Rust, Python puede ser menos eficiente en términos de rendimiento, especialmente en aplicaciones que requieren velocidad y recursos intensivos.

Gestión de Tipos Dinámicos: Aunque la gestión de tipos dinámicos brinda flexibilidad, puede resultar en errores difíciles de detectar durante la ejecución.

Limitaciones en Desarrollo Móvil: Aunque existen frameworks como Kivy y BeeWare para desarrollo móvil en Python, este no es tan dominante como Java (para Android) o Swift (para iOS).

Problemas de Escalabilidad: Aunque Python es escalable, puede haber limitaciones en el manejo de grandes volúmenes de tráfico en comparación con lenguajes altamente eficientes

como Go.

GIL (Global Interpreter Lock): En el contexto de aplicaciones de subprocesos múltiples, el GIL puede limitar el rendimiento, ya que solo permite que un hilo ejecute instrucciones a la vez.

La elección de un lenguaje de programación depende del contexto y los requisitos específicos del proyecto. Python es excelente para muchos escenarios, pero hay situaciones en las que otros lenguajes pueden ser más adecuados en términos de rendimiento, control o dominio de aplicación específico.

Pregunta: ¿En particular qué grado de dificultad le asignarías al aprendizaje de Python para un principiante en la programación?

ChatGPT: Python es ampliamente considerado como uno de los lenguajes de programación más amigables para principiantes, y muchos educadores lo eligen como el primer lenguaje para enseñar programación. Aquí hay algunas razones por las cuales Python es considerado accesible para los principiantes:

Bajas Barreras de Entrada:

Sintaxis Clara y Legible: La sintaxis de Python es intuitiva y se asemeja al lenguaje natural, lo que facilita su lectura y escritura. Ausencia de Punto y Coma: Python no requiere el uso de punto y coma al final de las líneas, lo que reduce la complejidad sintáctica para los principiantes.

Amplios Recursos de Aprendizaje:

Documentación Clara: La documentación oficial de Python es extensa y clara, proporcionando información detallada y ejemplos útiles. Comunidad Activa: La comunidad de Python es acogedora y ofrece una gran cantidad de recursos en línea, tutoriales y foros de ayuda.

Versatilidad y Aplicaciones Prácticas:

Versatilidad de Aplicaciones: Python se utiliza en una variedad de campos, desde desarrollo web y automatización hasta inteligencia artificial y análisis de datos. Esto permite a los principiantes aplicar sus habilidades en diferentes áreas según sus intereses.

Frameworks y Bibliotecas Amigables:

Frameworks Intuitivos: Hay frameworks y bibliotecas en Python diseñados para simplificar tareas específicas, como Flask y Django para desarrollo web, lo que facilita a los principiantes

dar sus primeros pasos en la creación de aplicaciones.

En general, el grado de dificultad para aprender Python como principiante se considera bajo en comparación con muchos otros lenguajes de programación. Sin embargo, la facilidad de aprendizaje puede variar según la experiencia y el enfoque del individuo. La clave para un aprendizaje exitoso es la práctica constante y la aplicación práctica de los conceptos aprendidos. Con recursos de calidad y un enfoque práctico, la mayoría de los principiantes encuentran que Python es un lenguaje accesible y gratificante para comenzar su viaje en la programación.

Pregunta: ¿Considera a Python como un lenguaje apropiado para dar soporte al desarrollo profesional, de actividades no directamente vinculadas a las ciencias de la computación?

ChatGPT: Sí, Python es ampliamente considerado como un lenguaje apropiado para dar soporte al desarrollo profesional en una variedad de actividades no directamente vinculadas a las ciencias de la computación. Aquí hay algunas razones clave:

Automatización de Tareas: Python es excelente para automatizar tareas repetitivas y simplificar procesos en diversas áreas profesionales. Puede utilizarse para crear scripts que ahorren tiempo en la manipulación y procesamiento de datos, la gestión de archivos, y otras operaciones.

Análisis de Datos: Python es muy popular en el ámbito del análisis de datos y la ciencia de datos. Puede ayudar en la manipulación, limpieza y visualización de datos, proporcionando insights valiosos para la toma de decisiones en negocios, marketing, finanzas, y otras disciplinas.

Desarrollo Web y Creación de Prototipos: Python cuenta con frameworks como Flask y Django que facilitan el desarrollo web. Esto permite a profesionales no especializados en programación crear prototipos de aplicaciones web o sitios con relativa facilidad.

Automatización de Informes y Generación de Documentos: Python puede utilizarse para automatizar la generación de informes y documentos, lo que es útil en entornos empresariales y administrativos.

Integración con Herramientas Existentes: Python es compatible con una amplia gama de bibliotecas y APIs, lo que facilita la integración con herramientas y servicios existentes, independientemente del sector profesional.

Machine Learning e Inteligencia Artificial (IA): Python es líder en el ámbito de la inteligencia artificial y el machine learning, y muchas empresas están utilizando estas tecnologías para mejorar procesos y tomar decisiones más informadas.

Facilidad de Aprendizaje: Python es conocido por su sintaxis clara y legible, lo que

facilita su aprendizaje para aquellos que no tienen experiencia en programación.

Comunidad y Recursos: La amplia comunidad de Python significa que hay abundantes recursos disponibles, incluyendo tutoriales, documentación y foros de ayuda, lo que facilita el aprendizaje y la resolución de problemas.

En resumen, Python es un lenguaje versátil que puede ser beneficioso en una variedad de campos profesionales, permitiendo a los individuos automatizar tareas, analizar datos, desarrollar prototipos y más, independientemente de su especialización inicial. Su accesibilidad y amplio conjunto de bibliotecas hacen que sea una elección sólida para aquellos que desean mejorar su eficiencia y capacidad en el ámbito laboral.

Pregunta: ¿Tiene ejemplos para dar que conozca o haya desarrollado? ¿En qué prácticas profesionales considera de mayor utilidad su aplicación?

ChatGPT: Aunque no tengo la capacidad de recordar o almacenar información específica sobre proyectos o ejemplos específicos que haya desarrollado, puedo proporcionarte ejemplos generales de cómo Python se utiliza en prácticas profesionales y en qué áreas puede ser de mayor utilidad:

Automatización de Tareas Administrativas: Desarrollar scripts para la automatización de tareas administrativas como la generación de informes, el procesamiento de archivos y la gestión de datos.

Análisis de Datos y Ciencia de Datos: Utilizar Python con bibliotecas como Pandas, NumPy y Matplotlib para analizar y visualizar datos, generando informes y gráficos significativos.

Desarrollo Web y Prototipos: Crear aplicaciones web o prototipos rápidos utilizando frameworks como Flask o Django, incluso por profesionales no especializados en desarrollo.

Automatización de Pruebas y Control de Calidad: Desarrollar scripts para automatizar pruebas de software y garantizar la calidad del código en el desarrollo de software.

Automatización de Marketing y SEO: Implementar scripts para automatizar tareas de marketing, como la extracción de datos de redes sociales o la optimización de contenido para motores de búsqueda (SEO).

Procesamiento de Texto y Procesamiento del Lenguaje Natural (PLN): Aplicar Python en el análisis y procesamiento de texto para tareas como la extracción de información, el análisis de sentimientos y la creación de modelos de lenguaje natural.

Machine Learning e Inteligencia Artificial (IA): Utilizar bibliotecas como TensorFlow

y scikit-learn para implementar soluciones de machine learning en áreas como pronósticos financieros, recomendaciones personalizadas y reconocimiento de patrones.

Automatización de Flujos de Trabajo Empresariales: Desarrollar scripts o pequeñas aplicaciones para automatizar flujos de trabajo internos, desde la gestión de recursos humanos hasta la logística empresarial.

Desarrollo de Herramientas de Productividad: Crear herramientas personalizadas para mejorar la productividad, como scripts que simplifican tareas repetitivas o aplicaciones internas para equipos específicos.

Automatización en Finanzas y Análisis Económico:

Utilizar Python para analizar datos financieros, realizar pronósticos económicos y automatizar procesos en el sector financiero.

Estos son solo ejemplos generales, y la utilidad de Python en prácticas profesionales puede extenderse a una amplia variedad de campos. Su versatilidad y facilidad de uso lo convierten en una herramienta valiosa en diversos entornos laborales.

Pregunta: ¿En particular aplicado a las Ciencias Económicas, en la labor del Contador Público como profesional independiente, ¿qué tareas o acciones le parece que puedan ser potenciadas con el uso de un programa como Python?

ChatGPT: En el ámbito de las Ciencias Económicas y la labor del Contador Público, Python puede ser una herramienta valiosa para potenciar diversas tareas y acciones. Aquí hay algunas áreas específicas donde Python podría ser aplicado de manera beneficiosa:

Análisis de Datos Financieros: Utilizar Python junto con bibliotecas como Pandas y NumPy para realizar análisis detallados de estados financieros, identificar tendencias, y generar informes financieros más rápidos y precisos.

Automatización de Tareas Contables: Desarrollar scripts para automatizar tareas contables rutinarias, como la conciliación de cuentas, la clasificación de transacciones y la generación de informes contables.

Proyecciones y Pronósticos: Implementar modelos de proyección financiera utilizando bibliotecas de machine learning o métodos estadísticos en Python para ayudar en la toma de decisiones basadas en datos.

Automatización de Informes Tributarios: Desarrollar scripts que automatizan la preparación de informes tributarios, ayudando a asegurar la conformidad con las normativas fiscales y

reduciendo la posibilidad de errores humanos.

Análisis de Costos y Rentabilidad: Utilizar Python para analizar costos y evaluar la rentabilidad de productos o servicios, permitiendo una toma de decisiones más informada.

Gestión de Facturas y Pagos: Desarrollar herramientas para gestionar el flujo de facturas, seguimiento de pagos y generación de informes de cuentas por cobrar y cuentas por pagar.

Automatización de Procesos de Conciliación Bancaria: Crear scripts para automatizar procesos de conciliación bancaria, comparando automáticamente los registros contables con los extractos bancarios.

Desarrollo de Dashboards Financieros: Implementar dashboards interactivos utilizando bibliotecas como Plotly o Dash para visualizar datos financieros y proporcionar una visión más clara de la situación financiera.

Análisis de Riesgos y Fraudes: Utilizar Python para desarrollar modelos analíticos que ayuden en la detección de riesgos financieros y posibles fraudes a través del análisis de patrones y anomalías.

Integración con Sistemas ERP: Desarrollar scripts para la integración y automatización de datos entre sistemas ERP y otras herramientas contables, mejorando la eficiencia y la consistencia de la información.

La aplicación de Python en el ámbito contable puede ser especialmente útil para profesionales independientes, ya que les permite optimizar procesos, reducir la carga manual de trabajo y ofrecer análisis más avanzados para apoyar la toma de decisiones estratégicas.

Pregunta ¿Ha recibido consultas o intereses de profesionales en Ciencias Económicas para el desarrollo o aplicación de Python en su práctica profesional? ¿Podría mencionar algún caso?

ChatGPT: No tengo la capacidad de recordar interacciones individuales ni detalles específicos de las consultas que recibo, ya que mi diseño garantiza la privacidad y confidencialidad de las interacciones. Cada conversación se trata como única y no se almacena información personal o datos específicos del usuario.

Sin embargo, puedo decirte que es común que profesionales en Ciencias Económicas, incluyendo Contadores Públicos y economistas, muestren interés en el uso de Python y otras tecnologías en su práctica profesional. Muchos de ellos buscan aprovechar las ventajas de la automatización, análisis de datos avanzado y la optimización de procesos para mejorar la eficiencia y precisión en sus tareas cotidianas.

La creciente popularidad de Python en campos como la ciencia de datos, análisis financiero y automatización empresarial ha llevado a un aumento en el interés de profesionales en Ciencias Económicas por aprender y aplicar este lenguaje en sus actividades laborales. La comunidad y los recursos en línea ofrecen un amplio respaldo para aquellos que buscan integrar Python en su práctica profesional.

Pregunta: De las siguientes tareas que pueden ser automatizadas con Python que reflexión breve le merece cada una:

- Leer, compatibilizar y procesar automáticamente archivos CSV y de Excel.
- Manipular el contenido de archivos PDF.
- Generar acciones y procesos rutinarios.
- Automatización de tareas en el Sitio de AFIP (Descargar, y procesar archivos por ej. facturas electrónicas).
- Podría citar otras...

ChatGPT: Leer, compatibilizar y procesar automáticamente archivos CSV y de Excel: Automatizar la lectura y manipulación de archivos CSV y Excel con Python es una práctica común y altamente beneficiosa. Python ofrece bibliotecas como Pandas que simplifican estas tareas, permitiendo la manipulación de datos de manera eficiente. Esto puede ahorrar tiempo significativo al tratar con grandes conjuntos de datos y facilita la integración de datos de múltiples fuentes.

Manipular el contenido de archivos PDF: La manipulación de archivos PDF mediante Python, utilizando bibliotecas como PyPDF2 o pdfplumber, permite extraer texto, imágenes y metadatos de documentos PDF. Esto es útil para procesos como la extracción de datos de informes financieros, la creación de resúmenes automáticos y la automatización de tareas relacionadas con documentos PDF.

Generar acciones y procesos rutinarios: Python es excelente para automatizar acciones y procesos rutinarios mediante la creación de scripts o pequeñas aplicaciones. Esto puede incluir la ejecución de tareas diarias, la programación de copias de seguridad, la automatización de procesos de informes y otras actividades que se repiten con regularidad. La automatización de tareas rutinarias libera tiempo para que los profesionales se centren en tareas más estratégicas.

Automatización de tareas en el Sitio de AFIP: Automatizar tareas en el sitio de AFIP (Administración Federal de Ingresos Públicos de Argentina) con Python puede ser especialmente valioso para los profesionales en Ciencias Económicas. Esto puede incluir la

descarga y procesamiento automático de archivos, como facturas electrónicas, generando eficiencia en la gestión tributaria y cumplimiento de obligaciones fiscales.

Otras tareas posibles:

Automatización de correos electrónicos: Enviar correos electrónicos automáticos, por ejemplo, para recordatorios de pagos o comunicaciones periódicas. Extracción de datos de la web: Utilizar herramientas como Beautiful Soup o Scrapy para extraer información específica de páginas web relevantes para la toma de decisiones. Automatización de informes y análisis: Generar automáticamente informes detallados y análisis a partir de conjuntos de datos, facilitando la presentación de resultados y la toma de decisiones informada.

En general, la automatización con Python puede transformar significativamente la eficiencia y precisión de diversas tareas en el ámbito de las Ciencias Económicas, permitiendo a los profesionales dedicar más tiempo a actividades estratégicas y analíticas. La versatilidad de Python y su amplio ecosistema de bibliotecas facilitan la implementación de soluciones personalizadas para una amplia variedad de necesidades.

Pregunta: De 1 (min) a 5 (max) valorice cada lenguaje como óptimo para su introducción en la currícula de las carreras universitarias de Ciencias Económicas. Java Visual Basic C Ruby C# Python C++ Otro **ChatGPT**: La valoración de la idoneidad de un lenguaje de programación en la currícula de Ciencias Económicas puede depender de varios factores, incluyendo la versatilidad del lenguaje, la demanda en la industria y la facilidad de aprendizaje. A continuación, proporciono una valoración general del 1 al 5 para cada lenguaje mencionado:

Java: 4, es un lenguaje versátil y ampliamente utilizado en la industria, especialmente en entornos empresariales. Tiene una sintaxis más estricta, pero proporciona portabilidad y escalabilidad.

Visual Basic: 2, puede ser útil para aplicaciones de escritorio, pero no es tan popular en la actualidad. La demanda y versatilidad son limitadas en comparación con otros lenguajes.

C: 3, es un lenguaje de bajo nivel que puede proporcionar una comprensión sólida de los fundamentos de programación, pero su sintaxis puede ser menos amigable para principiantes.

Ruby: 3, es conocido por su elegancia y facilidad de lectura, pero su uso no es tan extendido como otros lenguajes. Es especialmente útil en el desarrollo web con el framework Ruby on Rails.

C#: 4, es un lenguaje de programación de propósito general desarrollado por Microsoft. Es ampliamente utilizado en el desarrollo de aplicaciones empresariales y en la plataforma .NET.

Python: 5, es muy recomendado para principiantes y es extremadamente versátil. Su sintaxis clara y amplio uso en ciencia de datos, inteligencia artificial y desarrollo web hacen que sea una elección fuerte.

C++: 3, es poderoso y se utiliza en sistemas de alto rendimiento y juegos, pero su complejidad puede ser un desafío para principiantes.

Otro: Depende del lenguaje específico.

La elección de otros lenguajes puede depender de las necesidades específicas del plan de estudios y de las tendencias de la industria en el momento.

Es importante considerar el equilibrio entre introducir a los estudiantes en conceptos de programación y proporcionar habilidades prácticas que sean relevantes en la industria actual. La elección de lenguajes también puede depender de los objetivos específicos del programa académico y de la disponibilidad de recursos de enseñanza.

Pregunta: Podrías vincular los lenguajes mencionados anteriormente con las siguientes afirmaciones: Eficiencia del lenguaje Fácil aprendizaje Existencia de desarrolladores Velocidad de desarrollo Mano de obra calificada Demanda del mercado Tendencia del mercado Bajo costo/ Software Libre

ChatGPT: Claro, vinculemos los lenguajes de programación mencionados con las siguientes afirmaciones:

Eficiencia del Lenguaje: C, C++, C#: Estos lenguajes suelen ser eficientes en términos de velocidad de ejecución y gestión de recursos, especialmente C y C++ en aplicaciones de bajo nivel. Java, Python, Ruby: Aunque pueden ser menos eficientes en términos de velocidad que C o C++, ofrecen un buen equilibrio entre eficiencia y facilidad de uso.

Fácil Aprendizaje: Python, Ruby, Visual Basic: Estos lenguajes son conocidos por tener una sintaxis clara y son más fáciles de aprender, especialmente para principiantes. C, C++, Java, C#: Pueden ser más desafiantes para principiantes debido a su sintaxis más estricta y conceptos más avanzados.

Existencia de Desarrolladores: Java, C#, Python: Estos lenguajes tienen una gran base de desarrolladores y comunidades activas, lo que facilita encontrar recursos y soluciones en línea. Ruby, Visual Basic, C, C++: Aunque tienen comunidades sólidas, pueden no ser tan extensas como las de Java, C# o Python.

Velocidad de Desarrollo: Python, Ruby, Visual Basic: Estos lenguajes suelen ser más rápidos para el desarrollo de prototipos y aplicaciones, gracias a su sintaxis concisa y su enfoque

en la productividad. C, C++, Java, C#: Pueden tener una curva de aprendizaje ligeramente más pronunciada, pero ofrecen un mayor control y eficiencia en términos de rendimiento.

Mano de Obra Calificada: Java, Python, C#: Hay una amplia disponibilidad de desarrolladores calificados en estos lenguajes, lo que facilita la contratación. Ruby, Visual Basic, C, C++: Aunque hay profesionales altamente calificados, puede haber menos disponibilidad que en los lenguajes más populares.

Demanda del Mercado: Java, Python, C#: Existe una fuerte demanda en el mercado laboral para profesionales que dominan estos lenguajes. Ruby, Visual Basic, C, C++: Aunque hay demanda, podría no ser tan alta como para los lenguajes más populares.

Tendencia del Mercado: Python, Java, C#: Estos lenguajes están en constante evolución y se utilizan en tendencias actuales como inteligencia artificial, desarrollo web y aplicaciones empresariales. Ruby, Visual Basic, C, C++: Aunque siguen siendo relevantes, su adopción puede ser menos prominente en las tendencias tecnológicas más recientes.

Bajo Costo/Software Libre: Python, Ruby: Estos lenguajes son de código abierto y tienen una comunidad activa que contribuye a su desarrollo. Java, C, C++, C#: Aunque Java es de código abierto, algunos entornos de desarrollo pueden no serlo. C y C++ también son de código abierto, pero C# está más vinculado a las tecnologías de Microsoft.

Es importante tener en cuenta que estas vinculaciones son generalizaciones y la elección de un lenguaje puede depender de los requisitos específicos del proyecto, la infraestructura existente y las metas profesionales individuales.

Pregunta: De 1 (min) a 5 (max) valorice cada una de las siguientes afirmaciones relacionadas a Python.

Para alguien que trabaja en contabilidad en este momento, que tiene que obtener datos de varios lugares (por ejemplo, SAP o alguna otra base de datos corporativa) y unirlos en Excel, aprender Python debería ser extremadamente convincente. Simplemente copiar y pegar datos de varios archivos grandes de Excel en una nueva hoja de cálculo generará mucha frustración. Y si necesita repetir esta operación cada pocos días, su única opción es hacerlo todo de nuevo, manualmente. En Python, puede hacer esto en unas pocas líneas de código y, una vez escrito, puede reutilizar el código repetidamente Para la mayoría de las personas, su computadora es solo un dispositivo en lugar de una herramienta. Pero al aprender a programar, obtendrás acceso a una de las herramientas más poderosas del mundo moderno y te divertirás en el camino Las bibliotecas de Python son colecciones de código que le permiten ejecutar diferentes operaciones en sus datos sin escribir todo el código para esas operaciones. Casi siempre son de uso gratuito y de código abierto (generalmente son desarrollados por comunidades de programadores, ingenieros, científicos o entusiastas que contribuyen con su tiempo y experiencia porque beneficia a todos

(a ellos mismos y a la comunidad en general). Usted también puede contribuir.

Si Ud. usa Python con Excel, usted estará capacitado para usar un lenguaje de programación que es bueno para todos los aspectos de esta historia, ya sea la automatización de Excel, el acceso y preparación de conjuntos de datos, o el desarrollo de análisis de datos o tareas de visualización, Mas importante, es que Ud. puede reutilizar sus habilidades en Python por fuera de Excel: Si necesita escalar el poder computacional, Ud. podrá fácilmente mover su modelo cuantitativo, de simulación o aplicación de machine learning a la nube, donde lo estarán esperando recursos informáticos prácticamente ilimitados El primer inconveniente es que el trabajo que haces en Excel no es fácil reproducibles. Con Excel, no tiene un registro de todos los pasos que tomó en su análisis, por lo que no puede volver a ejecutar esos pasos si algo sale mal o si obtiene un nuevo conjunto de datos que necesita el mismo tipo de análisis Repetir las mismas acciones una y otra vez, cada vez los cambios de datos o los bloqueos de Excel no solo consumen mucho tiempo y molesto, pero extremadamente propenso a errores. Python te permite redefinir tu trabajo. Pasas de ser un "mono que tipea y clickea.ª un desarrollador que usa su cerebro para decirle a la computadora lo que tiene que tipear y/o clickear.

Pregunta: Podría citar alguna otra....

ChatGPT: Voy a valorar cada afirmación del 1 al 5, según su persuasión para alguien que trabaja en contabilidad y se enfrenta a las tareas mencionadas:

Aprender Python para la automatización de tareas en Excel: Valoración: 5 Comentario: Python brinda una solución eficiente y reutilizable para la manipulación de datos, especialmente cuando provienen de diversas fuentes como SAP o bases de datos corporativas.

Computadora como herramienta poderosa al aprender a programar: Valoración: 5 Comentario: Destaca el poder transformador de aprender a programar, convirtiendo la computadora en una herramienta más allá de un simple dispositivo.

Utilidad de bibliotecas en Python: Valoración: 5 Comentario: Subraya la ventaja de las bibliotecas en Python para realizar operaciones sin tener que escribir todo el código, además de fomentar la contribución a la comunidad.

Capacitación en Python y Excel para aspectos diversos: Valoración: 5 Comentario: Destaca la versatilidad de Python en relación con Excel, permitiendo la automatización, el acceso a datos y la realización de análisis avanzados.

Dificultades de reproducibilidad en Excel: Valoración: 5 Comentario: Señala las limitaciones de Excel en términos de reproducibilidad y destaca cómo Python puede resolver este problema al proporcionar un registro de pasos.

Transformación del trabajo con Python: Valoración: 5 Comentario: Resalta el cambio de paradigma al pasar de tareas manuales en Excel a la programación en Python, empoderando al usuario.

Escalabilidad y flexibilidad de Python: Valoración: 5 Comentario: Subraya la capacidad de Python para escalar desde tareas de automatización hasta aplicaciones de machine learning en la nube.

Inconvenientes de trabajo en Excel no reproducibles: Valoración: 5 Comentario: Expone las dificultades inherentes a la falta de reproducibilidad en Excel y cómo Python aborda este problema.

En general, todas las afirmaciones reciben una valoración alta, indicando que Python ofrece soluciones convincentes y ventajas significativas para profesionales en contabilidad que deseen mejorar la eficiencia y la reproducibilidad de sus tareas.

Pregunta ¿Que bibliografía recomendaría para su estudio en general y en particular en casos de utilidad para las profesiones de las ciencias económicas?

ChatGPT: Recomendar bibliografía para aprender Python puede depender del nivel de experiencia del lector, sus objetivos específicos y el contexto de aplicación, como en el caso de profesionales en ciencias económicas. Aquí hay algunas sugerencias generales y específicas para las Ciencias Económicas: Libros Generales sobre Python:

"Python Crash Course"por Eric Matthes: Un libro ideal para principiantes que abarca desde conceptos básicos hasta proyectos prácticos.

."utomate the Boring Stuff with Python"por Al Sweigart: Enfocado en la automatización de tareas cotidianas, es práctico y accesible.

"Fluent Python"por Luciano Ramalho: Un libro más avanzado que explora las características avanzadas de Python, adecuado para lectores con cierta experiencia.

Libros para Ciencias Económicas:

"Python for Data Analysis"por Wes McKinney: Enfocado en el uso de Python para el análisis de datos, con énfasis en herramientas como Pandas y NumPy. Relevante para profesionales de ciencias económicas.

"Quantitative Financial Analytics with Python"por Tony Bell: Ofrece una perspectiva práctica sobre el uso de Python en análisis cuantitativos financieros.

"Python for Finance"por Yves Hilpisch: Dirigido a profesionales financieros, explora cómo utilizar Python en finanzas cuantitativas.

"Data Science for Business"por Foster Provost y Tom Fawcett: No es específico de Python, pero proporciona una visión general valiosa de cómo se aplica la ciencia de datos en los negocios, algo crucial para los profesionales de ciencias económicas.

Recursos en Línea:

Sitio web oficial de Python: La documentación oficial de Python es un recurso valioso para aprender y entender la sintaxis y las características del lenguaje.

Kaggle: Ofrece conjuntos de datos y kernels en Python, lo que permite aprender mediante proyectos prácticos y colaborar con la comunidad.

Real Python: Ofrece tutoriales, artículos y cursos para diferentes niveles de experiencia en Python.

Recuerda que la práctica es clave al aprender a programar. Los libros proporcionan una base sólida, pero aplicar tus conocimientos en proyectos reales es esencial para el desarrollo de habilidades.

17. Identificación de casos aplicables

Actividades dentro de la práctica profesional propiamente dicha cuyos resultados puedan ser extrapolados a diferentes situaciones de la actividad del Contador

17.1. Analizar los diferentes reportes de salida que utiliza el sistema de gestión de una empresa en cuanto a su estructura de presentación

El sistema de registración contable analizado presenta diferentes posibilidades de exportación de la información.

Una vez realizada la consulta esta se muestra en la pantalla de la computadora y ofrece posibilidades de ser impresa en papel mediante la conexión a través de una impresora física, también en formato PDF mediante una impresora virtual.

Estas opciones, si bien pueden tener cierta utilidad, no permite ser procesadas posteriormente mediante otros programas, de una manera eficiente y rápida.

Las otras opciones que presenta para exportar la información son en formato digital mediante archivos de salida tipo Microsoft Excel®o CVS.

La exportación de archivo bajo formato Microsoft Excel®, compila toda la información en la columna A de la hoja de cálculo por lo que es necesario posteriormente su separación en columnas mediante los comandos de Microsoft Excel®: Datos - Texto en columnas.

Debido a la falta de actualización del software contable los archivos generados se descargan en la versión antigüa de Microsoft Excel®97-2003, por lo que también luego es necesario operar para que sea guardado como la versión vigente en el ordenador en que se esté utilizando.

Estas tareas si bien son sencillas de llevar a cabo presentan una paso extra que puede ser subsanada fácilmente si interactuamos con el formato de exportación tipo CVS y las posibilidades que las librerías de Python ofrecen para este tipo de archivos.

17.2. Identificar aquellas prácticas que se realizan mediante planillas de cálculos y que tengan mayores riesgos de errores producto de la manipulación de los datos por acción humana para programarlas en lenguaje Python

Una tarea de repetición mensual que se analizó fue la creación de archivos que requiere el banco con el que se opera en la empresa para el pago de haberes de los empleados.

Este archivo debe ser generado en la computadora del empleador y subido mediante la página web del banco que realizará las transferencias con un formato TXT o CSV.

El origen de los datos proviene del sistema liquidador de sueldos del cual se obtiene el valor neto a depositar por cada empleado.

En función del CUIT como campo identificador del empleado, es necesario generar un archivo que identifique a cada uno, con el importe a cobrar y la cuenta bancaria unificada (CBU) a la cual se deben transferir los sueldos.

La generación de este archivo en forma manual puede generar riesgos asociados con la manipulación humana, como por ejemplo:

- Error en la carga de los importes a transferir desde la cuenta del empleador a la cuenta del empleado.
- Alterar los importes a transferir entre diferentes empleados.
- Generar el archivo a subir con inconsistencias que impidan su correcta aceptación por parte de la web del banco.

Estas acciones se justifican mas cuánto mayor sea la cantidad de empleados a procesar toda vez que la información debe ser cotejada desde una fuente de información para ser procesada en otro tipo de formato.

Las ventajas no sólo están asociadas a la minimización de los riesgos indicados sino también en virtud del tiempo que insumen su realización en forma manual versus su automatización.

17.3. Identificar aquellos datos de mayor utilidad en la exposición de la información

La información contable principal está contenida en los saldos y movimientos de las cuentas contables.

A modo de ejemplo el reporte del Balance de Sumas y Saldos que se puede obtener del sistema permite analizar preliminarmente una estado de resultados según el período analizado: mes, año, etc.

17.4. Relacionar los reportes del sistema que sirven de base de datos para la confección de otros reportes a utilizar en sistemas externos

El balance de sumas y saldos, puede ser analizado trimestralmente para analizar comparativamente los resultados de cada mes, para lo cual es posible generar una interacción entre tres archivos tipo CSV con un único informe gerencial para ser presentados ante del Directorio de la empresa.

También es posible unificar en un único informe gerencial la información proveniente del balance de sumas y saldos de distintas unidades de negocios que consolidan su información para ser puesta a consideración de otros actores partícipes en la toma de deicisiones.

17.5. Compatibilizar archivos propios de origen con archivos externos de otras entidades como la Administración Federal de impuestos, bancos, etc. mediante el uso del lenguaje de programación Python

17.6. Redacción de problemas y casos a los fines de presentarlos en las instancias educativas

Esta acción se describe a partir de la página 122 donde se exponen algunos de los casos prácticos resueltos que se ofrecen como material de estudio basado en problemas reales.

Etapa 3: Prueba Piloto de Instancia educativa

18. Selección de estudiantes y recientes graduados voluntarios para aprender PYTHON aplicado a las ciencias económicas

Los trabajos de campo como entrevistas, encuestas y asistencia a los seis encuentros de taller propuestos contaron con el compromiso de todos y cada uno de los integrantes del equipo de investigación seleccionados a partir de invitaciones personales.

Se pudo comprobar la existencia de un genuino interés por sus aportes concretos, el alto grado de compromiso individual y colectivo como así también la dedicación para el cumplimiento de los objetivos previstos.

Todos los participantes estudiantes, graduados y también docentes y no docentes que solicitaron formar parte de la propuesta de prueba piloto que participaron de las jornadas de capacitación, dieron muestra de un interés propio por la búsqueda de la excelencia académica en el caso de los estudiantes y por cumplir con los objetivos de la capacitación continua en el caso de los colegas graduados.

19. Definición y descripción de: contenidos disciplinares, objetivos; cronograma; método de trabajo y evaluación de los aprendizajes de la instancia educativa

A efectos de no redundar aquí, se anticipa que los logros resultantes de esta tarea se encuentran detallados en la propuesta académica propiamente dicha que se expone mas adelante en página 115.

20. Evaluación de la prueba piloto del Taller. Contendrá opiniones de los asistentes respecto a la metodología aplicada. Análisis de logros

20.1. Devolución de integrantes del PROMCE

Durante las primeras clases, se recabaron por parte de los asistentes a las clases de la instancia piloto las apreciaciones personales respecto al contenido de los talleres.

Se puede resumir, que existió una consideración unánime acerca de lo innovador de la propuesta y de la aplicabilidad práctica en el quehacer profesional de los contenidos brindados.

Se les solicitó a quienes quisieran que la calificaran en una escala de uno a cinco. La valoración general fue considera muy positiva, y se resumen algunas de las opiniones recibidas por parte de los alumnos:

- Karen, estudiante, 26 años.

> Hola! Mi comentario es el siguiente: La primera clase de este experimento me pareció muy interesante en cuanto al desarrollo de contenidos abocados principalmente a ejemplos prácticos sin dejar de lado los conceptos teóricos como la sintaxis de la programación. Particularmente los códigos desarrollados en la clase relacionados al tiempo resultan útiles para trabajar con informes periódicos y también para filtrar datos según fechas o incluso horas. Considero muy valiosa la retroalimentación que se dio dada las preguntas y cuestionamientos por parte de todos los participantes con distinto nivel de conocimiento respecto al tema. Puntualmente califico la clase con 5 estrellas. Espero que les sirva Saludos!

- Pablo, docente, 45 años.

> El primer encuentro/ clase me pareció muy bien organizada y enfocada en optimizar los tiempos para la búsqueda de objetivos. Entretenida y prácticamente en todo momento percibir que lo que se está exponiendo tiene en el fondo una potencia en términos de aplicabilidad. En cuanto a mi expectativa profesional/ laboral me concentro en el potencial de automatización de tareas que posibilita Python, más aún extendiendo esa expectativa a que la automatización integre varias aplicaciones que ya uso. Creo que aún no he podido ver en toda su dimensión el espectro de posibilidades que permite Python.

Como miembro de una generación X que se vinculó con la informática de una forma co convencional (teoría y práctica coordinadas pero separadas) me cuestan un poco los modos actuales de aprendizaje donde teoría y práctica tiene bordes difusos y todo parece una suerte de aprendizaje por experimentación casi intuitiva. Eso genera cierta ansiedad porque la formación convencional con el método "índice- libro"te permitía saber qué contenidos eran .el todoz en qué tema o punto uno estaba parado. Pero entiendo que esta nueva modalidad es la regla porque simplemente demostró ser superadora a la anterior.
Por eso voy a necesitar cierta ayuda para saber el cómo hacer las cosas solo.
No se donde calificar la actividad con estrellas. No tengo IG y no uso el Facebook sino Twitter. Pero lo pondría 5 estrellas tranquilamente.

- Andrés, profesional en ciencias económicas, 33 años.

Hola, me parece una idea sólida y por el momento bien ejecutada. Sumaría alguna breve explicación focalizando la sintaxis de Python sobre todo para cuando haya alumnos que lo tomen de 0 sepan cómo se compone un código, los distintos caracteres, etc. Entiendo que la idea de era ir explicándolo sobre el ejemplo, pero unos minutos previos con este desarrollo vienen bien para parar en la cancha al que se sume.
Como les comenté estaría bueno armar o ver algún código que te permita acceder a sitios que requieran credenciales o tengan algún nivel de seguridad, por ejemplo AFIP (cuit+clave), sitios de operaciones web, web de tarjetas de crédito, etc.. lugares que se requiera ingresar para descargar algún reporte o información para la diaria en una empresa, estudio o institución pública.
Mi calificación a esta clase: 4
Saludos

- Juan, Ingeniero Industrial, 28 años.

Para mi 10 puntos la primer clase (5 estrellas)
Lástima que no llegamos con el tiempo para hacer la actividad práctica, pero por otro lado se dejó planteado el problema y estaría bueno si la semana que viene con las herramientas que dimos en la clase cada uno pudo resolverlo.
Está bueno si cada clase se plantea algún desafío o tarea a resolver
Por otro lado, ejemplo práctico para el uso de la librería time que vimos en esta clase: Obtener información sobre las facturas vencidas en cuentas corrientes de una Empresa, que te calcule cuántos días lleva vencida y que con una tasa de interés que tengas definida calcule los costos de dichos intereses. Alertas de vencimientos de pagos.
Nos estamos viendo el viernes!
Saludos!

- Walter, ingeniero en sistemas, 50 años.

Buen día, les brindo brevemente mis comentarios sobre el Curso de Automatización con Python que vienen desarrollando en la FCE. Me parece una excelente idea la de abordar esta temática, ya que adquirir estos conocimientos dan un plus muy grande en el desempeño profesional. En particular, lo visto en estas primeras clases, relacionado a distintos tratamientos de grandes archivos de formato Microsoft Excel®me pareció muy interesante y valioso.
Desde ya, agradecido por la posibilidad de tomar la capacitación. Saludos!

21. Informe de evaluación que contenga ajustes y mejoras

Como complemento de la propuesta educativa formal, es posible sintetizar que se ha corroborado la vinculación sinérgica entre la práctica profesional y Python como lenguaje de programación que confirman algunos presupuestos analizados al comienzo de este proyecto educativo.

Los resultados obtenidos han sido altamente satisfactorios comprobándose la existencia de un amplio campo de aplicaciones como la automatización de tareas rutinarias y el análisis de datos avanzado.

Para aprovechar al máximo esta sinergia es imprescindible promover la capacitación, des-arrollar soluciones específicas y fomentar la colaboración entre los profesionales contables y los expertos en programación, tanto en la formación de grado como de posgrado.

Al mismo tiempo, y como se ha planteado, esto es sólo el punto de partida para abordar nuevos desafíos en relación a los avances tecnológicos sobre los cuáles no se conoce con exactitud el formato o impacto que tendrán pero que indudablemente seguirán reconfigurando la tarea de todas las actividades profesionales en el corto plazo.

Es posible dejar aquí planteadas algunas áreas de interés para investigaciones futuras: Desarrollo de herramientas específicas para áreas contables especializadas: como la auditoría, la gestión de riesgos y el análisis financiero. Integración de Python con otros sistemas contables ampliamente utilizados en Argentina que faciliten la interoperabilidad y el intercambio de datos entre diferentes plataformas, mejorando la eficiencia y la calidad de la información contable.

Desafíos profesionales

Python: una oportunidad de potenciar la labor en Ciencias Económicas

La aplicación de lenguajes de programación que fortalezcan la práctica del profesional en Ciencias Económicas está en una etapa embrionaria pero en crecimiento. Desafíos a futuro.

repetitivas mediante la automatización de las mismas para emplearlo en tareas y actividades de mayor valor.

¿POR QUÉ PYTHON?

Python es un lenguaje de programación que ha sido desarrollado en la década de los 80 por el matemático holandés Guido Van Rossum pero que ha ido ganando mayor popularidad en los últimos años, siendo de código abierto y gratuito. Tiene como virtud una estructura similar a cómo piensa el ser humano y cuenta con una sintaxis accesible para las personas con un nivel de 'alfabetización' básico en lenguajes de programación.

Se puede destacar que Python:

- Es el lenguaje de programación más popular del mundo: ha sido reconocido como el más accesible, fácil y simple de aprender.
- Tiene una sintaxis muy 'intuitiva que ayuda a los que están empezando (siempre que

do con un espíritu colaborativo que se puede comprobar en numerosos foros de intercambio y/o de desafíos por proyectos.

Al Sweigart, en su libro 'Automate the boring stuff with Python' refiere que un usuario capacitado puede aprovechar la potencia de las computadoras para lograr en segundos realizar tareas que tardarían decenas de horas humanas, con el sólo hecho de programar algunas pocas líneas de código.

PYTHON Y EXCEL

La aplicación de lenguajes de programación que fortalezcan la práctica del profesional en Ciencias Económicas está en una etapa embrionaria, pero en crecimiento en diversos formatos, tanto en bibliografía tradicional como en cursos completos en videos pre-grabados.

En la formación del profesional de Ciencias Económicas se ha incorporado desde hace más de 15 años la enseñanza de las planillas de cálculo, siendo Microsoft Excel una de las más utilizadas. Un archivo Excel agrupa una o más hojas de cálculos, en las cuales se puede manejar una cierta cantidad de datos almacenados en celdas identificadas mediante filas y columnas, sobre las que se puede operar estableciendo referencias desde simples operaciones matemáticas a algoritmos más complejos.

Ser usuario de una planilla de cálculo pareciera ser el techo del conocimiento en el manejo de datos de nuestros egresados de la Facultad, pero cada vez más se encontrará en la necesidad de aumentar su productividad, manejar información de diferentes orígenes, formatos y en grandes volúmenes, cuestión que le resultará más difícil con un software 'enlatado' como el Excel.

Poder implementar por sí mismo una solución 'a medida' que le permita resolver la necesidad de su puesto de trabajo combinando el potencial de Python con la familiaridad del Excel, es lo que vislumbramos como camino a seguir por el profesional.

Figura 27: Publicación en Periódico El Paraninfo Nº 179.

Parte IV

Propuesta Educativa

Asignatura optativa para las carreras de grado de la UNL-FCE

22. Contenido de la Propuesta

.

En cumplimiento del tercer objetivo específico, se tiene en cuenta lo aprobado por la Res. CS Nº 502/18, en cuanto a que las asignaturas optativas «contemplarán propuestas que impliquen trayectos formativos orientados hacia los diversos ámbitos laborales en el que el Contador Público desempeñe sus funciones ... en empresas locales, nacionales e internacionales, en la administración pública y como profesional independiente», en el marco de una carga horaria total de 270 horas para toda la carrera.

A continuación se desarrollan los ítems que conforman la propuesta que será puesta a consideración ante el Consejo Directivo de la Facultad de Ciencias Económicas de la Universidad Nacional del Litoral. La misma sigue la estructura de una presentación de típica.

22.1. Denominación de la Asignatura

Python para las Ciencias Económicas

22.2. Inserción en la carreras

Se consideran destinatarios de esta propuesta los estudiantes de las carreras de Contador Público y Licenciatura en Administración que hayan aprobado la totalidad de las asignaturas correspondientes al Bachiller Universitario en Ciencias Económicas.

Se los considera un público propenso a absorber nuevos conocimientos y habilidades que los preparen de mejor manera para la práctica profesional. Estas nuevas herramientas les permitirán, en sus postulaciones a pasantías o puestos laborales iniciales, aplicar a tareas de mayor jerarquía y responsabilidad.

22.3. Sistema de evaluación, condiciones de regularidad y promoción

Para poder promocionar la asignatura, el alumno deberá:

- Registrar una asistencia mínima de 80 % del total de las clases dictadas, y
- Presentar y aprobar un trabajo final integrador en grupo de a dos alumnos el que será analizado y compartido en una clase especial ante docentes y estudiantes.

Los trabajos finales serán evaluados por la pertinencia en la formulación del caso, las fuentes de información utilizadas, los métodos de búsqueda y selección de los recursos empleados y las herramientas propuestas para la resolución del mismo.

22.4. Carga horaria

La carga horaria prevista total es de 60 horas, las cuales se desagregan en 20 horas destinadas a clases presenciales, 10 horas clases o consultas virtuales y las restantes 30 horas se considera el tiempo estimado para la realización de actividades prácticas fuera del aula y la realización del trabajo práctico final.

22.5. Objetivos de la asignatura

Se persigue como objetivo general brindar a los alumnos nuevos conocimientos, habilidades y competencias derivadas de la iniciación en programación mediante el uso de Python.

Para ello se seguirán la siguientes líneas de acción estratégicas:

- Introducir a los estudiantes en el conocimiento de un lenguaje de programación que optimice eficazmente las tareas profesionales en las que se va a desempeñar.
- Transmitirles las ventajas de utilizar el lenguaje de programación PYTHON en los procesos, procedimientos e instrucciones de análisis de datos en tareas profesionales de las Ciencias Económicas de ámbito público o privado de manera tal que se logre eficiencia y confiabilidad de la información administrativa y/o contable
- Presentarles la naturaleza e importancia actual del manejo de grandes volúmenes de datos y el impacto de la Inteligencia Artificial en la administración de las organizaciones.
- Permitirles desarrollar habilidades para interactuar con profesionales de otras ramas (Ingenieros de datos, ingenieros en sistemas, entre otros) para poder desarrollar herramientas de gestión eficientes, rápidas y seguras para la toma de decisiones confiables. .

22.6. Régimen de cursado

Optativa dentro del plan de estudios de las carreras de Contador Público y Licenciatura en Administración, de duración cuatrimestral.

22.7. Modalidad de cursado

Presencial y semipresencial con videoclases sincrónicas, y clases de tutoría o consultas por videollamadas en forma grupal.

22.8. Propuesta metodológica

El programa se desarrollará conforme un cronograma de 8 clases explicativas basadas en casos prácticos, y clases de consultas según las necesidades de los estudiantes.

Se desarrollarán actividades formativas desde una metodología activa y participativa de modo que, a través de la conjunción de lo deductivo e inductivo, el alumno alcance el máximo grado de capacidad, conocimiento y seguridad, en lo que se refiere a los objetivos de la asignatura.

La propuesta está basada en el desarrollo del pensamiento complejo.

Entendiendo por complejo, del latín complexus: lo que está tejido en conjunto. Constituyentes heterogéneos inseparablemente asociados, presenta lo uno y lo múltiple, tejido de acciones e interacciones. En consecuencia, el desarrollo del pensamiento complejo demanda:

- interrelación entre estudiantes y docentes en continuo intercambio de ideas, opiniones y mutuas críticas;
- análisis de casos particulares para obtener síntesis globales. Es decir, de los reiterados casos particulares de la actividad profesional contable con fuerte carga rutinaria se obtiene el algoritmo que representa una síntesis global;
- dialógicos entre pares y con expertos de manera tal que se pongan a prueba los hallazgos, el análisis y la síntesis obtenidos;
- el uso de razonamientos inductivos y deductivos aplicados para dar respuestas a situaciones del quehacer profesional de las ciencias económicas.

La cátedra sugerirá a los estudiantes la previa lectura de los temas de las clases a abordar con anterioridad al dictado a los efectos de posibilitar el proceso de transmisión de conocimientos.

Esto se logrará dentro de un marco de óptima y favorable interrelación entre profesores y alumnos donde es clave la metodología del Aprendizaje Basado en Problemas, y el rol del docente como guía.

Para los alumnos representa una novedad dentro de la FCE-UNL ya que no es una práctica habitual, pero a la vez se pretende lograr un ida y vuelta con casos reales que los propios estudiantes traigan desde sus propios entornos laborales o personales al momento de presentar el trabajo práctico integrador.

Las clases se dictarán en aulas informatizadas donde se encontrarán instaladas diferentes aplicaciones y programas destinados al uso en clases y se promoverá que los estudiantes hagan lo propio en sus hogares o lugares de trabajo.

A partir de ciertos conceptos básicos e introductorios, las clases girarán en torno a cinco casos prácticos que servirán como disparadores para que el alumno descubra y comprenda

el potencial que les brinda la aplicación de un lenguaje de programación en relación a tareas rutinarias, con manejo de grandes volúmenes de datos o que impliquen posibles errores de carga manual.

Desde este enfoque es que se pretende que surjan las ideas en los alumnos para diseñar y ejecutar el trabajo final, aplicables a situaciones reales que puedan repercutir en su entorno laboral o de estudios mas cercano.

22.9. Estrategias de Enseñanza

El aprendizaje para el desarrollo de los contenidos de la asignatura, se llevarán adelante en base a la siguiente planificación:

a) Contenidos teóricos - Descripción: Presentación en el aula de los conceptos propios de la asignatura haciendo uso de una metodología expositiva concisa y aplicada buscando acciones participativas a partir de exposiciones en soporte audiovisuales: pantallas y videos.

- Propósito: Transmitir los contenidos de la materia motivando al alumnado a la búsqueda de posibles soluciones, facilitándole el descubrimiento de las relaciones entre diversos conceptos y formarle una mentalidad crítica y orientada a la resolución de la situación problemática.

b) Actividades Prácticas - Descripción: Actividades a través de las cuales se pretende mostrar al estudiante cómo debe actuar a partir de la aplicación de los conocimientos adquiridos mediante el método de Aprendizaje Basado en Problemas con la provisión de una guía práctica elaborada por la cátedra y presentada como material bibliográfico.

- Propósito: Desarrollo en el estudiante de las habilidades instrumentales de la materia. Se busca que los alumnos comprendan y resuelvan problemas, en particular razonando para encontrar nuevas técnicas cuando fuera necesario, o basándose en la reutilización de técnicas y procedimientos ya utilizados.

c) Actividades no presenciales - Descripción: Actividades –guiadas y no guiadas– propuestas por el cuerpo docente a través de las cuales se profundiza en aspectos concretos de la materia posibilitando a los estudiantes avanzar en la adquisición de determinados conocimientos, metodologías, procedimientos, técnicas, y permitiéndoles expresar sus dudas y hallazgos.

- Propósito: Favorecer en los estudiantes la generación e intercambio de ideas, la identificación y análisis de diferentes puntos de vista sobre un problema, la generalización o transferencia de conocimiento y la valoración crítica del mismo. A través de estas actividades se busca desarrollar en los alumnos valores de investigación, creatividad, búsqueda e innovación.

22.10. Programa analítico.

Se encuentra estructurado en cinco casos prácticos puntuales que se pondrán en consideración en ocho clases totales, de 150 minutos de duración cada una. Con respecto al tercer objetivo del proyecto "Diseñar un plan de capacitación dirigido a Contadores Públicos y estudiantes de ciencias económicas para las etapas iniciales en la formación de Python como lenguaje soporte de las tareas de la práctica profesional", ha quedado incorporado a los ambientes virtuales de la FCE-UNL el esquema de módulos que podrían ser transferidos a alumnos de la comunidad educativa.

Presentación e Introducción: Clase 1.

Conocimiento de entornos para desarrollo de programación. Jupyter Notebooks, Google Colab.

Características principales de Python.

Fuentes de información y consultas. Stack Overflow, Kaggle, Chat GPT.

Importar librerías. Incorporar módulos adicionales de librerías.

Asignar valores a variables.

Ejercicio práctico N°1: Clase 2.

Lectura de PDF.

Importar librería de manejo de sistema operativo.

Expresiones regulares.

Bucle FOR.

Exportar en formato Excel.

Ejercicio práctico N°2: Clase 3.

Librería Pandas.

La sistematización en el guardado de archivos.

Manipulación de Datos y Limpieza de datos.

Combinar archivos de diferentes orígenes.

El uso de variables.

Exportar en formato TXT.

Ejercicio práctico N°3: Clase 4.

Los dataframes.

Creación de nuevos registros.

Cálculos matemáticos en base a los registros.

Filtrado de datos

Ordenamiento de datos.

Integración de datos.

Ejercicio práctico N°4: Clases 5 y 6.

Librerías gráficas. Matplotlib, Plotly, Seaborn.

Agrupamiento de registros.

Representación gráfica de datos.

Ejercicio práctico N°5: Clases 7 y 8.

Automatizando el envío de correos electrónicos en bloque.

Reiteración del bucle FOR.

Ciclo IF.

Adjuntar archivos diferentes por cada destinatario.

22.11. Cronograma de clases.

Contenidos	Clases							
	1	2	3	4	5	6	7	8
Introducción	■							
Ejercicio N° 1		■						
Ejercicio N° 2			■					
Ejercicio N° 3				■				
Ejercicio N° 4					■	■		
Ejercicio N° 5							■	■

22.12. Bibliografía básica.

- BOTA, H. Y GOSA A. (2021) Python for accounting. A modern guide to using programming in Accounting. E-Book. Estados Unidos.
- DRISCOLL, M. (2020) Python 101. Leanpub book.
- GONZALEZ DUQUE, R. (2015) Python para todos. Free E-Book.
- SWEIGART, A. (2015) Automate the boring stuff with Python. No Starch Press. San Francisco.
- ZUMSTEIN, F. (2021) Python for Excel. A modren enviroment for automation anda data analysis. O'Reilly.
- Material elaborado por la cátedra.

23. Trabajos Prácticos de elaboración propia

Retomando la acción propuesta como "I) Redacción de problemas y casos a los fines de presentarlos en las instancias educativas" se exponen seguidamente cinco casos concretos de ejemplos de Aprendizaje Basado en Problemas.

Este compendio de casos expuestos como ejercicios prácticos son incorporados como material de estudio dentro de la bibliografía básica que tiene el programa de la asignatura Python para las Ciencias Económicas.

Se propone como método expositivo ir mostrando las líneas de código por tramos, los que se visualizan e identifican con un número, por ejemplo: "**In[1]:**", y a su vez cada línea de código independiente también tiene un número identificador del renglón por cada entrada.

A continuación, y en los casos que sea necesario se expone la salida o "**Out[1]:**" relativa a la entrada brindada.

Es decir que el lector podrá seguir en forma secuenciada, el código, su explicación y el resultado obtenido.

Estos resultados son expuestos, como salidas de código que brinda el intérprete de Python, que en este caso se han generado dentro de Jupyter Notebook. En otros casos, para mayor detalle se muestran capturas de pantalla.

23.1. Ejercicio Práctico N° 1: Renombrar archivos PDF según su contenido

Presentación del problema: Asuma que parte de sus tareas diarias consiste en la realización de una gran cantidad de pagos vía transferencias bancarias que realiza a través de la web del banco, cuyos comprobantes, son generados por el banco en forma automática en un archivo con formato de extensión PDF de idénticas características al siguiente.

La descarga de los archivos que acreditan las transferencias son guardados en su ordenador con un nombre "Ticket.pdf" y en caso de ya existir un archivo anterior con dicho nombre en el directorio seleccionado el nombre del mismo es cambiado automáticamente anexándole un número correlativo superior, por ejemplo: "Ticket(1).pdf".

Analice esta situación a los efectos de proponer mediante un proceso automatizado que se renombren cada uno de los archivos con datos de utilidad para su posterior uso e identificación rápida.

Figura 28: Modelo de comprobante descargable del NBSF

El presente problema se comienza a resolver importando librerías que permiten la O incluir ciertas líneas, donde se indican que librerías se importarán:

```
In [1]: import os
        import re
        import PyPDF2
```

estas importaciones son herramientas poderosas que permiten a los estudiantes de ciencias económicas manipular y analizar datos de manera efectiva. Desde la manipulación de archivos y la búsqueda de patrones en texto hasta el procesamiento de documentos PDF, estas herramientas son fundamentales para la investigación, el análisis y la toma de decisiones informadas en el ámbito económico.

import os: El módulo os proporciona funciones para interactuar con el sistema operativo. En el contexto de ciencias económicas, este módulo podría ser útil para manipular archivos de datos, crear y eliminar directorios, acceder a variables de entorno del sistema, entre otras tareas. Por ejemplo, podría ser utilizado para automatizar la gestión de archivos de datos financieros o económicos.

import re: El módulo re permite trabajar con expresiones regulares en Python. Las

expresiones regulares son patrones de búsqueda utilizados para encontrar y manipular cadenas de texto. En el ámbito de las ciencias económicas, esto podría ser útil para analizar y extraer información de documentos financieros, informes económicos, o para procesar grandes cantidades de datos textuales relacionados con tendencias económicas o financieras.

import PyPDF2: PyPDF2 es una biblioteca de Python que permite trabajar con archivos PDF. Con esta biblioteca, es posible extraer texto, fusionar, dividir y modificar archivos PDF. En el ámbito de las ciencias económicas, donde los informes, documentos financieros y análisis son comunes en formato PDF, esta biblioteca puede ser útil para extraer información clave de dichos documentos, realizar análisis de texto o recopilar datos para análisis económicos

Y a continuación, se asigna con el nombre de una variable las carpetas con los archivos originales que deseamos abrir y analizar, y la ruta de la carpeta destino de los archivos ya renombrados.

In [2]:
```
directorio\_originales = 'C:/Tickets originales/'
directorio\_renombrados = 'C:/Tickets renombrados/'
```

Estas dos líneas de código están asignando rutas de directorio a dos variables: directorio_originales y directorio_renombrados. Aquí está la explicación orientada a estudiantes de ciencias económicas: En la línea 1: directorio_originales: Esta variable almacena la ruta del directorio donde se encuentran ubicados los "Tickets originales". En el contexto de ciencias económicas, estos "tickets" podrían referirse a facturas, recibos, o cualquier otro tipo de documento relacionado con transacciones financieras o de negocios. La ruta del directorio indicada, C:\Ticketsoriginales, sugiere que los archivos originales se encuentran en una ubicación específica en el disco duro del sistema, en la unidad C, dentro de una carpeta llamada "Tickets originales".

En la línea 2: directorio_renombrados: Similarmente, esta variable almacena la ruta del directorio donde se guardarán los "Tickets renombrados" después de algún procesamiento. En el contexto económico, el proceso de renombrar archivos podría implicar la organización de documentos, la adición de metadatos, o la estandarización de nombres para facilitar su gestión y análisis. La ruta indicada, 'C:\Tickets renombrados', sugiere que los archivos renombrados se guardarán en una ubicación específica en la unidad C, dentro de una carpeta llamada "Tickets renombrados".

Para resumir, estas líneas de código son importantes para definir las ubicaciones de los archivos originales y procesados, lo que facilita la manipulación y gestión de datos financieros o comerciales en proyectos relacionados con ciencias económicas.

Se trata de encontrar visualmente cuál es el patrón con el cual están estructurados los archivos y la extensión de caracteres que tiene cada campo de datos.

In [3]:
```
patron_ticket = re.compile(r'(FECHA Y HORA)(\d{2})(/)(\d{2})(/)(\d{4})(.\*)(
    PAGO DE )(.\*)(NRO. DE CLIENTE: )(.\*)(CON DEBITO EN: )(.\*)(IMPORTE: \$
```

```
)(.*)(FECHA VENCIMIENTO: )(.\*)(CUOTA: )(.\*)(ESTE RECIBO ES CONSTANCIA
DE PAGOPAGADO)',re.DOTALL)
```

Esta línea de código utiliza la biblioteca “re” de Python para compilar una expresión regular que busca un patrón específico en un texto. Aquí está la explicación orientada a estudiantes de ciencias económicas:

En la línea 1: patron_ticket = re.compile(...): La función “re.compile()” compila una expresión regular en un objeto de patrón de expresión regular. Este objeto se puede utilizar para buscar o hacer coincidir patrones dentro de cadenas de texto de manera más eficiente que si se utilizara la función ‘re.match()‘ o “re.search()” directamente.

La letra ‘r‘ antes de la cadena de texto indica una cadena de texto en crudo (raw string), lo que significa que los caracteres especiales dentro de la cadena de texto no se interpretan como escapados. Esto es útil cuando se trabajan con expresiones regulares, ya que evita la necesidad de escapar caracteres especiales con barras invertidas.

En la línea 3: Expresión regular: La expresión regular definida busca un patrón específico en un texto que representa un “Ticket”. Aquí está la desglosado del patrón:

- (FECHA Y HORA): Busca la cadena “FECHA Y HORA” literalmente.
- (2): Busca dos dígitos seguidos (que representan la fecha).
- (/): Busca un caracter de barra (“/”).
- (2): Busca dos dígitos seguidos (para el mes).
- (/): Busca otro caracter de barra (“/”).
- (4): Busca cuatro dígitos seguidos (para el año).
- (.*): Busca cualquier otro caracter (información adicional).
- (PAGO DE): Busca la cadena “PAGO DE” literalmente.
- (.*): Busca cualquier otro caracter (información sobre el pago).
- (NRO. DE CLIENTE:): Busca la cadena “NRO. DE CLIENTE: ” literalmente.
- (.*): Busca cualquier otro caracter (el número de cliente).

- (CON DEBITO EN:): Busca la cadena “CON DEBITO EN: ” literalmente.
- (.*): Busca cualquier otro caracter (la ubicación del débito).
- (IMPORTE: $): Busca la cadena “IMPORTE: $ ” literalmente.
- (.*): Busca cualquier otro caracter (el monto del importe).
- (FECHA VENCIMIENTO:): Busca la cadena “FECHA VENCIMIENTO: ” literalmente.
- (.*): Busca cualquier otro caracter (la fecha de vencimiento).
- (CUOTA:): Busca la cadena “CUOTA: ” literalmente.
- (.*): Busca cualquier otro caracter (información sobre la cuota).
- (ESTE RECIBO ES CONSTANCIA DE PAGO): Busca la cadena “ESTE RECIBO ES CONSTANCIA DE PAGO” literalmente.

En la línea 4: re.DOTALL: Esta bandera le indica a la expresión regular que el meta-caracter ‘.‘ también coincida con el caracter de nueva línea (‘n‘). Esto permite que el patrón abarque múltiples líneas de texto.

Resumiendo, esta expresión regular busca un patrón específico dentro de un texto que representa un “Ticket” o recibo de pago, y extrae información importante como la fecha, el monto del pago, el número de cliente, entre otros, para su posterior procesamiento en el contexto de las ciencias económicas.

Es preciso poder asignar un patrón único para todos y cada uno de los archivos (Tickets) los cuáles contienen datos que son comunes y otros que sólo le corresponden a cada ticket en particular.

Si se observa la imagen de página 122 se podrá advertir que todos los tickets contendrán “FECHA Y HORA”, pero cada uno tendrá la propia en la que se generó con una estructura unívoca que consta de dos dígitos numéricos para el día seguido por una barra inclinada hacia la derecha, dos dígitos para el mes seguido de otra barra inclinada hacia la derecha para finalmente terminar con cuatro dígitos para indicar el año en que se generó el ticket.

La utilización de la librería “re” de Expresiones Regulares va a permitir prefijar estas estructuras predeterminadas, para luego poder extraer los datos de utilidad en forma separada.

Colocando cada expresión regular entre paréntesis será posible ya distinguir entre que

el texto “FECHA Y HORA” no aporta información precisa para el objeto de este problema pero si los datos consignados seguidamente como el día, mes u hora de generación.

El presente caso es sencillo de explicar ya que se trata de separar el texto fijo del variable. El ejemplo comienza con el texto fijo “FECHA Y HORA” como el primer grupo, a continuación se genera el segundo grupo que está compuesto por un valor variable de dos dígitos numéricos que expresan el día en que se hizo la operación.

El tercer grupo es de tipo fijo y lo representa la barra inclinada a la derecha que separa el día de del mes de la operación, lo que se repite a continuación para el año que sería el sexto grupo de caracter variable predefinido con cuatro dígitos numéricos.

A continuación del año se encuentran datos, fijos y variables, que no son de utilidad para el caso por lo tanto se decide agruparlos como séptimo grupo con la expresión regular (.*) que se extenderá hasta que encuentra el texto fijo (PAGO DE) que se establece como el octavo grupo.

La expresión regular encontrada para el séptimo grupo es: 11:23:27OPERACIÓNAgenda de pagosNÚMERO DE OPERACIÓN3d0a8106c83047cf8e5cd3b7d1b4ff02.

El script se continúa con un ciclo FOR:

In [4]:
```
for nombre_archivo in os.listdir(directorio_originales):

    if nombre_archivo.endswith('.pdf'):

        with open(os.path.join(directorio_originales, nombre_archivo), 'rb')
    as archivo_pdf:
            lector_pdf = PyPDF2.PdfFileReader(archivo_pdf)

            texto = ''
            for pagina in range(lector_pdf.getNumPages()):
                texto += lector_pdf.getPage(pagina).extractText()
                archivo_pdf.close()


            datos_encontrados = patron_ticket.search(texto)

            if datos_encontrados:

                datos = datos_encontrados.group(9)+datos_encontrados.group
   (6)+datos_encontrados.group(4)+datos_encontrados.group(2)+'$'+
   datos_encontrados.group(15)

                nuevo_nombre_archivo = datos + '.pdf'

                try:
```

```
23                      os.rename(os.path.join(directorio_originales,
    nombre_archivo), os.path.join(directorio_renombrados,
    nuevo_nombre_archivo))
24                      print(f'El archivo {nombre_archivo} fue renombrado como
    {nuevo_nombre_archivo}')
25                  except OSError as e:
26                      print(f'Error al renombrar {nombre_archivo}: {e}')
27              else:
28                      print(f'No se encontraron coincidencias de formato en {
    nombre_archivo}')
```

Este bloque de código realiza una serie de acciones en relación con archivos PDF en el contexto de ciencias económicas, brindando una explicación orientada a estudiantes no especializados en programación.

En la línea 1: Iteración sobre archivos en el directorio original:

for nombre_archivo in os.listdir(directorio_originales):

Esta línea inicia un bucle que itera sobre los archivos en el directorio original (‘directorio_originales‘).

‘os.listdir()‘ devuelve una lista de nombres de archivos en el directorio especificado.

En la línea 2: Filtrado de archivos PDF:

if nombre_archivo.endswith(‘.pdf’)

Se verifica si el nombre del archivo actual termina con ‘.pdf’. Esto filtra solo los archivos PDF en el directorio.

En la línea 3: Se realiza la lectura del contenido del PDF:

with open(os.path.join(directorio_originales, nombre_archivo), ‘rb’) as archivo_pdf:

lector_pdf = PyPDF2.PdfFileReader(archivo_pdf) texto = ‘’

for pagina in range(lector_pdf.getNumPages()): texto += lector_pdf.getPage(pagina).extractText() archivo_pdf.close()

Se utiliza “PyPDF2” para leer el contenido del archivo PDF página por página y concatenar el texto extraído. El archivo se cierra después de la lectura.

En la línea 4: Búsqueda de patrón en el texto del PDF:

datos_encontrados = patron_ticket.search(texto)

Se utiliza el patrón definido anteriormente ("patron_ticket") para buscar coincidencias en el texto del PDF. El patrón está diseñado para extraer información específica relacionada con pagos y fechas.

En la línea 5: Renombrar el archivo según los datos encontrados:

Si se encuentran datos según el patrón, se extraen y se utiliza esta información para construir un nuevo nombre para el archivo PDF. Luego, se intenta renombrar el archivo original con el nuevo nombre en el directorio de archivos renombrados ('directorio_renombrados'). Si ocurre algún error, se imprime un mensaje de error.

Sintetizando, este código realiza la búsqueda y procesamiento de archivos PDF en un directorio específico, extrayendo información relevante y renombrando los archivos de acuerdo con ciertos criterios definidos por el patrón definido. Está diseñado para trabajar con recibos o documentos relacionados con pagos y fechas en el contexto de ciencias económicas.

En este punto la variable texto creada en blanco, ya habrá tomado el valor correspondiente a la lectura de todas las expresiones regulares indicadas al crear el patrón.

Si se hiciera print(texto): luego de la línea 11 se obtendría el siguiente resultado:

FECHA Y HORA05/05/2023 11:23:27OPERACIÓNAgenda de pagosNÚMERO DE OPERACIÓN3d0a8106c83047cf8e5cd3b7d1b4ff02PAGO DE ADMPROVIMPUESTOSSANTAFENRO. DE CLIENTE: 00241430225164040CON DEBITO EN: CAJA DE AHORRO EN PESOSNRO DE CUENTA: 500098066908IMPORTE:
$ 5448,55FECHA VENCIMIENTO: 18\05\23CUOTA: 002\23ESTE RECIBO ES CONSTANCIA DE PAGOPAGADO

Es decir que para esa altura del proceso la variable texto para el primero de los archivos analizados se encuentra definida por una cadena de caracteres continuos separados convenientemente por el uso de las expresiones regulares detectadas en todos los archivos PDF que fueron analizados.

In [5]:
```
import pandas as pd

patron_ticket = re.compile(r'(FECHA Y HORA)(\d{2}/\d{2}/\d{4})(.*)(PAGO DE )
    (.*)(NRO. DE CLIENTE: )(.*)(CON DEBITO EN: )(.*)(IMPORTE: \$ )(.*)(FECHA
    VENCIMIENTO: )(.*)(CUOTA: )(.*)(ESTE RECIBO ES CONSTANCIA DE PAGOPAGADO)'
    , re.DOTALL)
```

```
4
5  def extraer_datos(texto):
6      match = patron_ticket.search(texto)
7      if match:
8          return match.groups()
9      else:
10         return None
11
12 def pdf_a_datos(pdf_path):
13     with open(pdf_path, 'rb') as archivo_pdf:
14         pdf_reader = PyPDF2.PdfFileReader(archivo_pdf)
15
16         datos_extraidos = []
17         for pagina_num in range(pdf_reader.numPages):
18             pagina = pdf_reader.getPage(pagina_num)
19             texto_pagina = pagina.extractText()
20
21             datos = extraer_datos(texto_pagina)
22             if datos:
23                 datos_extraidos.append(datos)
24
25     return datos_extraidos
26
27 archivos_pdf = [archivo for archivo in os.listdir(directorio_renombrados) if
       archivo.endswith('.pdf')]
28
29 datos_totales = []
30
31 for archivo_pdf in archivos_pdf:
32     ruta_pdf = os.path.join(directorio_renombrados, archivo_pdf)
33     datos_pdf = pdf_a_datos(ruta_pdf)
34     datos_totales.extend(datos_pdf)
35
36 columnas = ['Texto1', 'Fecha', 'Texto2', 'Texto3', 'Beneficiario', 'Texto4',
       'Conc', 'Texto5', 'CtaDebito', 'Texto6', 'Importe', 'Texto7', 'Vto', '
      Texto8', 'Cuota', 'Texto9']
37 df = pd.DataFrame(datos_totales, columns=columnas)
```

Estas líneas de código aplican a la manipulación de datos extraídos de archivos PDF, con el objetivo de convertirlos en un DataFrame de pandas. Con una orientación para estudiantes de ciencias económicas que se inician en programación se puede relacionar cada línea con su objetivo:

En la línea 1: Importación de bibliotecas y módulos: - import pandas as pd: Importa la biblioteca pandas bajo el alias “pd”, que se usa comúnmente para manipular y analizar datos tabulares en Python. - Se importa el módulo “re” para el manejo de expresiones regulares, y “PyPDF2” para trabajar con archivos PDF.

En la línea 2: Definición del patrón de búsqueda: - Se define una expresión regular con el nombre “patron_ticket” que se utiliza para buscar patrones específicos en el texto de los archivos PDF. Este patrón logra extraer información relacionada con los pagos, fechas, cuentas bancarias, etc.

En la línea 3: Funciones de extracción de datos: - extraer_datos(texto): Esta función toma el texto extraído de un PDF y busca coincidencias con el patrón definido. Si se encuentra una coincidencia, devuelve una tupla con los grupos coincidentes. - pdf_a_datos(pdf_path): Esta función toma la ruta de un archivo PDF, lo lee página por página, extrae el texto de cada página y utiliza la función “extraer_datos()” para extraer los datos relevantes. Devuelve una lista de todas las tuplas de datos extraídas.

En la línea 4: Procesamiento de archivos PDF: - Se genera una lista “archivos_pdf” que contiene los nombres de todos los archivos PDF en el directorio “directorio_renombrados”. - Para cada archivo PDF, se llama a la función “pdf_a_datos()” para extraer los datos de ese archivo y se agregan a la lista “datos_totales”.

En la línea 5: Creación de un DataFrame de pandas: - Se crea un DataFrame de pandas “df” utilizando la lista de datos extraídos “datos_totales”. Se especifican las columnas del DataFrame como una lista de nombres. - Cada fila del DataFrame corresponde a una tupla de datos extraída de un archivo PDF. - Este DataFrame puede ser utilizado para realizar análisis posterior de los datos extraídos de los archivos PDF, como la creación de informes, visualizaciones o cálculos estadísticos.

Para resumir, este código automatiza el proceso de extracción y estructuración de datos de archivos PDF, lo cual es fundamental en ciencias económicas para analizar información financiera, informes, facturas, y otros documentos relacionados con transacciones comerciales y financieras.

Se puede visualizar el resultado de la creación de un Dataframe llamado “df”, con la siguiente instrucción:

```
In [6]: df.head()
```

En la pantalla se visualizará:

Out[6]:

	Fecha	Texto3	Beneficiario	Texto4	Conc	Texto5	CtaDebito	Texto6	Importe	Texto7	Vto	Texto8	C
0	05/05/2023	PAGO DE	ADMPROVIMPUESTOSSANTAFE	NRO. DE CLIENTE:	00241430225164040	CON DEBITO EN:	CAJA DE AHORRO EN PESOSNRO DE CUENTA: 50009806...	IMPORTE: $	5448,55	FECHA VENCIMIENTO:	18/05/23	CUOTA:	0(
1	05/05/2023	PAGO DE	COMP.DE SEGUROS EL NORTECUOTA	NRO. DE CLIENTE:	1329385POLIZA: 3618733 000	CON DEBITO EN:	CAJA DE AHORRO EN PESOSNRO DE CUENTA: 50009806...	IMPORTE: $	15820,00	FECHA VENCIMIENTO:	27/04/23	CUOTA:	0(
2	05/05/2023	PAGO DE	MUN DE SANTA FE INMOBI	NRO. DE CLIENTE:	267000727672	CON DEBITO EN:	CAJA DE AHORRO EN PESOSNRO DE CUENTA: 50009806...	IMPORTE: $	4754,30	FECHA VENCIMIENTO:	31/05/23	CUOTA:	0(

Una vez creado un dataframe, es posible continuar con la librería Pandas que tiene un enorme potencial para la limpieza de datos.

Se puede proponer la eliminación de aquellas columnas que no sean de utilidad. Se propone a continuación la eliminación de la primera, la tercera y la última columna por su orden

In [7]:
```
columnas_a_eliminar = [0, 2, -1]
df = df.drop(df.columns[columnas_a_eliminar], axis=1)
df.head(5)
```

En la pantalla se visualizará:

Out[7]:

	Fecha	Texto3	Beneficiario	Texto4	Conc	Texto5	CtaDebito	Texto6	Importe	Texto7	Vto	Texto8	C
0	05/05/2023	PAGO DE	ADMPROVIMPUESTOSSANTAFE	NRO. DE CLIENTE:	00241430225164040	CON DEBITO EN:	CAJA DE AHORRO EN PESOSNRO DE CUENTA: 50009806...	IMPORTE: $	5448,55	FECHA VENCIMIENTO:	18/05/23	CUOTA:	0(
1	05/05/2023	PAGO DE	COMP.DE SEGUROS EL NORTECUOTA	NRO. DE CLIENTE:	1329385POLIZA: 3618733 000	CON DEBITO EN:	CAJA DE AHORRO EN PESOSNRO DE CUENTA: 50009806...	IMPORTE: $	15820,00	FECHA VENCIMIENTO:	27/04/23	CUOTA:	0(
2	05/05/2023	PAGO DE	MUN DE SANTA FE INMOBI	NRO. DE CLIENTE:	267000727672	CON DEBITO EN:	CAJA DE AHORRO EN PESOSNRO DE CUENTA: 50009806...	IMPORTE: $	4754,30	FECHA VENCIMIENTO:	31/05/23	CUOTA:	0(

In [8]:
```
df = df.loc[:, ~df.columns.str.startswith('Texto')]
```

Esta línea de código filtra las columnas del DataFrame df para excluir aquellas cuyos nombres comienzan con "Texto".

In [9]:
```
df.drop('Conc', axis=1, inplace=True)
df.head(2)
```

df.drop('Conc', axis=1, inplace=True): Utilizando el método drop(), elimina la columna llamada 'Conc' del DataFrame df. El parámetro axis=1 especifica que se trata de una eliminación de columna (en contraposición a las filas), y inplace=True indica que los cambios deben realizarse directamente en el DataFrame df.

df.head(2): Utilizando el método head(), muestra las primeras dos filas del DataFrame df. Esto es útil para revisar las primeras filas del DataFrame y asegurarse de que las operaciones de manipulación de datos se realizaron correctamente.

En síntesis, estas líneas de código eliminan la columna 'Conc' del DataFrame df y luego muestran las primeras dos filas del DataFrame actualizado. Esto podría ser útil para eliminar columnas que no son necesarias en el análisis de datos y para verificar el estado del DataFrame después de la eliminación de la columna.

De esta manera el dataframe quedará compuesto por solo por seis columnas, y es posible avanzar con el proceso de limpieza del mismo eliminando la expresión "CAJA DE AHORRO EN PESOSNRO DE CUENTA: " que se presenta en todas las celdas de la columna CtaDebito:

In [10]:
```
df = df.apply(lambda col: col.str.replace('CAJA DE AHORRO EN PESOSNRO DE
    CUENTA: ', ''))
df.head(3)
```

Define una función anónima que toma cada columna (col) del DataFrame y reemplaza todas las instancias de la cadena 'CAJA DE AHORRO EN PESOSNRO DE CUENTA: ' por una cadena vacía ' '. Esto se hace utilizando el método str.replace() de pandas, que reemplaza todas las ocurrencias de una cadena con otra en objetos de tipo cadena.

Out[10]:

	Fecha	Beneficiario	CtaDebito	Importe	Vto	Cuota
0	05/05/2023	ADMPROVIMPUESTOSSANTAFE	500098066908	5448,55	18/05/23	002/23
1	05/05/2023	COMP.DE SEGUROS EL NORTECUOTA	500098066908RAMA: 04	15820,00	27/04/23	000/66

In [11]:
```
nombre = input('Ingrese el nombre con el cual quiere guardar el archivo: ')
```

Esta línea de código muestra un mensaje al usuario solicitando que ingrese un nombre. La función input() toma la entrada del usuario desde la consola y espera a que el usuario escriba algo y presione Enter.

El texto que consgnado entre paréntesis es el mensaje que se muestra al usuario, en este caso, “Ingrese el nombre con el cual quiere guardar el archivo: ”.

Una vez que el usuario proporciona una entrada, esta se guarda en la variable nombre. La entrada se guarda como una cadena de caracteres (string), y puede ser utilizada más adelante en el código para nombrar un archivo o realizar otras operaciones, según sea necesario.

Se asume que el nombre con el que se guardarán los registros de pago, se refiere a la fecha en que se han realizado.

```
Out[11]: Ingrese el nombre con el cual quiere guardar el archivo: 20230505
```

Este paso también podría automatizarse, si se adoptara la convención de realizarse una sola vez al día para todos los pagos del día.

```
In [12]: ruta_destino = (f'C:/Tickets renombrados/Registros de pagos {nombre}.xlsx')
```

Con la inicialización con una letra f y comilla simple se indica que la ruta destino tiene la forma de una f-string, es decir una cadena de formato en Python que permite insertar valores de variables, entre llaves, directamente dentro de una cadena. Se identifica por el prefijo f antes de las comillas. Dentro de las llaves , puedes incluir el nombre de la variable cuyo valor deseas insertar en la cadena.

‘C:/Tickets renombrados/Registros de pagos nombre.xlsx’: Esta es la cadena de formato en sí. Representa la ruta completa del archivo donde se desea guardar el registro de pagos en formato Excel. ‘C:/Tickets renombrados/’ es la carpeta donde se guardarán los archivos. ‘Registros de pagos nombre.xlsx’ es el nombre del archivo. El valor de nombre será insertado en este lugar debido a la f-string. Se asume que nombre es una variable definida anteriormente en el código.

Resumiendo, esta línea de código crea una ruta de archivo dinámica que incluye el nombre proporcionado por el usuario (nombre) dentro del nombre del archivo. Esto permite guardar el archivo en una ubicación específica con un nombre que el usuario haya elegido.

```
In [13]: df_a_crear.to_excel(ruta_destino, index=False)
```

df_a_crear: Este es el DataFrame que se desea guardar en el archivo Excel.

.to_excel(): Este es el método de pandas que se utiliza para guardar un DataFrame en un archivo Excel.

ruta_destino: Es la ruta completa del archivo Excel donde se guardará el DataFrame. Esta variable se ha definido anteriormente utilizando una f-string y contiene el nombre de archivo indicado por el usuario.

index=False: Este argumento le indica a pandas que no incluya el índice del DataFrame en el archivo Excel. Si index=True, el índice del DataFrame se guardará como una columna adicional en el archivo Excel.

En síntesis, esta línea de código toma el DataFrame df_a_crear y lo guarda en un archivo Excel en la ruta especificada por ruta_destino, sin incluir el índice del DataFrame en el archivo Excel. Esto es útil para exportar datos de pandas a un formato más ampliamente utilizado como Excel para su análisis o presentación

23.2. Ejercicio Práctico N° 2: Combinar archivos del sistema con el que se liquidan los sueldos de los empleados para generar un archivo compatible para la carga de transferencias bancarias a las cuentas sueldos

Presentación del problema: La compañía LA POSADA carga en la página web del banco un archivo de extensión TXT para realizar el pago, por lotes, de los haberes de sus empleados.

El archivo debe contener un renglón por cada empleado identificando la Clave Bancaria Uniforme, el día del pago, el monto a transferir, la clave única de identificación tributaria del empleado y el motivo por el cual se transfiere.

Cada campo tiene características específicas y deben estar separados por "punto y coma" lo cual debe respetarse para que el archivo sea correctamente procesado por el sistema web dispuesto por la entidad bancaria.

Analice esta situación a los efectos de proponer mediante un proceso automatizado que se integren diferentes archivos de información a efectos de generar el archivo de carga al Banco.

- Campo CBU: tipo numérico de 22 dígitos, Ej: 4260010100200014270003.
- Campo Fecha de pago: tipo numérico de 10 dígitos con separadores de "/" de formato dd/mm/aaaa, Ej: 02/03/2023.
- Campo Monto: a transferir tipo numérico con decimales separados por ".", Ej: 288833.02.
- Campo CUIT: tipo numérico de 11 dígitos, Ej: 20309041810.
- Campo Motivo: tipo alfanumérico de 12 caracteres máximo, Ej: Haberes202402.

Ejemplo de renglón por empleado:

4260010100200014270006;02/03/2024;288833.02;20309041810;Haberes202402

In [1]:
```
1 import pandas as  pd
2 import os
3 import openpyxl
```

Se analizarán a continuación el significado y utilidad de las líneas de código mas sustanciales para cada entrada. Comenzando con la entrada 1 o In [1]:

En la línea 1: import pandas as pd: Con este código, se logra importar una biblioteca de Python llamada "pandas". Pandas es una herramienta poderosa que permite el análisis y la

manipulación de datos. Se utiliza comúnmente en ciencias económicas para manejar conjuntos de datos, realizar cálculos y análisis estadísticos, y visualizar datos de manera eficiente. Al importar pandas como "pd", estamos creando un alias que nos permite hacer referencia a las funciones y clases de pandas de una manera más concisa y legible en nuestro código.

En la línea 2: import os: Esta línea importa el módulo "os" de Python. El módulo "os" proporciona una manera de interactuar con el sistema operativo subyacente en el que se ejecuta Python. Esto es útil en la práctica profesional cuando necesitamos acceder a archivos en el sistema de archivos, manipular rutas de archivos, o realizar otras operaciones relacionadas con el sistema.

En la línea 3: import openpyxl: Openpyxl es una biblioteca de Python que permite leer y escribir archivos de Excel en formato xlsx. En ciencias económicas, a menudo trabajamos con datos almacenados en hojas de cálculo de Excel, y openpyxl nos proporciona las herramientas necesarias para interactuar con estos archivos directamente desde Python. Podemos leer datos de hojas de cálculo, escribir nuevos datos en hojas de cálculo existentes y realizar diversas manipulaciones de datos utilizando esta biblioteca. La importación de openpyxl nos permite utilizar sus funciones en nuestro código.

In [2]:
```
periodo_liquidado = input('Ingrese el periodo liquidado con formato AAAAMM:
    ')
año=periodo_liquidado[0:4]
mes= periodo_liquidado[4:6]
print(periodo_liquidado)
print(año)
print(mes)
```

Estas líneas de código Python está diseñado para capturar la entrada del usuario para el período liquidado en un formato específico AAAAMM, y luego separar el año y el mes de esa entrada para su posterior procesamiento. Se puede relacionar el significado de cada línea a partir de:

En la línea 1: periodo_liquidado = input(‘Ingrese el periodo liquidado con formato AAAAMM: ’): Esta línea solicita al usuario que ingrese el período liquidado en un formato específico, AAAAMM. La función "input()" muestra un mensaje al usuario y espera a que el usuario ingrese una respuesta. La respuesta del usuario se guarda en la variable "periodo_liquidado".

En la línea 2: año = periodo_liquidado[0:4]: Esta línea extrae los primeros cuatro caracteres de la cadena ‘periodo_liquidado‘, que representan el año (los primeros cuatro dígitos del formato AAAAMM). Estos caracteres se asignan a la variable "año".

En la línea 3: mes = periodo_liquidado[4:6]: Similarmente, esta línea extrae los caracteres 5º y 6º de la cadena "periodo_liquidado", que representan el mes (los dos dígitos siguientes en el formato AAAAMM). Estos caracteres se asignan a la variable "mes".

En la línea 4: print(periodo_liquidado): Esta línea imprime en la consola el valor ingresado por el usuario en ‘periodo_liquidado‘, simplemente para confirmar la entrada.

En la línea 5: print(año): Esta línea imprime en la consola el año extraído de la entrada del usuario.

En la línea 6: print(mes): Esta línea imprime en la consola el mes extraído de la entrada del usuario.

Para resumir, esta secuencia de código toma la entrada del usuario en formato AAAAMM, la separa en año y mes, y luego imprime tanto la entrada completa como los valores separados de año y mes. Es útil para procesar fechas introducidas por el usuario y descomponerlas en partes significativas para su posterior uso en cálculos o análisis, y en la salida N° 2 (Out:[2]) se verifica:

```
Out[2]: Ingrese el periodo liquidado con formato AAAAMM: 202402
        202402
        2024
        02
```

Se reitera la importancia de parametrizar nombres y directorios donde se guardan los archivos para que cuando se trata de tareas que son reiteradas, diaria o mensualmente los valores que se solicitan mediante la interfaz y que son ingresados por el usuario redirijan el procesamiento de los archivos correspondientes al día o mes deseado.

```
In [3]: Base = 'C:/Base empleados.xlsx'
        Liquidador = (f'C:/LiquidadorSueldos {periodo_liquidado}.xlsx')
```

La utilidad de estas líneas se explican mediante:

En la línea 1: Base = ‘C:/Base empleados.xlsx’: Esta línea asigna la ruta de un archivo Excel a la variable “Base”. La ruta especifica la ubicación del archivo “Base empleados.xlsx” en el sistema de archivos. En este caso, el archivo está ubicado en la raíz del disco C: en Windows. Este archivo contiene los valores estáticos de cada empleado, como ser el nombre, su clave de identifiación tributaria, y los datos de su cuenta bancaria.

En la línea 2: Liquidador = (f‘C:/LiquidadorSueldos{periodo_liquidado}.xlsx’): Esta línea utiliza una cadena de formato f-string para crear la ruta de un archivo Excel llamado “LiquidadorSueldos” seguido del periodo liquidado ingresado por el usuario. El periodo liquidado es una variable que ha sido definida capturada con el código en la Salida N°2. La variable “Liquidador” ahora contiene la ruta completa del archivo Excel basada en la ubicación en el disco C: y el periodo liquidado definido por el usuario. Desde este archivo, generado por el sistema liquidador de sueldos, se extraerán los datos correspondientes al mes de Febrero de 2024, por ejemplo el importe a transferir en concepto de sueldos.

Sintetizando, estas líneas de código definen dos rutas de archivo en formato de cadena. La primera ruta, “Base”, apunta a un archivo llamado “Base empleados.xlsx”. La segunda ruta, “Liquidador”, utiliza un nombre de archivo dinámico que incluye el periodo liquidado ingresado por el usuario y está destinado a un archivo llamado “LiquidadorSueldos AAAAMM.xlsx”. Esto tiene como utilidad dejar establecido la ubicación y nombre de los archivos que se procesarán automáticamente.

In [4]:
```
archivo_1 = pd.read_excel(Base, sheet_name = 'Nomina')
archivo_2 = pd.read_excel(Liquidador, sheet_name = periodo_liquidado)
```

Estas líneas de código en Python utilizan la biblioteca pandas para leer datos de archivos Excel. Su funcionalidad está dada por:

archivo_1 = pd.read_excel(Base, sheet_name=‘Nomina’): Esta línea lee datos del archivo Excel especificado por la variable Base. Utiliza la función pd.read_excel() de pandas para cargar los datos del archivo Excel en un DataFrame de pandas. El parámetro sheet_name especifica el nombre de la hoja dentro del archivo Excel de la cual se van a leer los datos, en este caso, la hoja se llama ‘Nomina’. El resultado se guarda en la variable archivo_1, que será un DataFrame de pandas que contiene los datos de la hoja ‘Nomina’ del archivo Excel.

archivo_2 = pd.read_excel(Liquidador, sheet_name=periodo_liquidado): Esta línea lee datos del archivo Excel especificado por la variable Liquidador, que apunta a un archivo de nombres dinámicos según el periodo liquidado. De manera similar al primer caso, se utiliza la función pd.read_excel() para cargar los datos del archivo Excel en un DataFrame de pandas. El parámetro sheet_name se establece en el valor de periodo_liquidado, que es utilizaddo como una variable que contiene el nombre de una hoja dentro del archivo Excel. Esta línea carga los datos de una hoja específica del archivo Excel en un DataFrame llamado archivo_2.

Resumiendo, estas líneas de código leen datos de dos archivos Excel diferentes utilizando pandas. archivo_1 contiene datos de la hoja ‘Nomina’ del archivo ‘Base empleados.xlsx’, mientras que archivo_2 contiene datos de una hoja específica en el archivo ’LiquidadorSueldos AAAAMM.xlsx’, donde AAAAMM es el periodo liquidado ingresado por el usuario.

In [5]:
```
archivo_1.dtypes
```

El método .dtypes aplicado a un DataFrame en pandas devuelve los tipos de datos de cada una de las columnas en ese DataFrame. Proporciona información sobre el tipo de datos que pandas ha inferido para cada columna en el DataFrame.

Cuando se ejecuta archivo_1.dtypes, se obtiene una serie que muestra los tipos de datos de cada columna en el DataFrame archivo_1.

El método .dtypes es útil para comprender la estructura de tus datos y asegurarte de

que estén en el formato correcto para realizar operaciones y análisis posteriores. Te permite conocer qué tipo de datos contiene cada columna, lo que es crucial para realizar manipulaciones y cálculos específicos en tu DataFrame.

Out[5]:
```
Apellido y Nombre     object
CBU                   object
CUIT                   int64
PUESTO                object
EMPLEADOR             object
dtype: object
```

In [6]:
```
archivo_1.head()
```

Out[6]:

	Apellido y Nombre	CBU	CUIT	PUESTO	EMPLEADOR
0	PEREZ MARTIN	4260010100200014270006	20309041810	MANTENIMIENTO	LA POSADA
1	MARTINEZ PEDRO	4260010100200014258015	27293540379	LAVADERO	LA POSADA
2	SALAS RAUL	0170210340000011351337	20266336978	FRONT	LA POSADA
3	VORLA MARINA	0170210340000011351719	20348278909	GERENTE	LA POSADA
4	KLEMENT SAMANTA	0170210340000011352019	27360556285	HOUSEKEEPING	LA POSADA
5	BIDEN MARTINA	4260010100200018445046	27377017361	HOUSEKEEPING	LA POSADA
6	NOCE AUGUSTO	0170210340000011351955	27356540641	HOUSEKEEPING	LA POSADA
7	CANSIDO RAMIRO	0170210340000011056423	20271482516	MANTENIMIENTO	LA POSADA

In [7]:
```
archivo_2.head()
```

Out[7]:

	Apellido y Nombre	CUIT	Remuneraciones a pagar	Empleador
0	PEREZ MARTIN	20309041810	288834.02	LA POSADA
1	MARTINEZ PEDRO	27293540379	288834.02	LA POSADA
2	SALAS RAUL	20266336978	299366.72	LA POSADA
3	VORLA MARINA	20348278909	289668.44	LA POSADA
4	KLEMENT SAMANTA	27360556285	243685.18	LA POSADA

In [8]:
```
left_df = archivo_1[['Apellido y Nombre',`CBU','CUIT']]
right_df = archivo_2[['CUIT', 'Remuneraciones a pagar']]
archivo_final=pd.merge(left_df, right_df, on='CUIT')
```

Con estas instrucciones en código Python se realiza una operación de fusión (merge) entre dos DataFrames utilizando pandas en Python. Puede facilitarse la interpretación del código mediante:

left_df = archivo_1[[‘Apellido y Nombre’,‘CBU’,‘CUIT’]]: Selecciona las columnas ‘Apellido y Nombre’, ‘CBU’ y ‘CUIT’ del DataFrame archivo_1 y las asigna al nuevo DataFrame left_df. Este DataFrame contendrá la información de los nombres, números de CBU y CUIT de los empleados.

right_df = archivo_2[[‘CUIT’, ‘Remuneraciones a pagar’]]: Selecciona las columnas ‘CUIT’ y “Remuneraciones a pagar” del DataFrame archivo_2 y las asigna al nuevo DataFrame right_df. Este DataFrame contendrá la información de los números de CUIT y las remuneraciones a pagar a los empleados.

archivo_final=pd.merge(left_df, right_df, on=‘CUIT’): Utiliza la función pd.merge() de pandas para fusionar (merge) los DataFrames left_df y right_df en base a la columna ‘CUIT’, que actúa como clave de fusión. Esto crea un nuevo DataFrame llamado archivo_final que contiene las columnas ‘Apellido y Nombre’, ‘CBU’, ‘CUIT’ y “Remuneraciones a pagar”. La fusión se realiza de manera que coincidan los valores de ‘CUIT’ en ambos DataFrames.

Sintetizando, este código combina la información de dos conjuntos de datos diferentes, basándose en un campo común (‘CUIT’ en este caso), lo que puede ser útil para consolidar información relacionada de diferentes fuentes de datos en un solo DataFrame.

In [9]:
```
archivo_final.head()
```

Out[9]:

	Apellido y Nombre	CBU	CUIT	Remuneraciones a pagar
0	PEREZ MARTIN	4260010100200014270006	20309041810	288834.02
1	MARTINEZ PEDRO	4260010100200014258015	27293540379	288834.02
2	SALAS RAUL	0170210340000011351337	20266336978	299366.72
3	VORLA MARINA	0170210340000011351719	20348278909	289668.44
4	KLEMENT SAMANTA	0170210340000011352019	27360556285	243685.18

In [10]:
```
1 fecha_pago = input('Ingrese la fecha de pago DDMMAAAA: ')
2 fecha_pago_corregida=str(fecha_pago[0:2]\+'/'\+fecha_pago[2:4]+'/'+
     fecha_pago[4:8])
3 print(fecha_pago_corregida)
```

Este fragmento de código Python permite al usuario ingresar una fecha en formato DDMMAAAA, y luego la transforma en el formato comúnmente utilizado Día/Mes/Año (DD/MM/AAAA).

Mediante la línea 1:

fecha_pago = input(‘Ingrese la fecha de pago DDMMAAAA: ’): Esta línea solicita al usuario que ingrese la fecha de pago en el formato DDMMAAAA utilizando la función “input()”. La respuesta del usuario se almacena en la variable “fecha_pago”.

En la línea 2:

fecha_pago_corregida=str(fecha_pago[0:2]+‘/’+fecha_pago[2:4]+‘/’+fecha_pago[4:8]): Esta línea crea una nueva cadena llamada “fecha_pago_corregida” que toma los primeros dos caracteres de “fecha_pago” (que representan el día), seguido por un ‘/’, luego los siguientes dos caracteres (que representan el mes), seguido por otro ’/’, y finalmente los últimos cuatro caracteres (que representan el año). Esto crea una cadena en el formato DD/MM/AAAA. El método ‘str()‘ se utiliza para asegurarse de que la concatenación se realice correctamente, ya que los slices o cortes realizan una selección de caracteres pero devuelven un cadena de tipo “string”.

En la línea 3: print(fecha_pago_corregida): Finalmente, esta línea imprime la fecha de pago corregida en el formato DD/MM/AAAA en la consola.

En resumen, este código permite al usuario ingresar una fecha en un formato específico

y luego la presenta de manera más convencional para facilitar su comprensión y visualización.

```
Out[10]: Ingrese la fecha de pago DDMMAAAA: 05032024
         05/03/2024

In [11]: motivo_pago = input('Ingrese el motivo del pago: ')
```

Este script de Python solicita al usuario que ingrese un motivo de pago y almacena la entrada en la variable motivo_pago.

La línea motivo_pago = input(‘Ingrese el motivo del pago: ’): muestra un mensaje al usuario solicitando que ingrese el motivo del pago utilizando la función input(). La entrada del usuario se guarda en la variable motivo_pago.

```
Out[11]: Ingrese el motivo del pago: HABERES

In [12]: archivo_final.insert(loc=2, column = 'Fecha de pago', value=
             fecha_pago_corregida)
         archivo_final.insert(loc=5, column = 'Motivo', value= motivo_pago)
```

Estas líneas de código en un contexto orientado a estudiantes de ciencias económicas.

import pandas as pd: Con este código, se “llama” a la biblioteca de Python llamada “Pandas”. Pandas es una herramienta poderosa que permite el análisis y la manipulación de datos. Se utiliza comúnmente en ciencias económicas para manejar conjuntos de datos, realizar cálculos y análisis estadísticos, y visualizar datos de manera eficiente. Al importar pandas como “pd”, estamos creando un alias que nos permite hacer referencia a las funciones y clases de pandas de una manera más concisa y legible en nuestro código.

import os: Esta línea importa el módulo “os” de Python. El módulo “os” proporciona una manera de interactuar con el sistema operativo subyacente en el que se ejecuta Python. Esto es útil en la práctica profesional cuando necesitamos acceder a archivos en el sistema de archivos, manipular rutas de archivos, o realizar otras operaciones relacionadas con el sistema.

from IPython.display import Latex: IPython es un entorno interactivo para ejecutar código Python de manera interactiva y exploratoria. Esta línea de código importa una función específica llamada “Latex” del módulo “display” dentro de IPython. La función Latex se utiliza para mostrar fórmulas matemáticas escritas en el lenguaje de marcado LaTeX dentro de un entorno IPython. Esto es útil en la práctica profesional cuando necesitamos presentar ecuaciones matemáticas de una manera legible y formateada en nuestros notebooks o scripts de Python.

import openpyxl: Openpyxl es una biblioteca de Python que permite leer y escribir archivos de Excel en formato xlsx. En ciencias económicas, a menudo trabajamos con datos almacenados en hojas de cálculo de Excel, y openpyxl nos proporciona las herramientas necesarias

para interactuar con estos archivos directamente desde Python. Podemos leer datos de hojas de cálculo, escribir nuevos datos en hojas de cálculo existentes y realizar diversas manipulaciones de datos utilizando esta biblioteca. La importación de openpyxl nos permite utilizar sus funciones en nuestro código.

periodo_liquidado = input(‘Ingrese el periodo liquidado con formato AAAAMM: ’) año=periodo_liquidado[0:4] mes= periodo_liquidado[4:6] print(periodo_liquidado) print(año) print(mes)

El objetivo de estas líneas es capturar en una variable la entrada que el usuario haya asignado para el período liquidado en un formato específico AAAAMM, y luego separar el año y el mes de esa entrada para su posterior procesamiento. Concretamente:

periodo_liquidado = input(‘Ingrese el periodo liquidado con formato AAAAMM: ’): Esta línea solicita al usuario que ingrese el período liquidado en un formato específico, AAAAMM. La función input() muestra un mensaje al usuario y espera a que el usuario ingrese una respuesta. La respuesta del usuario se guarda en la variable periodo_liquidado.

año = periodo_liquidado[0:4]: Esta línea extrae los primeros cuatro caracteres de la cadena periodo_liquidado, que representan el año (los primeros cuatro dígitos del formato AAAAMM). Estos caracteres se asignan a la variable año.

mes = periodo_liquidado[4:6]: Similarmente, esta línea extrae los caracteres 5^{o} y 6^{o} de la cadena periodo_liquidado, que representan el mes (los dos dígitos siguientes en el formato AAAAMM). Estos caracteres se asignan a la variable mes.

print(periodo_liquidado): Esta línea imprime en la consola el valor ingresado por el usuario en periodo_liquidado, simplemente para confirmar la entrada.

print(año): Esta línea imprime en la consola el año extraído de la entrada del usuario.

print(mes): Esta línea imprime en la consola el mes extraído de la entrada del usuario.

Para resumir, este código toma la entrada del usuario en formato AAAAMM, la divide en año y mes, y luego imprime tanto la entrada completa como los valores separados de año y mes. Es útil para procesar fechas introducidas por el usuario y descomponerlas en partes significativas para su posterior uso en cálculos o análisis.

Base = ‘C:/Base empleados.xlsx’

Liquidador = (f‘C:/LiquidadorSueldos periodo_liquidado.xlsx’)

Este script de Python define dos rutas de archivos, desde los cuales se procesará la información para el cumplimiento del objetivo planteado en el problema.

Base = 'C:/Base empleados.xlsx': Esta línea asigna la ruta de un archivo Excel a la variable Base. La ruta especifica la ubicación del archivo "Base empleados.xlsx" en el sistema de archivos. En este caso, el archivo está ubicado en la raíz del disco C: en Windows.

Liquidador = (f'C:/LiquidadorSueldos periodo_liquidado.xlsx'): Esta línea utiliza una cadena de formato f-string para crear la ruta de un archivo Excel llamado "LiquidadorSueldos" seguido del periodo liquidado ingresado por el usuario. El periodo liquidado es una variable definida previamente en el código. La variable Liquidador ahora contiene la ruta completa del archivo Excel basada en la ubicación en el disco C: y el periodo liquidado asignado por el usuario.

En resumen, estas líneas de código definen dos rutas de archivo en formato de cadena. La primera ruta, Base, apunta a un archivo llamado "Base empleados.xlsx". La segunda ruta, Liquidador, utiliza un nombre de archivo dinámico que incluye el periodo liquidado ingresado por el usuario y está destinado a un archivo llamado "LiquidadorSueldos AAAAMM.xlsx".

archivo_1 = pd.read_excel(Base, sheet_name = 'Nomina') archivo_2 = pd.read_excel(Liquidador, sheet_name = periodo_liquidado)

Estas líneas de código en Python utilizan la biblioteca pandas para leer datos de archivos Excel. Su explicación está dada por:

archivo_1 = pd.read_excel(Base, sheet_name='Nomina'): Esta línea lee datos del archivo Excel especificado por la variable Base. Utiliza la función pd.read_excel() de pandas para cargar los datos del archivo Excel en un DataFrame de pandas. El parámetro sheet_name especifica el nombre de la hoja dentro del archivo Excel de la cual se van a leer los datos, en este caso, la hoja se llama 'Nomina'. El resultado se guarda en la variable archivo_1, que será un DataFrame de pandas que contiene los datos de la hoja 'Nomina' del archivo Excel.

archivo_2 = pd.read_excel(Liquidador, sheet_name=periodo_liquidado): Esta línea lee datos del archivo Excel especificado por la variable Liquidador, que apunta a un archivo de nombres dinámicos según el periodo liquidado. De manera similar al primer caso, se utiliza la función pd.read_excel() para cargar los datos del archivo Excel en un DataFrame de pandas. El parámetro sheet_name se establece en el valor de periodo_liquidado, que es una variable que contiene el nombre de una hoja dentro del archivo Excel. Esta línea carga los datos de una hoja específica del archivo Excel en un DataFrame llamado archivo_2.

Resumiendo, estas líneas de código leen datos de dos archivos Excel diferentes utilizando pandas. archivo_1 contiene datos de la hoja 'Nomina' del archivo 'Base empleados.xlsx', mientras que archivo_2 contiene datos de una hoja específica en el archivo 'LiquidadorSueldos

AAAAMM.xlsx', donde AAAAMM es el periodo liquidado ingresado por el usuario.

El método .dtypes aplicado a un DataFrame en pandas devuelve los tipos de datos de cada una de las columnas en ese DataFrame. Proporciona información sobre el tipo de datos que pandas ha inferido para cada columna en el DataFrame.

Cuando se ejecuta archivo_1.dtypes, obtendrás una serie que muestra los tipos de datos de cada columna en el DataFrame archivo_1.

Por ejemplo, si archivo_1 contiene datos de empleados como nombres, edades, salarios, etc., archivo_1.dtypes te mostrará algo como esto:

vbnet

Nombre object Edad int64 Salario float64 Departamento object dtype: object

En este ejemplo hipotético, Nombre y Departamento son de tipo object, lo que generalmente significa que contienen cadenas de texto. Edad es de tipo int64, lo que significa que contiene números enteros de 64 bits. Salario es de tipo float64, lo que significa que contiene números de punto flotante de 64 bits.

El método .dtypes es útil para comprender la estructura de tus datos y asegurarte de que estén en el formato correcto para realizar operaciones y análisis posteriores. Te permite conocer qué tipo de datos contiene cada columna, lo que es crucial para realizar manipulaciones y cálculos específicos en tu DataFrame.

left_df = archivo_1[['Apellido y Nombre','CBU','CUIT']] right_df = archivo_2[['CUIT', 'Remuneraciones a pagar']] archivo_final=pd.merge(left_df, right_df, on='CUIT')

Con estas líneas de código se realiza una operación de fusión (merge) entre dos DataFrames utilizando pandas en Python. Se detalla paso a paso, el significado de cada una de las instrucciones:

left_df = archivo_1[['Apellido y Nombre','CBU','CUIT']]: Selecciona las columnas 'Apellido y Nombre', 'CBU' y 'CUIT' del DataFrame archivo_1 y las asigna al nuevo DataFrame left_df. Este DataFrame contendrá la información de los nombres, números de CBU y CUIT de los empleados.

right_df = archivo_2[['CUIT', 'Remuneraciones a pagar']]: Selecciona las columnas 'CUIT' y 'Remuneraciones a pagar' del DataFrame archivo_2 y las asigna al nuevo DataFrame right_df. Este DataFrame contendrá la información de los números de CUIT y las remuneraciones a pagar a los empleados.

archivo_final=pd.merge(left_df, right_df, on='CUIT'): Utiliza la función pd.merge() de pandas para fusionar (merge) los DataFrames left_df y right_df en base a la columna 'CUIT', que actúa como clave de fusión. Esto crea un nuevo DataFrame llamado archivo_final que contiene las columnas 'Apellido y Nombre', 'CBU', 'CUIT' y 'Remuneraciones a pagar'. La fusión se realiza de manera que coincidan los valores de CUIT' en ambos DataFrames.

Para resumir, este código combina la información de dos conjuntos de datos diferentes, basándose en un campo común ('CUIT' en este caso), lo que puede ser útil para consolidar información relacionada de diferentes fuentes de datos en un solo DataFrame.

fecha_pago = input('Ingrese la fecha de pago DDMMAAAA: ') fecha_pago_corregida=str(fecha_pago[0
print(fecha_pago_corregida)

Este fragmento de código Python permite al usuario ingresar una fecha en formato DDMMAAAA, y luego la transforma en el formato comúnmente utilizado Día/Mes/Año (DD/MM/AAAA). Cada fragmento del código se explica mendiante:

fecha_pago = input('Ingrese la fecha de pago DDMMAAAA: '): Esta línea solicita al usuario que ingrese la fecha de pago en el formato DDMMAAAA utilizando la función input(). La respuesta del usuario se almacena en la variable fecha_pago.

fecha_pago_corregida=str(fecha_pago[0:2]+'/'+fecha_pago[2:4]+'/'+fecha_pago[4:8]): Esta línea crea una nueva cadena llamada fecha_pago_corregida que toma los primeros dos caracteres de fecha_pago (que representan el día), seguido por un '/', luego los siguientes dos caracteres (que representan el mes), seguido por otro '/', y finalmente los últimos cuatro caracteres (que representan el año). Esto crea una cadena en el formato DD/MM/AAAA. El método str() se utiliza para asegurarse de que la concatenación se realice correctamente, ya que los slices realizan una selección de caracteres pero devuelven un tipo 'str'.

print(fecha_pago_corregida): Finalmente, esta línea imprime la fecha de pago corregida en el formato DD/MM/AAAA en la consola.

Sintetizando, este código permite al usuario ingresar una fecha en un formato específico y luego la presenta de manera más convencional para facilitar su comprensión y visualización.

motivo_pago = input('Ingrese el motivo del pago: ')

Este script de Python solicita al usuario que ingrese un motivo de pago y almacena la entrada en la variable motivo_pago. :

Mediante motivo_pago = input('Ingrese el motivo del pago: '): Esta línea muestra un mensaje al usuario solicitando que ingrese el motivo del pago utilizando la función input(). La

entrada del usuario se guarda en la variable motivo_pago.

Por ejemplo, si el programa se ejecuta y muestra “Ingrese el motivo del pago: ” en la consola, el usuario puede escribir un motivo de pago como “Compra de materiales” o “Pago de nómina”, y esa entrada se almacenará en la variable motivo_pago para su posterior procesamiento.

Este tipo de interacción permite al usuario proporcionar información relevante al programa, que luego puede ser utilizada para realizar diversas acciones, como registrar transacciones, generar informes o realizar cálculos específicos en función del motivo de pago proporcionado.

archivo_final.insert(loc=2, column = ’Fecha de pago’, value= fecha_pago_corregida) archivo_final.insert(loc=5, column = ’Motivo’, value= motivo_pago)

Estas líneas de código están insertando nuevas columnas en el DataFrame archivo_final. Estas dos líneas deben interpretarse a partir de:

archivo_final.insert(loc=2, column=’Fecha de pago’, value=fecha_pago_corregida): Esta línea inserta una nueva columna llamada ’Fecha de pago’ en la posición 2 del DataFrame archivo_final. El valor de cada fila en esta columna será el valor de fecha_pago_corregida, definido anteriormente en el código a partir de lo que haya cargado el usuario a través el input respectivo. Esto significa que cada fila del DataFrame archivo_final tendrá la misma fecha de pago.

archivo_final.insert(loc=5, column=’Motivo’, value=motivo_pago): Esta línea inserta otra nueva columna llamada ’Motivo’ en la posición 5 del DataFrame archivo_final. El valor de cada fila en esta columna será el valor de motivo_pago, que es la entrada proporcionada por el usuario. Cada fila en la columna ’Motivo’ contendrá el mismo motivo de pago ingresado por el usuario.

Estas líneas de código son útiles para agregar información adicional a un DataFrame, como la fecha de pago y el motivo del pago, lo que puede ser importante para el análisis y la gestión de los datos. Sin embargo, ten en cuenta que si fecha_pago_corregida y motivo_pago no están definidos previamente en el código, necesitarás definirlos antes de usar estas líneas.

In [13]:
```
archivo_final= archivo_final.round(2)
archivo_final.head(3)
```

La línea de código archivo_final= archivo_final.round(2) le da formato a todos los valores numéricos en el DataFrame archivo_final redondeándolos a dos decimales.

round(2) es un método de pandas que se aplica a un DataFrame y redondea todos los valores numéricos a un número específico de decimales, en este caso, dos decimales.

Out[13]:

	Apellido y Nombre	CBU	CUIT	Remuneraciones a pagar
0	PEREZ MARTIN	4260010100200014270006	20309041810	288834.02
1	MARTINEZ PEDRO	4260010100200014258015	27293540379	288834.02
2	SALAS RAUL	0170210340000011351337	20266336978	299366.72
3	VORLA MARINA	0170210340000011351719	20348278909	289668.44
4	KLEMENT SAMANTA	0170210340000011352019	27360556285	243685.18

In [14]:
```
orden_columnas = ['CBU', 'Fecha de pago', 'Remuneraciones a pagar', 'CUIT',
    'Motivo']
archivo_final = archivo_final[orden_columnas]
```

Estas líneas de código reordenan las columnas del DataFrame “archivo_final” según el orden especificado en la lista “orden_columnas”.

En la línea 1: orden_columnas = [‘CBU’, ’Fecha de pago’, ‘Remuneraciones a pagar’, ‘CUIT’, ’Motivo’]: En esta línea, se crea una lista llamada ‘orden_columnas‘ que contiene los nombres de las columnas en el orden deseado. Este orden reflejará cómo se desea que las columnas aparezcan en el DataFrame final.

En la línea 2: archivo_final = archivo_final[orden_columnas]: En esta línea, se reasigna el DataFrame “archivo_final” de manera que las columnas se reordenan según el orden especificado en la lista “orden_columnas”. Esto se logra al pasar la lista “orden_columnas” como un índice de columnas al DataFrame, lo que fuerza al DataFrame a reorganizar las columnas en el orden especificado.

Después de ejecutar esta línea de código, las columnas del DataFrame “archivo_final” estarán organizadas en el orden especificado por la lista “orden_columnas”. Esto es útil para presentar los datos de manera más coherente o para cumplir con requisitos específicos de presentación o análisis.

In [15]:
```
archivo_final.dtypes
```

Out[15]:
```
CBU                          object
Fecha de pago                object
Remuneraciones a pagar      float64
CUIT                          int64
Motivo                       object
dtype: object
```

RESULTADO FINAL DE DF

Out[11]:

4260010100200014270006	05/03/2024	288833.020000	20309041810	HABERES
4260010100200014258015	05/03/2024	288833.020000	27293540379	HABERES
017021034000011351337	05/03/2024	299363.720000	20266336978	HABERES
017021034000011351719	05/03/2024	289668.440000	20348278909	HABERES
017021034000011352019	05/03/2024	243683.180000	27360556285	HABERES
4260010100200018445046	05/03/2024	291449.630000	27377017361	HABERES
017021034000011351955	05/03/2024	292368.930000	27356540641	HABERES
017021034000011056423	05/03/2024	444592.780000	20271482516	HABERES
017021034000011056287	05/03/2024	278611.890000	27345087872	HABERES
017021034000011351573	05/03/2024	280177.770000	27254678592	HABERES
017021034000011060037	05/03/2024	279822.400000	27371458110	HABERES
017021034000011056355	05/03/2024	279783.320000	27366276535	HABERES

In [16]:
```
suma=archivo_final['Remuneraciones a pagar'].sum()
print(f'Usted esta por generar un archivo por Pesos: {suma}')
```

Este fragmento de código calcula la suma total de la columna 'Remuneraciones a pagar' en el DataFrame "archivo_final" y luego imprime un mensaje indicando la cantidad total en la variable "suma".

Línea 1: archivo_final['Remuneraciones a pagar'].sum(): Esta parte del código calcula la suma de todos los valores en la columna 'Remuneraciones a pagar' del DataFrame 'archivo_final'.

Línea 2: (f'Usted esta por generar un archivo por Pesos: suma'): Esta línea imprime un mensaje en la consola. Utiliza un formato de cadena f-string para incrustar la variable 'suma' dentro del mensaje. El mensaje indica al usuario que está a punto de generar un archivo por una cierta cantidad de pesos, donde la cantidad es la suma total de la columna 'Remuneraciones a pagar'.

Por ejemplo, si la suma de los valores en la columna 'Remuneraciones a pagar' es 5000, el mensaje impreso sería: "Usted está por generar un archivo por Pesos: 5000".

Esta función es útil para brindar información al usuario sobre el total de remuneraciones que se están a punto de procesar o generar en un archivo, lo que puede ser importante para la contabilidad, presupuestación u otros fines financieros.

Out[16]:
```
Usted esta por generar un archivo por Pesos 3557198.0999999996
```

In [17]:
```
archivo_final.to_csv(f'C:/Transf_Sueldos_Banco_{periodo_liquidado}.csv',
   index=False, header= False, sep=';')
with open(f'C:/Transf_Sueldos_Banco_{periodo_liquidado}.csv') as f:
    lines = f.readlines()
    last = len(lines) - 1
    lines[last] = lines[last].replace('\r','').replace('\n','')
with open((f'C:/Transf_Sueldos_Banco_{periodo_liquidado}.csv'), 'w') as wr:
```

```
7     wr.writelines(lines)
```

Este bloque de código lleva a cabo las siguientes operaciones:

En la línea 1:

archivo_final.to_csv(f‘C:/Transf_Sueldos_Banco_periodo_liquidado.csv’, index=False, header= False, sep=‘;’): Este fragmento de código guarda el DataFrame ‘archivo_final‘ en un archivo CSV en la ruta especificada. Las instrucciones, ‘index=False‘ y ‘header=False‘ aseguran que no se guarden los índices de las filas y el encabezado respectivamente en el archivo CSV. La indicación sep=‘;’ define el separador como punto y coma en lugar del predeterminado, que es la coma.

En la línea 2:

Se abre el archivo recién creado (‘C:/Transf_Sueldos_Banco_periodo_liquidado.csv’) en modo lectura y se leen todas las líneas del archivo.

En la línea 3: Se modifica la última línea del archivo para eliminar cualquier caracter de retorno de carro (‘\r‘) y/o nueva línea (‘\n‘) que pueda estar presente.

En la línea 4: Finalmente, el archivo se vuelve a abrir en modo escritura y se escriben las líneas modificadas de nuevo al archivo, sobrescribiendo el contenido anterior.

Este conjunto de operaciones está destinado a asegurar que el archivo CSV resultante esté formateado correctamente, sin caracteres de retorno de carro o nuevas líneas adicionales al final del archivo. Esto es importante para evitar posibles problemas de lectura o procesamiento cuando se trabaja con el archivo CSV posteriormente.

In [18]:
```
1 nombre_original = (f'C://Transf_Sueldos_Banco_{periodo_liquidado}.csv')
2 nueva_extension = 'txt'
3 nombre_sin_extension, _ = os.path.splitext(nombre_original)
4 nuevo_nombre = f'{nombre_sin_extension}.{nueva_extension}'
5 os.rename(nombre_original, nuevo_nombre)
```

Este bloque de código en Python cambia la extensión de un archivo de CSV a TXT. Cada una de las líneas de código tienen como finalidad:

En la línea 1: nombre_original = (f‘C://Transf_Sueldos_Banco_periodo_liquidado.csv’): Define el nombre del archivo original con la extensión ‘.csv‘. La variable ‘periodo_liquidado‘ contiene la información del periodo para la cual se están transfiriendo los sueldos.

En la línea 2: nueva_extension = ‘txt’: Define la nueva extensión deseada para el archivo.

En la línea 3: nombre_sin_extension, _ = os.path.splitext(nombre_original): Divide el nombre del archivo y su extensión utilizando la función ‘os.path.splitext()‘. Esto devuelve una tupla donde el primer elemento es el nombre del archivo sin la extensión y el segundo elemento es la extensión del archivo.

En la línea 4: nuevo_nombre = f‘nombre_sin_extension.nueva_extension’: Combina el nombre del archivo sin la extensión con la nueva extensión para formar el nuevo nombre del archivo con la extensión deseada.

En la línea 5: os.rename(nombre_original, nuevo_nombre): Utiliza la función ‘os.rename()‘ para cambiar el nombre del archivo original al nuevo nombre con la nueva extensión.

Después de ejecutar este bloque de código, el archivo original CSV se renombrará con la extensión TXT. Esto es útil si necesitas cambiar el formato del archivo para que sea compatible con ciertas aplicaciones o procesos de tu sistema.

23.3. Ejercicio Práctico N° 3: Combinar archivos del sistema de información contable .CSV y exportarlos a Microsoft Excel®

Presentación del problema: Asuma que se le ha solicitado consolidar mensualmente los estados de resultados de tres unidades de negocios de la empresa a partir de los reportes del sistema de información contable.

SYSCAS202403 - Notepad

File Edit Format View Help

CUENTA	NOMBRE	CUEPAD	IMPUTA	SDOINI	DEBE	HABER
1	SIN IMPUTACION CONTABLE		1	0	2500,01	0
100000	ACTIVO	0	0	70851610,54	72868236,86	
110000	ACTIVO CORRIENTE	100000	0	0	59986191,3	72868236,86
111000	CAJA Y BANCOS	110000	0	0	35613491,99	38867399,18
111100	CAJA HOTEL CASINO	111000	0	0	1841822,5	1752242,5
111101	Caja Moneda Nacional	111100	1	0	1841822,5	1752242,5
111500	BANCOS HOTEL CASINO	111000	0	0	21719170,97	24796413,67
111501	BANCO BICA H CASTELAR 47501/2	111500	1	0	16732783,38	20098230,28
111505	BANCO BICA CTA CTE -3181008	111500	1	0	0	0,09
111510	Mercado Pago cta cte	111500	1	0	55000	47003,3
111511	BANCO BICA Cuenta en USD Castelar		111500	1	0	4931387,59
111900	FONDO FIJO HOTEL CASINO	111000	0	0	4350,02	2500,01
111901	Fondo Fijo	111900	1	0	4350,02	2500,01
111920	VALORES A DEPOSITAR HOTEL CASINO	111000	0	0	12048148,5	12316243
111921	Cheques a Depositar	111920	1	0	11915681,1	12183775,6
111923	Cheques Rechazados	111920	1	0	132467,4	132467,4
112000	INVERSIONES	110000	0	0	5931304,97	3300000
112100	INVERSIONES HOTEL CASINO	112000	0	0	5931304,97	3300000
112104	FCI Bica	112100	1	0	5931304,97	3300000
113000	CREDITOS	110000	0	0	17672703,14	30381512,67
113100	CUENTAS A COBRAR HOTEL CASINO		113000	0	0	15015316,61
113101	Deudores por Ventas Cta. Cte.		113100	1	0	2642554,83
113104	Tarjeta Mastercard a Cobrar		113100	1	0	2086487,88
113105	Tarjeta Maestro a Cobrar		113100	1	0	30400
113107	Tarjeta Visa Crédito a Cobrar		113100	1	0	2793836,72
113108	Tarjeta Visa Electrón a Cobrar		113100	1	0	2531823,98
113112	Depósitos y transferencias pendientes		113100	1	0	4455592,2
113123	QR	113100	1	0	474621	474621
113500	OTROS CREDITOS HOTEL CASINO		113000	0	0	545378,3
113502	Anticipos a Proveedores	113500	1	0	545378,3	
113506	Deudores Varios	113500	1	0	0	
113700	CREDITOS POR IMPUESTOS HOTEL CASINO		113000	0	0	2112008,23
113701	Ganancias Retencion TC	113700	1	0	7048,47	0
113703	Ganancias Retencion Clientes		113700	1	0	34924,11
113704	Anticipos I G M Presunta		113700	1	0	0

Figura 29: Visualizacion de archivo exportado

```
In [1]: 1 import pandas as  pd
        2 import os
        3 os.listdir('C:/Archivos CSV y XLSX')
        4 directory = ('C:/Archivos CSV y XLSX')
```

El fragmento de código precedente llama o importa la biblioteca pandas y luego utiliza la función os.listdir() para listar los archivos y directorios en el directorio especificado.

import pandas as pd: Esta línea importa la biblioteca pandas bajo el alias pd. Pandas es una biblioteca de Python ampliamente utilizada para manipulación y análisis de datos.

import os: Se importa el módulo os, que proporciona funciones para interactuar con el sistema operativo, como acceder al sistema de archivos.

os.listdir('C:/Archivos CSV y XLSX'): Esto llama a la función listdir() del módulo os para listar los archivos y directorios en el directorio especificado, que es 'C:/Archivos CSV y XLSX'. Sin embargo, en este código, el resultado de os.listdir() no se está almacenando en ninguna variable o no se está imprimiendo en la consola, por lo que no verás la lista de archivos y directorios.

directory = ('C:/Archivos CSV y XLSX'): Esto asigna la ruta del directorio 'C:/Archivos CSV y XLSX' a la variable directory, pero no hace nada más con ella en Este script de .

Para ver la lista de archivos y directorios en el directorio 'C:/Archivos CSV y XLSX', necesitas almacenar el resultado de os.listdir() en una variable y luego imprimir esa variable, o usarla de otra manera en tu código.

In [2]:
```
periodo = input('Ingrese el periodo con formato YYYYMM: ')
print (periodo)
```

Este script de en Python solicita al usuario que ingrese un periodo con el formato YYYYMM y luego imprime el periodo ingresado por el usuario.

En la línea 1: periodo = input('Ingrese el periodo con formato YYYYMM: '): Esta línea solicita al usuario que ingrese un periodo utilizando la función 'input()'. El mensaje contenido dentro de 'input()' le indica al usuario que el formato esperado es YYYYMM. La entrada del usuario se guarda en la variable 'periodo'.

En la línea 2: print(periodo): Esta línea imprime en la consola el valor almacenado en la variable 'periodo'. Esto muestra al usuario el periodo que ha ingresado.

Por ejemplo, si el usuario ingresa "202301" cuando se le solicita, el programa imprimirá "202301" en la consola.

Este tipo de interacción es útil para permitir que los usuarios ingresen información relevante que luego puede ser utilizada en el programa para realizar cálculos, generar informes, o llevar a cabo otras operaciones específicas.

Out[2]:
```
Ingrese el periodo con formato YYYYMM: 202403
202403
```

In [3]:
```
file_1 = f'\SYSCAS{periodo}.csv'
```

```
file_2 = f'\SYSRES{periodo}.csv'
file_3 = f'\SYSPOS{periodo}.csv'
encabezado_1 = file_1[4:13]
encabezado_2 = file_2[4:13]
encabezado_3 = file_3[4:13]
```

Este script de Python permite definir nombres de archivos basados en un período de año y mes dado y luego extrae parte de esos nombres para formar los encabezados del DataFrame. Se realiza uno por cada una de las unidades económicas analizadas, CASINO, RESORT, y LA POSADA, a partir del reporte de Sumas Y Saldos del sistema de gestión contable.

En la línea 1: file_1 = f‘\SYSCASperiodo}.csv’: Construye el nombre del primer archivo CSV basado en el periodo dado. En este caso, ‘periodo‘ es una variable que contiene el periodo ingresado por el usuario.

En la línea 2: file_2 = f‘\SYSRES{periodo}.csv’: Construye el nombre del segundo archivo CSV basado en el periodo dado.

En la línea 3: file_3 = f‘\SYSPOS{periodo}.csv’: Construye el nombre del tercer archivo CSV basado en el periodo dado.

En la línea 4: encabezado_1 = file_1[4:13]: Extrae una parte específica del nombre del primer archivo (‘file_1‘). En este caso, comienza en el cuarto caracter y termina en el décimo tercer caracter, creando así un encabezado.

En la línea 5: encabezado_2 = file_2[4:13]: Extrae una parte específica del nombre del segundo archivo (‘file_2‘) de la misma manera que el paso anterior.

En la línea 6: encabezado_3 = file_3[4:13]: Extrae una parte específica del nombre del tercer archivo (‘file_3‘) de la misma manera que los pasos anteriores.

Sintetizando, este código construye nombres de archivos CSV basados en un periodo proporcionado por el usuario y luego extrae una parte específica de esos nombres para usarlos como encabezados. Sin embargo, ten en cuenta que podría haber un error potencial si el periodo no tiene la longitud esperada o si el formato del nombre del archivo no es consistente.

```
In [4]: print (encabezado_1, encabezado_2, encabezado_3)
```

```
Out[4]: CAS202403 RES202403 POS202403
```

```
In [5]: with open (directory+file_1, newline='') as f:
            archivo_1=pd.read_csv(f,sep='\t', decimal=',')
        with open (directory+file_2, newline='') as f:
            archivo_2=pd.read_csv(f,sep='\t', decimal=',')
        with open (directory+file_3, newline='') as f:
```

```
    archivo_3=pd.read_csv(f,sep='\t', decimal=',')
archivo_1.head()
archivo_1.dtypes
```

este código lee tres archivos CSV diferentes usando pandas, asumiendo que están separados por tabulaciones y que los decimales están representados por comas. Luego, muestra las primeras filas y los tipos de datos del primer DataFrame. Sin embargo, ten en cuenta que el valor de directory debe estar correctamente definido previamente para que la lectura de archivos sea exitosa.

Out[5]:
```
CUENTA           int64
NOMBRE          object
CUEPAD         float64
IMPUTA           int64
SDOINI           int64
DEBE           float64
HABER          float64
Unnamed: 7     float64
dtype: object
```

In [6]:
```
archivo_1 = archivo_1[archivo_1['IMPUTA'] !=0]
archivo_1[encabezado_1] = archivo_1['HABER']-archivo_1['DEBE']
archivo_1 = archivo_1 [archivo_1['CUENTA']>399999]
archivo_1 = archivo_1.fillna(0)
archivo_1.head()
```

En la línea 1: archivo_1 = archivo_1[archivo_1['IMPUTA'] !=0]: - Esta línea filtra el DataFrame 'archivo_1' eliminando todas las filas donde el valor de la columna 'IMPUTA' es igual a 0. - La expresión 'archivo_1['IMPUTA'] != 0' devuelve una serie de booleanos que indica True para las filas donde 'IMPUTA' es diferente de 0, y False para las filas donde 'IMPUTA' es igual a 0. - Al colocar esta expresión dentro de 'archivo_1[]', se filtran las filas del DataFrame 'archivo_1' que cumplen con la condición.

En la línea 2: archivo_1[encabezado_1] = archivo_1['HABER'] - archivo_1['DEBE']: - Esta línea calcula una nueva columna en el DataFrame 'archivo_1' que se corresponde con el encabezado extraído anteriormente ('encabezado_1'). - El cálculo se realiza restando el valor de la columna 'DEBE' del valor de la columna 'HABER' para cada fila del DataFrame 'archivo_1'. - La nueva columna se asigna a 'archivo_1[encabezado_1]'.

En la línea 3: archivo_1 = archivo_1[archivo_1['CUENTA'] 399999]: - Esta línea filtra el DataFrame 'archivo_1' eliminando todas las filas donde el valor de la columna 'CUENTA' es menor o igual a 399999. - Al igual que en el primer caso, se utiliza una expresión booleana para seleccionar las filas que cumplen con la condición de que 'CUENTA' sea mayor a 399999.

En la línea 4: archivo_1 = archivo_1.fillna(0): - Esta línea rellena los valores faltantes

(NaN) en el DataFrame ‘archivo_1‘ con el valor 0. - Esto significa que si hay celdas vacías en el DataFrame, serán reemplazadas por 0.

En síntesis, estas líneas de código realizan una serie de transformaciones en el DataFrame ‘archivo_1‘, incluyendo filtrar filas, calcular nuevas columnas, eliminar valores faltantes y rellenarlos con ceros. Estas operaciones son comunes en la manipulación y limpieza de datos para prepararlos para análisis o visualización.

Out[6]:

	CUENTA	NOMBRE	CUEPAD	IMPUTA	SDOINI	DEBE	HABER	Unnamed: 7	CAS202403
177	411101	Alojamiento	411100.0	1	0	355544.94	8296871.79	0.0	7941326.85
178	411103	No Shows	411100.0	1	0	0.00	49132.23	0.0	49132.23
179	411104	Late Check Out	411100.0	1	0	0.00	120690.09	0.0	120690.09
180	411105	Early Check In	411100.0	1	0	9586.78	114400.82	0.0	104814.04
181	411110	Estacionamiento	411100.0	1	0	7438.02	169834.80	0.0	162396.78

In [7]:
```
1 archivo_1 = archivo_1[['NOMBRE',encabezado_1]]
2 archivo_1 = archivo_1[archivo_1[encabezado_1] !=0]
3 archivo_1.tail()
```

Estas líneas de código realizan operaciones adicionales en el DataFrame ‘archivo_1‘. Cada línea se explica a partir de::

En la línea 1: archivo_1 = archivo_1[[’NOMBRE’, encabezado_1]]: - Esta línea selecciona solo las columnas ’NOMBRE’ y ‘encabezado_1‘ del DataFrame ‘archivo_1‘. - El resultado es un nuevo DataFrame que contiene solo estas dos columnas.

En la línea 2: archivo_1 = archivo_1[archivo_1[encabezado_1] != 0]: - Esta línea filtra las filas del DataFrame ‘archivo_1‘ donde el valor en la columna ‘encabezado_1‘ es diferente de 0. - La expresión ‘archivo_1[encabezado_1] != 0‘ devuelve una serie de booleanos que indica True para las filas donde el valor en la columna ‘encabezado_1‘ es diferente de 0, y False para las filas donde el valor es igual a 0. - Al colocar esta expresión dentro de ‘archivo_1[]‘, se filtran las filas del DataFrame ‘archivo_1‘ que cumplen con la condición.

En síntesis, después de ejecutar estas líneas de código, ‘archivo_1‘ contendrá solo las columnas ’NOMBRE’ y ‘encabezado_1‘, y se habrán eliminado las filas donde el valor en la columna ‘encabezado_1‘ es igual a 0. Estas operaciones, muy potentes a partir de la utilización de la librería pandas, están relacionadas con la manipulación y limpieza de los datos del DataFrame ‘archivo_1‘.

Out[7]:

	NOMBRE	CAS202403
177	Alojamiento	7941326.85
178	No Shows	49132.23
179	Late Check Out	120690.09
180	Early Check In	104814.04
181	Estacionamiento	162396.78

In [8]:
```
archivo_2 = archivo_2[archivo_2['IMPUTA'] !=0]
archivo_2[encabezado_2] = archivo_2['HABER']-archivo_2['DEBE']
archivo_2 = archivo_2 [archivo_2['CUENTA']>399999]
archivo_2 = archivo_2.fillna(0)
archivo_2= archivo_2[['NOMBRE',encabezado_2]]
archivo_2 = archivo_2[archivo_2[encabezado_2] !=0]
archivo_2.tail()
```

Out[8]:

	NOMBRE	RES202403
237	Mantenimiento de Software y Hardware	-161352.85
239	Mantenimiento Eléctrico	-49696.72
242	Mantenimiento Piscina	-86329.38
244	Fletes	-1131.99
247	Renta Fija Propietarios	-847934.41

In [9]:
```
archivo_3 = archivo_3[archivo_3['IMPUTA'] !=0]
archivo_3[encabezado_3] = archivo_3['HABER']-archivo_3['DEBE']
archivo_3 = archivo_3 [archivo_3['CUENTA']>399999]
archivo_3 = archivo_3.fillna(0)
archivo_3 = archivo_3[['NOMBRE',encabezado_3]]
archivo_3 = archivo_3[archivo_3[encabezado_3] !=0]
archivo_3.info()
```

Out[9]:
```
<class 'pandas.core.frame.DataFrame'>
Index: 21 entries, 104 to 156
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   NOMBRE      21 non-null     object
 1   POS202403   21 non-null     float64
```

```
dtypes: float64(1), object(1)
memory usage: 503.0+ bytes
```

In [10]:
```
comparativo = pd.merge(archivo_1, archivo_2, on='NOMBRE', how='outer')
comparativo\_2 = pd.merge(comparativo, archivo_3, on='NOMBRE', how='outer')
comparativo\_2 = comparativo\_2.fillna(0)
comparativo\_2.head(11)
```

Estas líneas de código realizan operaciones de fusión (merge) entre varios DataFrames y luego rellenan los valores faltantes con ceros. Se brinda una explicación de cada línea:

En la línea 1: comparativo = pd.merge(archivo_1, archivo_2, on='NOMBRE', how='outer'): - Esta línea fusiona los DataFrames 'archivo_1' y 'archivo_2' utilizando la columna 'NOMBRE' que se repite en ambos DataFrames como clave de fusión. - El parámetro 'how='outer'' indica que se realizará una unión externa, lo que significa que se conservarán todas las filas de ambos DataFrames, y se llenarán los valores faltantes con NaN en caso de que no haya coincidencias. - El resultado se almacena en el DataFrame denominado 'comparativo'.

En la línea 2: comparativo_2 = pd.merge(comparativo, archivo_3, on='NOMBRE', how='outer'): - Esta línea fusiona el DataFrame 'comparativo' (que ya contiene la fusión de 'archivo_1' y 'archivo_2') con el DataFrame 'archivo_3', nuevamente utilizando la columna 'NOMBRE' como clave de fusión y una unión externa. - El resultado se almacena en el DataFrame 'comparativo_2'.

En la línea 3: comparativo_2 = comparativo_2.fillna(0): - Esta línea rellena los valores faltantes en el DataFrame 'comparativo_2' con ceros. - Se usa 'fillna(0)' para reemplazar todos los valores NaN en el DataFrame 'comparativo_2' con ceros.

Resumiendo, estas líneas de código construyen un DataFrame 'comparativo_2' que contiene datos de los DataFrames 'archivo_1', 'archivo_2' y 'archivo_3', utilizando la columna 'NOMBRE' como clave de fusión. Luego, rellena los valores faltantes con ceros para asegurar que el DataFrame esté completamente poblado y sea consistente para el análisis posterior.

Out[10]:

	NOMBRE	CAS202403	RES202403	POS202403
0	Alojamiento	7941326.85	34650881.21	3564043.56
1	No Shows	49132.23	45041.33	0.00
2	Late Check Out	120690.09	374063.70	0.00
3	Early Check In	104814.04	74297.52	0.00
4	Estacionamiento	162396.78	304133.42	0.00
5	Alquiler Salón Auditorio	539014.57	0.00	0.00
6	Spa y Piscina	0.00	0.00	0.00
7	Otros Ingresos	133650.59	139008.26	1243726.90
8	Frigobar	44214.86	89834.60	0.00
9	Bar/Restaurante	0.00	54685.95	0.00
10	Descuentos Obtenidos	2193.00	2879.88	0.00

In [11]:
```
comparativo\_2['TOTAL'] = comparativo\_2[encabezado_1]+ comparativo\_2[
    encabezado_2] + comparativo\_2[encabezado_3]
comparativo\_2.head(5)
```

Esta línea de código agrega una nueva columna llamada 'TOTAL' al DataFrame 'comparativo_2', que obtenrá la suma de los valores en las columnas 'encabezado_1', 'encabezado_2' y 'encabezado_3' para cada fila.

comparativo_2['TOTAL'] = comparativo_2[encabezado_1] comparativo_2[encabezado_2] comparativo_2[encabezado_3]:

'comparativo_2[encabezado_1]', 'comparativo_2[encabezado_2]' y 'comparativo_2[encabezado_3]' acceden a las columnas del DataFrame 'comparativo_2' que contienen los valores que quieres sumar.

- Sumas estos valores columna por columna para cada fila del DataFrame 'comparativo_2'.

- El resultado de esta operación se almacena en la nueva columna 'TOTAL' del DataFrame 'comparativo_2'.

En resumen, esta línea de código calcula la suma total de los valores en las columnas 'encabezado_1', 'encabezado_2' y 'encabezado_3' para cada fila del DataFrame 'comparativo_2' y los almacena en la columna 'TOTAL', proporcionando así un resumen consolidado de los datos para cada fila.

Out[11]:

	NOMBRE	CAS202403	RES202403	POS202403	TOTAL
0	Alojamiento	7941326.85	34650881.21	3564043.56	46156251.62
1	No Shows	49132.23	45041.33	0.00	94173.56
2	Late Check Out	120690.09	374063.70	0.00	494753.79
3	Early Check In	104814.04	74297.52	0.00	179111.56
4	Estacionamiento	162396.78	304133.42	0.00	466530.20

In [12]:
```
comparativo\_2['NOMBRE'] = comparativo\_2['NOMBRE'].str.lstrip('          ')
```

Esta línea de código elimina los espacios en blanco iniciales en los valores de la columna 'NOMBRE' del DataFrame 'comparativo_2'.

- 'comparativo_2['NOMBRE'] = comparativo_2['NOMBRE'].str.lstrip(' '): - 'comparativo_2['NOMBRE']' accede a la columna 'NOMBRE' del DataFrame 'comparativo_2'. - '.str.lstrip(' ')' aplica el método 'lstrip()' a cada valor de la columna 'NOMBRE'. El método 'lstrip()' elimina los caracteres especificados (en este caso, espacios en blanco) del lado izquierdo de la cadena. - En este caso, se eliminan todos los espacios en blanco iniciales de cada valor de la columna 'NOMBRE'. - El resultado de esta operación se asigna nuevamente a la columna 'NOMBRE' del DataFrame 'comparativo_2'.

Esto es útil para limpiar y normalizar los datos, especialmente cuando hay inconsistencias en la forma en que se almacenan los datos, como espacios en blanco adicionales al comienzo o al final de los valores de texto. La eliminación de estos espacios en blanco adicionales ayuda a mantener la coherencia en los datos y facilita su procesamiento y análisis.

In [13]:
```
df_sorted = pd.concat([
    comparativo\_2[comparativo\_2['TOTAL'] > 0].sort_values('TOTAL',
   ascending=False),
    comparativo\_2[comparativo\_2['TOTAL'] <= 0].sort_values('TOTAL')
])
df_sorted.head(20)
```

Este bloque de código de Python crea un nuevo DataFrame llamado 'df_sorted' combinando dos subconjuntos de datos de 'comparativo_2'.

En la línea 1: comparativo_2[comparativo_2['TOTAL'] 0].sort_values('TOTAL', ascending=False): Este segmento selecciona todas las filas en 'comparativo_2' donde el valor en la columna 'TOTAL' es mayor que 0. Luego, ordena estas filas según los valores en la columna 'TOTAL' en orden descendente (de mayor a menor).

En la línea 2: comparativo_2[comparativo_2['TOTAL'] 0].sort_values('TOTAL'): Este segmento selecciona todas las filas en 'comparativo_2' donde el valor en la columna 'TOTAL' es menor o igual a 0. Luego, ordena estas filas según los valores en la columna 'TOTAL' en orden ascendente (de menor a mayor).

En la línea 3: pd.concat([...]): Este es el método concat de pandas que combina los dos subconjuntos de datos creados anteriormente en un solo DataFrame. Esto se hace proporcionando una lista de los subconjuntos de datos a concatenar.

En la línea 4: El resultado final es un DataFrame llamado 'df_sorted', que contiene todas las filas de 'comparativo_2' ordenadas primero por aquellas con valores positivos en la columna 'TOTAL' (en orden descendente) y luego por aquellas con valores no positivos (en orden ascendente).

En síntesis, este bloque de código ordena el DataFrame 'comparativo_2' según los valores en la columna 'TOTAL', con las filas que tienen valores positivos primero y las filas que tienen valores no positivos después.

Out[13]:

	NOMBRE	CAS202403	RES202403	POS202403	TOTAL
0	Alojamiento	7941326.85	34650881.21	3564043.56	46156251.62
11	Ganancias Financieras	131304.97	1543399.42	1966096.23	3640800.62
7	Otros Ingresos	133650.59	139008.26	1243726.90	1516385.75
92	Alojamiento Corporativos	0.00	0.00	557713.88	557713.88
5	Alquiler Salón Auditorio	539014.57	0.00	0.00	539014.57
2	Late Check Out	120690.09	374063.70	0.00	494753.79
4	Estacionamiento	162396.78	304133.42	0.00	466530.20
3	Early Check In	104814.04	74297.52	0.00	179111.56
8	Frigobar	44214.86	89834.60	0.00	134049.46
1	No Shows	49132.23	45041.33	0.00	94173.56
84	Alquiler Oficina	0.00	82644.62	0.00	82644.62
83	Alquiler SUM	0.00	70247.92	0.00	70247.92
9	Bar/Restaurante	0.00	54685.95	0.00	54685.95
82	Lavanderia	0.00	6611.57	0.00	6611.57
10	Descuentos Obtenidos	2193.00	2879.88	0.00	5072.88
15	Sueldos y Jornales	-5475730.22	-5619709.08	-466358.58	-11561797.88
16	Cargas Sociales	-1378872.26	-1458485.52	-117759.08	-2955116.86
65	Energia Electrica	-616714.33	-735658.13	-795937.51	-2148309.97
55	Impuesto a los Ingresos Brutos	0.00	-1495269.09	-241446.80	-1736715.89
33	Servicios Profesionales	-276742.55	-766665.20	-449503.38	-1492911.13

In [14]:
```
def color_negative_red(val):
    color = 'red' if val < 0 else 'green'
```

```
    return 'color: %s' % color
df_sorted_coloured = df_sorted.style.applymap(color_negative_red, subset=['
    TOTAL']).format()
```

La función ‘color_negative_red(val)‘ es una función que toma un valor ‘val‘ y devuelve un estilo CSS que cambiará el color del texto en función del valor. Si el valor es negativo, el texto será rojo; de lo contrario, será verde.

- ‘color = ’red’ if val 0 else ’green’: Esto establece el color en ’red’ si el valor es negativo y en ’green’ si el valor es igual o mayor que cero.

- ‘return ’color: %s’ % color: Esto devuelve una cadena de estilo CSS con el color calculado.

Luego, se aplica una función de estilo a un DataFrame utilizando el método ‘applymap()‘ de pandas. ‘applymap()‘ aplica una función a cada elemento del DataFrame. En este caso, aplica la función de color a la columna ’TOTAL’ del DataFrame ‘df_sorted‘.

Finalmente, se logra dar formato al DataFrame utilizando el método ‘format()‘ para asegurarte de que el estilo se aplique correctamente.

En resumen, esta función y su aplicación a través de ‘applymap()‘ permite cambiar el color del texto en la columna ’TOTAL’ del DataFrame ‘df_sorted‘ según si los valores son negativos o no. Esto es útil para resaltar visualmente ciertos aspectos de los datos cuando se presentan en un formato tabular, como un DataFrame pandas.

In [15]:
```
df_sorted_coloured.to_excel(directory + f'/{periodo}hoteles.xlsx', index=
    False)
```

Este código toma el DataFrame formateado ‘df_sorted_coloured‘ y lo guarda como un archivo Excel en la ubicación especificada.

Una explicación detallada de esta entrada N° 15 es:

- ‘df_sorted_coloured.to_excel(directory + f‘/periodohoteles.xlsx’, index=False):
- ‘df_sorted_coloured‘ es el DataFrame que se formateó previamente y al que se le aplicaron estilos.
- ‘to_excel()‘ es un método de Pandas que permite escribir el contenido de un DataFrame en un archivo Excel.
- ‘directory + f‘/periodohoteles.xlsx’: Especifica la ruta completa y el nombre del archivo

Excel resultante. Se usa el valor de ‘directory‘ (el directorio donde se desea guardar el archivo) y ‘periodo‘ para nombrar el archivo Excel.

- ‘index=False‘ indica que no se debe incluir el índice del DataFrame en el archivo Excel. En general, si no se necesita conservar el índice del DataFrame como una columna en el archivo Excel, es buena práctica establecer ‘index=False‘.

En síntesis, esta línea de código guarda el DataFrame formateado ‘df_sorted_coloured‘ en un archivo Excel en la ubicación y con el nombre especificados. Esto te permite guardar los datos formateados para su posterior análisis o presentación en un formato ampliamente compatible como Excel.

23.4. Ejercicio Práctico N° 4: Análisis de datos y gráficos desde archivos del sistema de información contable .CSV

Presentación del problema: Analice la información de un reporte de ventas anual y haga la limpieza de la base de datos y analice información útil para la toma de decisiones generando los gráficos que considere pertinente.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Este bloque de código importa las bibliotecas necesarias: pandas para la manipulación de datos, numpy para cálculos numéricos y matplotlib.pyplot para trazar gráficos. Las funcionalidades de cada una de las bibliotecas importadas en la entrada N° 1 a utilizar son:

En la línea 1: import pandas as pd: Esto importa la biblioteca pandas con el alias ‘pd‘. Pandas es una biblioteca de Python utilizada para el análisis y la manipulación de datos tabulares.

En la línea 2: import numpy as np: Esto importa la biblioteca numpy con el alias ‘np‘. Numpy es una biblioteca de Python utilizada para realizar cálculos numéricos en matrices y matrices multidimensionales.

En la línea 3: import matplotlib.pyplot as plt: Esto importa el módulo pyplot de la biblioteca matplotlib con el alias ‘plt‘. Matplotlib es una biblioteca de trazado para Python que produce gráficos de alta calidad en varios formatos y entornos.

Con estas importaciones, el código está listo para realizar operaciones de manipulación de datos con pandas, cálculos numéricos con numpy y trazado de gráficos con matplotlib.

In [2]:
```
with open ('tmprptRPTFORVTA.CSV', newline='') as f:
    df=pd.read_csv(f,sep='\t', decimal=',')
df.head()
```

El bloque de código precedente abre un archivo CSV llamado ’tmprptRPTFORVTA.CSV’, lee su contenido utilizando pandas y luego muestra las primeras filas del DataFrame resultante.

Cada línea puede ser interpretada como:

En la línea 1: ‘with open(’tmprptRPTFORVTA.CSV’, newline=”) as f: Este código abre el archivo CSV llamado ’tmprptRPTFORVTA.CSV’ en modo lectura (‘’r‘‘) utilizando un contexto ‘with‘. El parámetro ‘newline=”‘ se utiliza para asegurarse de que no se interpreten incorrectamente las nuevas líneas en el archivo CSV.

En la línea 2: ‘df = pd.read_csv(f, sep=‘‘, decimal=’,’): Utilizando pandas, se lee el contenido del archivo CSV abierto (representado por ‘f‘) y se carga en un DataFrame llamado ‘df’. Se especifica que el separador de campos en el archivo CSV es un tabulador (sep=‘‘) y que el separador decimal es una coma (‘decimal=’,’‘).

En la línea 3: df.head(): Este método muestra las primeras filas del DataFrame ‘df’. Por defecto, muestra las primeras 5 filas si no se especifica un número diferente.

En síntesis, este bloque de código carga un archivo CSV en un DataFrame de pandas llamado ‘df’ y luego muestra las primeras filas del DataFrame para que puedas echar un vistazo inicial a los datos y su estructura.

Out[2]:

	FCHMOV	CODFOR	NROFOR	CODORI	NOMBRE	NROHAB	NROFOL	FORPAG	CODMON	IMPNAC	...	MODITM	CODANU	NROANU	USRALT	
0	01/01/2023	FCA012	5064	A 0012-00004817	MENDELER, SILVIA	105.0	68813.0	CDO	ARS	65416.03	...	0	NaN	NaN	USUARIO1	0
1	01/01/2023	FCB012	20387	B 0012-00015869	SINNER, ORLANDO	108.0	68820.0	CDO	ARS	24000.00	...	R	NaN	NaN	USUARIO3	0
2	01/01/2023	FCB012	20388	NaN	LEAO CARVALHO, JUAN	214.0	68827.0	CDO	ARS	0.00	...	R	NaN	NaN	USUARIO2	0
3	01/01/2023	FCB012	20389	NaN	TINDOR, LAURA	207.0	68828.0	CDO	ARS	0.00	...	R	NaN	NaN	USUARIO2	0
4	01/01/2023	FCB012	20390	B 0012-00015870	SUSE DURO, MATIAS	110.0	68836.0	CDO	ARS	17300.00	...	R	NaN	NaN	USUARIO2	0

In [3]:
```
df.info()
```

Out[3]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11731 entries, 0 to 11730
Data columns (total 23 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   FCHMOV       11731 non-null  object
 1   CODFOR       11731 non-null  object
 2   NROFOR       11731 non-null  int64
 3   CODORI       9527 non-null   object
 4   NOMBRE       11731 non-null  object
 5   NROHAB       8370 non-null   float64
 6   NROFOL       8424 non-null   float64
 7   FORPAG       11731 non-null  object
 8   CODMON       11731 non-null  object
 9   IMPNAC       11731 non-null  float64
 10  IMPDOL       11731 non-null  int64
 11  IMPEXT       11731 non-null  int64
 12  CAMBIO       11731 non-null  int64
 13  MODITM       11731 non-null  object
 14  CODANU       0 non-null      float64
 15  NROANU       483 non-null    float64
 16  USRALT       11731 non-null  object
 17  FCHALT       11731 non-null  object
 18  CODCLI       11731 non-null  object
 19  SUCCLI       11731 non-null  int64
 20  EMPRESA      11731 non-null  object
 21  TIPCOM       11731 non-null  object
 22  Unnamed: 22  0 non-null      float64
dtypes: float64(6), int64(5), object(12)
memory usage: 2.1+ MB
```

```
In [4]: 1 columnas_a_eliminar = [0,1,2,6,8,10,11,12,13,14,15,16,18,19,20,21,22]
        2 df = df.drop(df.columns[columnas_a_eliminar], axis=1)
        3 df.head(5)
```

El código previamente expuesto elimina columnas específicas del DataFrame ‘df’ basado en las posiciones de las columnas indicadas en la lista ‘columnas_a_eliminar‘. Aquí está la explicación de cada línea:

En la línea 1: columnas_a_eliminar = [0,1,2,6,8,10,11,12,13,14,15,16,18,19,20,21,22]: Esta línea define una lista de índices de columnas que se van a eliminar del DataFrame ‘df’. Los índices de las columnas comienzan desde 0. En la línea 2:

df = df.drop(df.columns[columnas_a_eliminar], axis=1): Utilizando el método ‘drop()‘ de pandas, se eliminan las columnas especificadas en ‘columnas_a_eliminar‘. Internamente en esta línea df.columns[columnas_a_eliminar] selecciona las columnas del DataFrame ‘df’ basado en los índices proporcionados en ‘columnas_a_eliminar‘; y el parámetro ‘axis=1‘ indica que

las operaciones se realizan a lo largo de las columnas. El resultado se asigna nuevamente al DataFrame 'df', por lo que 'df' ahora contiene solo las columnas que no fueron eliminadas. En la línea 3: df.head(5) permite mostrar en pantalla el encabezado del DataFrame 'df' hasta sus primeros cinco registros.

En síntesis, este bloque de código elimina las columnas especificadas del DataFrame 'df' y conserva las columnas restantes para su posterior análisis o procesamiento. Las columnas se eliminan según sus índices de posición en el DataFrame.

Out[4]:

	CODORI	NOMBRE	NROHAB	FORPAG	IMPNAC	FCHALT
0	A 0012-00004817	MENDELER, SILVIA	105.0	CDO	65416.03	01/01/2023 22:56
1	B 0012-00015869	SINNER, ORLANDO	108.0	CDO	24000.00	01/01/2023 05:56
2	NaN	LEAO CARVALHO, JUAN	214.0	CDO	0.00	01/01/2023 08:02
3	NaN	TINDOR, LAURA	207.0	CDO	0.00	01/01/2023 09:43
4	B 0012-00015870	SUSE DURO, MATIAS	110.0	CDO	17300.00	01/01/2023 09:44

In [5]:
```
df.rename(columns={
    'CODORI': 'Nro_Comprobante',
    'NOMBRE': 'Cliente',
    'IMPNAC': 'Importe',
    'FORPAG': 'Forma_Pago',
    'NROHAB': 'Habitacion',
    'FCHALT': 'Fecha'}
        , inplace=True)
df.fillna(0, inplace=True)
df.tail()
```

Con estas instrucciones se realizan dos operaciones en el DataFrame 'df': cambia los nombres de algunas columnas y rellena los valores faltantes con ceros. Se expone la explicación de cada línea:

En la línea 1: df.rename(columns=..., inplace=True): Esta línea cambia los nombres de las columnas del DataFrame 'df' según el diccionario proporcionado. De aquí en mas:

- La columna 'CODORI' se pasa a llamar 'Nro_Comprobante'.
- La columna 'NOMBRE' se pasa a llamar 'Cliente'.

- La columna ‘IMPNAC’ se pasa a llamar ‘Importe’.
- La columna ‘FORPAG’ se pasa a llamar ‘Forma_Pago’.
- La columna ‘NROHAB’ se pasa a llamar ‘Habitacion’.
- La columna ‘FCHALT’ se pasa a llamar ‘Fecha’, y la instrucción
- ‘inplace=True’ indica que los cambios deben hacerse directamente en el DataFrame ‘df’, en lugar de devolver un nuevo DataFrame con los cambios aplicados.

En la línea 2: df.fillna(0, inplace=True): Esta línea rellena los valores faltantes (NaN) en el DataFrame ‘df’ con ceros. La sintáxis entre paréntesis se comprende mediante: - ‘0’ es el valor con el que se llenarán los valores faltantes. - ‘inplace=True’ indica que los cambios deben hacerse directamente en el DataFrame ‘df’.

Resumiendo, este bloque de código renombra algunas columnas importantes del DataFrame ‘df’ para hacerlas más descriptivas y luego rellena los valores faltantes con ceros para asegurarse de que el DataFrame esté completo y listo para el análisis posterior.

Out[5]:

	CODORI	NOMBRE	NROHAB	FORPAG	IMPNAC	FCHALT
0	A 0012-00004817	MENDELER, SILVIA	105.0	CDO	65416.03	01/01/2023 22:56
1	B 0012-00015869	SINNER, ORLANDO	108.0	CDO	24000.00	01/01/2023 05:56
2	NaN	LEAO CARVALHO, JUAN	214.0	CDO	0.00	01/01/2023 08:02
3	NaN	TINDOR, LAURA	207.0	CDO	0.00	01/01/2023 09:43
4	B 0012-00015870	SUSE DURO, MATIAS	110.0	CDO	17300.00	01/01/2023 09:44

In [6]:
```
1 df.Habitacion = df.Habitacion.astype(int)
2 df['Fecha'] = pd.to_datetime(df['Fecha'],format='%d/%m/%Y %H:%M')
3 df.info()
```

Con este código se realizan dos transformaciones en el DataFrame ‘df’:

En la línea 1: df.Habitacion = df.Habitacion.astype(int): Esta línea convierte la columna ‘Habitacion’ del DataFrame ‘df’ a tipo de datos entero (‘int‘). Esto es útil si los números de habitación son valores enteros y necesitas realizar cálculos numéricos con ellos.

En la línea 2: df['Fecha'] = pd.to_datetime(df['Fecha'], format=' %d/ %m/ %Y %H: %M'}): Esta línea convierte la columna 'Fecha' del DataFrame 'df' a tipo de datos datetime utilizando la función 'pd.to_datetime()' de Pandas. - 'pd.to_datetime() convierte objetos tipo string en objetos tipo datetime. - 'format=' %d/ %m/ %Y %H: %M'' especifica el formato en el que están presentes las fechas en la columna 'Fecha'. En este caso, el formato es día/mes/año hora:minuto. - Después de la conversión, la columna 'Fecha' contiene objetos datetime, lo que permite realizar operaciones de fecha y hora.

Estas transformaciones son útiles para preparar los datos para su análisis posterior, especialmente cuando se necesita trabajar con fechas y números enteros en el DataFrame.

Out[6]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11731 entries, 0 to 11730
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Nro_Comprobante  11731 non-null  object
 1   Cliente         11731 non-null  object
 2   Habitacion      11731 non-null  int32
 3   Forma_Pago      11731 non-null  object
 4   Importe         11731 non-null  float64
 5   Fecha           11731 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int32(1), object(3)
memory usage: 504.2+ KB
```

In [7]:
```
df['DiaSemana'] = df['Fecha'].dt.dayofweek
dias_semana = {0: 'Lunes',1: 'Martes', 2: 'Miercoles', 3: 'Jueves', 4: '
    Viernes',5: 'Sabado',6: 'Domingo'}
df['DiaSemana'] = df['DiaSemana'].map(dias_semana)
print(df)
```

El código arriba expuesto agrega una nueva columna llamada 'DiaSemana' al DataFrame 'df', que representa el día de la semana correspondiente a la fecha en la columna 'Fecha'. Se detalla, a continuación, la explicación de cada línea:

En la línea 1: df['DiaSemana'] = df['Fecha'].dt.dayofweek: Esta línea utiliza el atributo 'dt.dayofweek' de la serie datetime 'df['Fecha']' para obtener el día de la semana como un número entero, donde 0 es lunes y 6 es domingo. Este número se asigna a la nueva columna 'DiaSemana'.

En la línea 2: dias_semana = 0: 'Lunes', 1: 'Martes', 2: 'Miércoles', 3: 'Jueves', 4: 'Viernes', 5: 'Sábado', 6: 'Domingo': Este diccionario mapea los números del 0 al 6 a los nombres de los días de la semana correspondientes.

En la línea 3: df['DiaSemana'] = df['DiaSemana'].map(dias_semana): Esta línea utiliza el método 'map()' para mapear los valores numéricos en la columna 'DiaSemana' a los nombres de los días de la semana utilizando el diccionario 'dias_semana'. Esto reemplaza los números del 0 al 6 con los nombres de los días de la semana correspondientes.

En síntesis, estas líneas de código agregan una nueva columna al DataFrame ‘df’ que muestra el nombre del día de la semana correspondiente a la fecha en la columna ’Fecha’, lo que puede ser útil para el análisis y la visualización de los datos.

In [8]:
```
df['Año'] = df['Fecha'].dt.year
df['Mes'] = df['Fecha'].dt.month
df['Dia'] = df['Fecha'].dt.day
df['Hora'] = df['Fecha'].dt.hour
df['Minuto'] = df['Fecha'].dt.minute
df.head(5)
```

Out[8]:

	CODORI	NOMBRE	NROHAB	FORPAG	IMPNAC	FCHALT
0	A 0012-00004817	MENDELER, SILVIA	105.0	CDO	65416.03	01/01/2023 22:56
1	B 0012-00015869	SINNER, ORLANDO	108.0	CDO	24000.00	01/01/2023 05:56
2	NaN	LEAO CARVALHO, JUAN	214.0	CDO	0.00	01/01/2023 08:02
3	NaN	TINDOR, LAURA	207.0	CDO	0.00	01/01/2023 09:43
4	B 0012-00015870	SUSE DURO, MATIAS	110.0	CDO	17300.00	01/01/2023 09:44

In [9]:
```
horas = [hora for hora, df in df.groupby('Hora')]
plt.plot (horas, df.groupby(['Hora']).count())
plt.grid()
plt.xticks(horas);
```

El código bajo expuesto crea un gráfico de líneas que muestra la distribución de los datos según la hora del día. Se expone la explicación de cada una de las líneas antecedentes:

- En la línea 1: horas = [hora for hora, df in df.groupby(‘Hora’)]: Esta línea crea una lista llamada ‘horas‘ que contiene todas las horas únicas presentes en la columna ‘Hora’ del DataFrame ‘df’. Se hace esto mediante la técnica de comprensión de listas junto con ‘groupby()‘.

- En la línea 2: plt.plot(horas, df.groupby([‘Hora’]).count()): Aquí se traza el gráfico de líneas. df.groupby([‘Hora’]).count() agrupa los datos por hora y cuenta el número de ocurrencias en cada hora. Luego, ‘plt.plot()‘ traza el número de ocurrencias para cada hora. - ‘horas‘ representa el eje x (las horas del día). - df.groupby([‘Hora’]).count() representa el eje y (el número de ocurrencias).

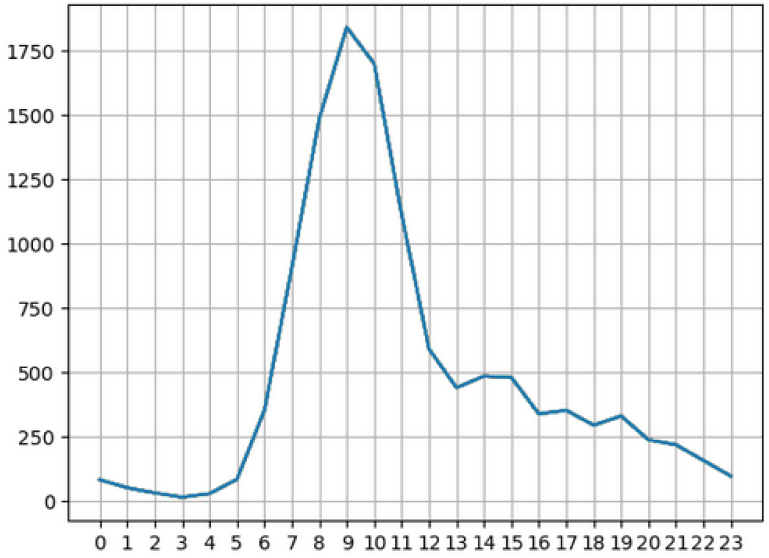
- En la línea 3: plt.grid(): Esta línea añade una cuadrícula al gráfico, lo que facilita la lectura

y la interpretación de los datos.

- En la línea 4: plt.xticks(horas): Esto establece las marcas en el eje x del gráfico, utilizando las horas extraídas anteriormente. Esto asegura que las horas estén correctamente etiquetadas en el eje x.

En síntesis, este código traza un gráfico de líneas que muestra cómo varía el número de ocurrencias de datos en el DataFrame 'df' según la hora del día. Esto es útil para visualizar patrones o tendencias en los datos a lo largo del tiempo. Sin embargo, es posible que desees asegurarte de que los datos estén correctamente ordenados antes de trazar el gráfico.

Out[9]:



In [10]:
```
lun = df[df['DiaSemana'] == 'Lunes'].copy()
mar = df[df['DiaSemana'] == 'Martes'].copy()
mie = df[df['DiaSemana'] == 'Miércoles'].copy()
jue = df[df['DiaSemana'] == 'Jueves'].copy()
vie = df[df['DiaSemana'] == 'Viernes'].copy()
sab = df[df['DiaSemana'] == 'Sábado'].copy()
dom = df[df['DiaSemana'] == 'Domingo'].copy()
```

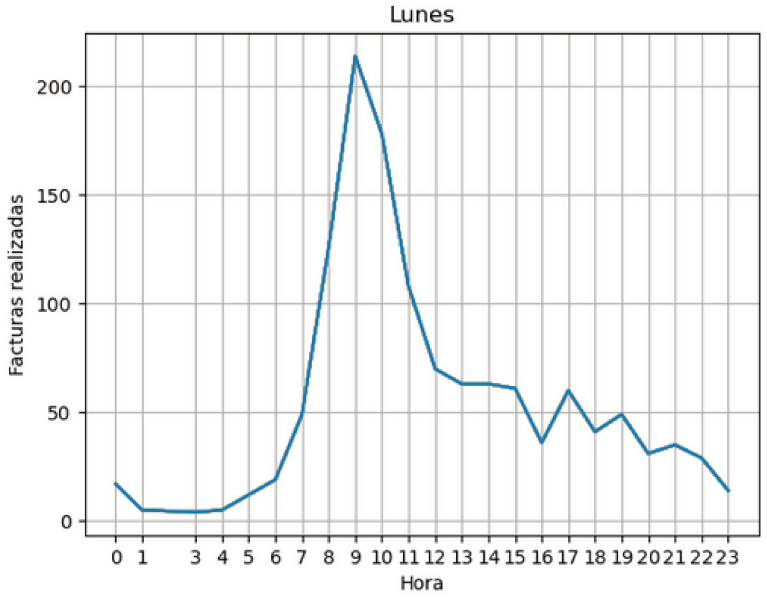
En el código antecedente, se logra filtrar tu DataFrame 'df' en diferentes DataFrames para cada día de la semana. Sin embargo, hay un error en la forma en que estás asignando los valores.

Con '.copy()' al final de cada filtro, se está creando una copia independiente de los datos filtrados en un nuevo DataFrame. Esto asegura que cada variable ('lun', 'mar', 'mie', 'jue', 'vie', 'sab' y 'dom') apunte a un DataFrame diferente, evitando así la modificación involuntaria de 'df'.

In [11]:
```
horas = [hora for hora, lun in lun.groupby('Hora')]
plt.plot (horas, lun.groupby(['Hora']).count())
plt.grid()
plt.xlabel ('Hora')
plt.title ('Lunes')
plt.ylabel ('Facturas realizadas')
plt.xticks(horas);
```

Similar a lo realizado, en la entrada N° 9, se logra un gráfico de líneas, habiendo filtrado sólo los registros pertencientes a los días lunes.

Out[11]:



In [12]:
```
fig = plt.figure(figsize = (22,6))
plt.title('Salidas La Posada 2023')
fig.tight_layout()

plt.subplot(3,3,1)
plt.title('Lunes')
fig.tight_layout()
horas = [hora for hora, lun in lun.groupby('Hora')]
plt.plot (horas, lun.groupby(['Hora']).count())
plt.grid()
plt.xlabel ('Hora')
plt.title ('Lunes')
plt.ylabel ('Facturas realizadas')
plt.xticks(horas)

plt.subplot(3,3,2)
plt.title('Martes')
```

```
fig.tight_layout()
horas = [hora for hora, mar in mar.groupby('Hora')]
plt.plot (horas, mar.groupby(['Hora']).count())
plt.grid()
plt.xlabel ('Hora')
plt.xticks(horas)

plt.subplot(3,3,3)
plt.title('Miercoles')
fig.tight_layout()
horas = [hora for hora, mie in mie.groupby('Hora')]
plt.plot (horas, mie.groupby(['Hora']).count())
plt.grid()
plt.xlabel ('Hora')
plt.xticks(horas)

plt.subplot(3,3,4)
plt.title('Jueves')
fig.tight_layout()
horas = [hora for hora, jue in jue.groupby('Hora')]
plt.plot (horas, jue.groupby(['Hora']).count())
plt.grid()
plt.xlabel ('Hora')
plt.ylabel ('Facturas realizadas')
plt.xticks(horas)

plt.subplot(3,3,5)
plt.title('Viernes')
fig.tight_layout()
horas = [hora for hora, vie in vie.groupby('Hora')]
plt.plot (horas, vie.groupby(['Hora']).count())
plt.grid()
plt.xlabel ('Hora')
plt.xticks(horas);

plt.subplot(3,3,6)
plt.title('Sabado')
fig.tight_layout()
horas = [hora for hora, sab in sab.groupby('Hora')]
plt.plot (horas, sab.groupby(['Hora']).count())
plt.grid()
plt.xlabel ('Hora')
plt.xticks(horas);

plt.subplot(3,3,7)
plt.title('Domingo')
fig.tight_layout()
horas = [hora for hora, dom in dom.groupby('Hora')]
plt.plot (horas, dom.groupby(['Hora']).count())
plt.grid()
plt.xlabel ('Hora')
```

```
69 plt.ylabel ('Facturas realizadas')
70 plt.xticks(horas)
71
72 plt.subplot(3,3,(8,9))
73 plt.title('Total Promedio')
74 fig.tight_layout()
75 horas = [hora for hora, df in df.groupby('Hora')]
76 plt.plot (horas, df.groupby(['Hora']).count())
77 plt.grid()
78 plt.xlabel ('Hora')
79 plt.xticks(horas)
80
81 plt.suptitle('La Posada salidas 2023',fontsize=20)
82 fig.tight_layout();
83
84 plt.savefig ('La Posada 2023 Salidas por hora.png', bbox_inches='tight')
```

Este código genera un gráfico de múltiples subgráficos utilizando Matplotlib para visualizar la cantidad de facturas realizadas por hora para cada día de la semana y el total promedio. Se detalla, paso a paso, la funcionalidad de cada línea de código para una mejor comprensión:

En la línea 1: fig = plt.figure(figsize=(22, 6)): Se crea una nuevo gráfico con un tamaño de 22x6 pulgadas.

En la línea 5: plt.subplot(3, 3, 1): Divide la figura en una cuadrícula de 3x3 y selecciona el primer subgráfico.

En la línea 9: plt.plot(horas, lun.groupby([‘Hora’]).count()): Traza un gráfico de líneas que muestra la cantidad de facturas realizadas por hora para el lunes. Se utiliza la función ‘groupby()‘ para agrupar los datos por hora y luego se cuenta el número de facturas realizadas en cada hora.

En la línea 12: plt.title(‘Lunes’): Establece el título del subgráfico.

Con las líneas 11 y 13: plt.xlabel(‘Hora’) y plt.ylabel(’Facturas realizadas’): Establecen etiquetas para los ejes x e y respectivamente.

En la línea 14: plt.xticks(horas): Establece las marcas del eje x en las horas correspondientes.

Desde la línea 25 hasta la línea 70: Se repiten estos pasos para los subgráficos restantes, cada uno mostrando la cantidad de facturas realizadas por hora para un día específico de la semana.

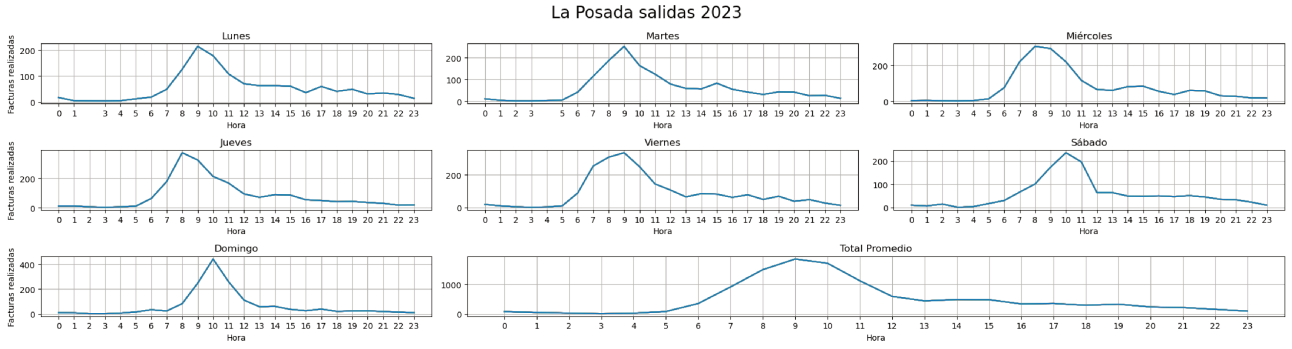
En la línea 72: plt.subplot(3, 3, (8, 9)): Este subgráfico está destinado a mostrar el total promedio de facturas realizadas por hora. El parámetro (8, 9) especifica que este subgráfico abarca las posiciones 8 y 9 de la cuadrícula 3x3.

En la línea 81: plt.suptitle('La Posada salidas 2023', fontsize=20): Establece un título general para toda la figura.

En la línea 84: plt.savefig('La Posada 2023 Salidas por hora.png', bbox_inches='tight'): Guarda la figura como un archivo PNG llamado 'La Posada 2023 Salidas por hora.png'. El parámetro 'bbox_inches='tight'' ayuda a recortar el espacio en blanco no deseado alrededor del gráfico.

Sintetizando, este código crea una visualización detallada de la cantidad de facturas realizadas por hora para cada día de la semana y el total promedio de facturas realizadas por hora para La Posada en el año 2023.

Out[12]:



In [13]:
```
hora_group = df.groupby('DiaSemana')
cantidad_facturas = hora_group.count()['Hora']
```

El código bajo expuesto realiza un conteo de la cantidad de facturas por día de la semana. Aquí está la explicación de cada parte del código:

En la línea 1: hora_group = df.groupby('DiaSemana'): Esta línea agrupa el DataFrame 'df' por la columna 'DiaSemana', lo que significa que los datos se dividirán en grupos según los diferentes días de la semana.

En la línea 2: cantidad_facturas = hora_group.count()['Hora']: Después de agrupar los datos, utilizas la función 'count()' para contar el número de filas en cada grupo. Como estás interesado en el número de facturas por día, seleccionas la columna 'Hora' (o cualquier otra columna que tenga valores no nulos para todas las filas) para realizar el conteo. Esto devuelve

una Serie que contiene el número de facturas por día de la semana.

El resultado 'cantidad_facturas' es una Serie que muestra la cantidad de facturas realizadas por día de la semana. Cada índice de la Serie representa un día de la semana y el valor asociado es la cantidad de facturas realizadas ese día.

Out[13]:

```
DiaSemana
Domingo      1604
Jueves       2003
Lunes        1289
Martes       1464
Miercoles    1842
Sabado       1373
Viernes      2156
Name: Hora, dtype: int64
```

In [14]:
```
clientes_agrupados = df.groupby('Cliente').count()['Nro_Comprobante'].
    reset_index()
clientes_agrupados_sorted = clientes_agrupados.sort_values(by='
    Nro_Comprobante', ascending=False)
clientes_agrupados_sorted.head(5)
```

El código previament expuesto realiza las siguientes operaciones:

En la línea 1:

clientes_agrupados = df.groupby('Cliente').count()['Nro_Comprobante'].reset_index(): Se agrupan los datos por el nombre del cliente ('Cliente'), cuenta el número de comprobantes para cada cliente y luego restablece el índice para que 'Cliente' sea una columna en lugar de un índice.

En la línea 2:

clientes_agrupados_sorted = clientes_agrupados.sort_values(by='Nro_Comprobante', ascending=False): Se ordenan los datos de los clientes agrupados por el número de comprobantes en orden descendente (de mayor a menor) según la columna 'Nro_Comprobante'.

En la línea 3: clientes_agrupados_sorted.head(5): Muestra las primeras 5 filas del DataFrame resultante, que contiene los clientes ordenados por el número de comprobantes en orden descendente.

Resumiendo, este código te proporciona una lista de los 5 clientes principales (los que tienen el mayor número de comprobantes) en tu conjunto de datos, junto con el recuento de comprobantes para cada uno de ellos. Esto es útil para identificar a los clientes más importantes

o frecuentes en tu negocio.

Out[14]:

	CODORI	NOMBRE	NROHAB	FORPAG	IMPNAC	FCHALT
0	A 0012-00004817	MENDELER, SILVIA	105.0	CDO	65416.03	01/01/2023 22:56
1	B 0012-00015869	SINNER, ORLANDO	108.0	CDO	24000.00	01/01/2023 05:56
2	NaN	LEAO CARVALHO, JUAN	214.0	CDO	0.00	01/01/2023 08:02
3	NaN	TINDOR, LAURA	207.0	CDO	0.00	01/01/2023 09:43
4	B 0012-00015870	SUSE DURO, MATIAS	110.0	CDO	17300.00	01/01/2023 09:44

In [15]:
```
clientes_agrupados2 = df.groupby('Cliente')['Importe'].sum().reset_index()
clientes_agrupados2 = clientes_agrupados2.sort_values(by='Importe',
    ascending=False)
clientes_agrupados2.head(5)
```

El código bajo expuesto realiza las siguientes operaciones:

En la línea 1: clientes_agrupados2 = df.groupby('Cliente')['Importe'].sum().reset_index(): Se agrupan los datos por el nombre del cliente ('Cliente') y calcula la suma total de los importes para cada cliente. Luego, restablece el índice para que 'Cliente' sea una columna en lugar de un índice.

En la línea 2: clientes_agrupados2 = clientes_agrupados2.sort_values(by='Importe', ascending=False): Se ordenan los datos de los clientes agrupados por la suma total de importes en orden descendente (de mayor a menor) según la columna 'Importe'.

En la línea 3: clientes_agrupados2.head(5): Muestra las primeras 5 filas del DataFrame resultante, que contiene los clientes ordenados por la suma total de importes en orden descendente.

Resumiendo, este código te proporciona una lista de los 5 clientes principales (los que tienen el mayor total de importes) en tu conjunto de datos, junto con la suma total de importes para cada uno de ellos. Esto es útil para identificar a los clientes más importantes en términos de ingresos generados.

Out[15]:

	Cliente	Importe
1221	CONSUMIDOR FINAL	71403637.03
66	ACADEMIA NACIONAL DE ESTUDIOS	15112800.00
1216	CONFEDERACION SUDAMERICANA PATIN (CONSUR)	11983000.00
1545	DIRECCION PROVINCIAL DE ASISTENCIA	10371284.74
4696	USINA CENTRAL SACIF	10101800.00

In [16]:
```
facturas_por_dia_y_hora = df.groupby(['DiaSemana', 'Hora']).size().
    reset_index(name='Totales')
facturas_por_dia_y_hora.head(11)
```

El código bajo expuesto realiza lo siguiente:

En la línea 1: df.groupby([‘DiaSemana’, ‘Hora’]).size(): Se agrupan los datos por la combinación de ‘DiaSemana’ y ‘Hora’ y cuenta el número de ocurrencias para cada combinación. Esto devuelve una Serie que contiene el recuento de filas para cada combinación de día de la semana y hora.

En la línea 2: .reset_index(name=’Totales’): Restablece el índice del DataFrame resultante para convertir los índices de los grupos (‘DiaSemana’ y ‘Hora’) en columnas y asigna un nombre a la columna que contiene los totales, que se llama ’Totales’.

El resultado es un DataFrame que muestra el número de facturas realizadas para cada día de la semana y hora específica, con una columna adicional llamada ’Totales’ que contiene el recuento correspondiente. Este tipo de análisis puede ser útil para entender las tendencias de facturación a lo largo de los días de la semana y durante diferentes horas del día.

Out[16]:

	DiaSemana	Hora	Totales
0	Domingo	0	11
1	Domingo	1	10
2	Domingo	2	4
3	Domingo	3	4
4	Domingo	4	8
5	Domingo	5	17
6	Domingo	6	35
7	Domingo	7	25
8	Domingo	8	84
9	Domingo	9	247
10	Domingo	10	441

In [17]:
```
plt.figure(figsize=(15,8))
x = facturas_por_dia_y_hora['DiaSemana']
y = facturas_por_dia_y_hora['Hora']
sizes = facturas_por_dia_y_hora['Totales']*5
plt.scatter (x,y, s=sizes, c='red', alpha=0.5, edgecolor='red', linewidth
    =0.5)
```

```
hora = range(0,24)
plt.yticks(hora)
plt.grid(True)
plt.title('Salidas en horas', fontsize=20)
plt.xlabel('Dia de la semana', fontsize=16)
plt.ylabel('Hora de Salida', fontsize=16);
```

El código descripto anteriormente utiliza ‘matplotlib‘ para crear un diagrama de dispersión que muestra las salidas en diferentes horas de cada día de la semana. Aquí está el desglose del código:

En la línea 1: plt.figure(figsize=(15,8)): Se crea una nueva figura con una ancho de 15 pulgadas de ancho por 8 pulgadas de alto.

En la línea 2:

x = facturas_por_dia_y_hora[‘DiaSemana’] e y = facturas_por_dia_y_hora[‘Hora’]: Se seleccionan los datos del eje x (día de la semana) y del eje y (hora de salida) del DataFrame “facturas_por_dia_y_hora”.

En la línea 4: sizes = facturas_por_dia_y_hora[’Totales’]*5: Define el tamaño de los puntos en el gráfico. Se multiplican los totales de facturas por 5 para hacer que los puntos sean más visibles.

En la línea 5: plt.scatter(x, y, s=sizes, c=’red’, alpha=0.5, edgecolor=’red’, linewidth=0.5): Crea un gráfico de dispersión donde los puntos representan las salidas. Los puntos son de color rojo con una transparencia de 0.5 y tienen un borde rojo. El tamaño de los puntos está determinado por la variable “sizes”.

En la línea 6: hora = range(0,24) y plt.yticks(hora): Establecen las marcas en el eje y para representar las 24 horas del día.

En la línea 7: plt.grid(True): Añade una cuadrícula al gráfico.

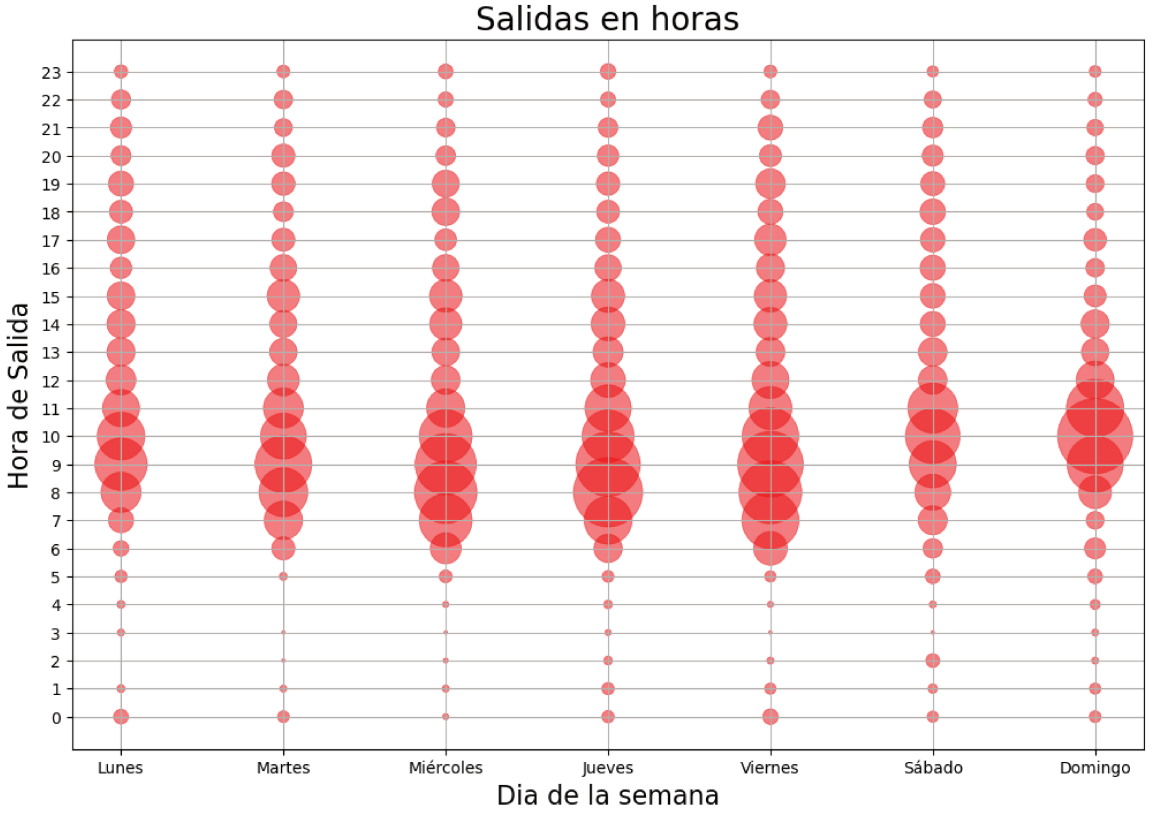
En la línea 9: plt.title(‘Salidas en horas’, fontsize=20): Establece el título del gráfico.

En las líneas 10 y 11: plt.xlabel(’Dia de la semana’, fontsize=16) y plt.ylabel(‘Hora de Salida’, fontsize=16): Establecen las etiquetas de los ejes x e y, y el tamaño de letras de dichas etiquetas.

Para resumir, el gráfico de dispersión proporciona una visualización de las salidas en diferentes horas para cada día de la semana, lo que puede ayudar a identificar patrones de

comportamiento a lo largo del tiempo.

Out[17]:



In [18]:
```
lista_ordenada = ['Lunes','Martes','Miercoles','Jueves','Viernes','Sabado','
    Domingo']
facturas_por_dia_y_hora1 = facturas_por_dia_y_hora
facturas_por_dia_y_hora1['DiaSemana'] = pd.Categorical(
    facturas_por_dia_y_hora['DiaSemana'], lista_ordenada, ordered=True)
facturas_por_dia_y_hora1.sort_values('DiaSemana')
```

El script anterior intenta ordenar el DataFrame “facturas_por_dia_y_hora” según el orden de los días de la semana especificado en la lista “lista_ordenada”. Aquí está el desglose del código:

En la línea 1: lista_ordenada = [’Lunes’, ’Martes’, ’Miércoles’, ’Jueves’, ’Viernes’, ’Sábado’, ’Domingo’]: Define una lista que contiene los días de la semana en el orden deseado.

En la línea 2: facturas_por_dia_y_hora1 = facturas_por_dia_y_hora: Crea una copia del DataFrame ‘facturas_por_dia_y_hora’ llamada ‘facturas_por_dia_y_hora1’.

En la línea 3: facturas_por_dia_y_hora1[‘DiaSemana’] =

pd.Categorical(facturas_por_dia_y_hora[‘DiaSemana’], lista_ordenada, ordered=True): Convierte la columna ‘DiaSemana’ del DataFrame ‘facturas_por_dia_y_hora1’ en una variable categórica de Pandas. Se especifica el orden de los días de la semana utilizando la lista ‘lista_ordenada’.

En la línea 4: facturas_por_dia_y_hora1.sort_values(‘DiaSemana’): Ordena el DataFrame ‘facturas_por_dia_y_hora1’ según la columna ‘DiaSemana’.

El código permite ordenar el DataFrame según el orden de los días de la semana. Sin embargo, para visualizar el DataFrame ordenado, necesitarías asignar el resultado de la función ‘sort_values()’ a una variable o imprimirlo. Por ejemplo:

facturas_por_dia_y_hora_ordenado = facturas_por_dia_y_hora1.sort_values(‘DiaSemana’) print(facturas_por_dia_y_hora_ordenado)

O bien, si se requiere que los cambios se reflejen directamente en ‘facturas_por_dia_y_hora1’, puedes usar ‘inplace=True’ en la función ‘sort_values()’:

facturas_por_dia_y_hora1.sort_values(‘DiaSemana’, inplace=True)

print(facturas_por_dia_y_hora1)

Esto imprimirá el DataFrame ‘facturas_por_dia_y_hora1’ ordenado según el orden de los días de la semana especificado en ‘lista_ordenada‘.

In [19]:
```
facturas_por_dia_y_mes = df.groupby(['DiaSemana', 'Mes']).size().reset_index
    (name='Totales')
facturas_por_dia_y_mes.head(15)
```

In [20]:
```
plt.figure(figsize=(12,8))

facturas_por_dia_y_mes.sort_values(by=['DiaSemana'], inplace=True)

x = facturas_por_dia_y_mes['DiaSemana']
y = facturas_por_dia_y_mes['Mes']
sizes = facturas_por_dia_y_mes['Totales']*5
plt.scatter (x,y, s=sizes, c='yellow', alpha=0.75, edgecolor='black',
    linewidth=0.5)
hora = range(0,13)
plt.yticks(hora)
plt.grid(True)
plt.title('Salidas en horas', fontsize=20)
plt.xlabel('Dia de la semana', fontsize=16)
plt.ylabel('Mes', fontsize=16);
```

```
15 plt.savefig ('La Posada 2023 Salidas por dia y mes.png', bbox_inches='tight'
    )
```

El código de python que antecede genera un gráfico de dispersión que muestra las salidas en diferentes meses de cada día de la semana. Aquí hay un desglose del código:

En la línea 1: plt.figure(figsize=(12,8)): Se crea un nuevo gráfico con una anchura de 12 pulgadas y una altura de 8 pulgadas.

En la línea 3: facturas_por_dia_y_mes.sort_values(by=[‘DiaSemana’], inplace=True): Se ordena el DataFrame ‘facturas_por_dia_y_mes‘ por la columna ‘DiaSemana’ de forma ascendente.

En las líneas 5 y 6:

x = facturas_por_dia_y_mes[‘DiaSemana’], y y = facturas_por_dia_y_mes[’Mes’]: Se seleccionan los datos del eje x (día de la semana) y del eje y (mes) del DataFrame ‘facturas_por_dia_y_mes’.

En la línea 7: sizes = facturas_por_dia_y_mes[‘Totales’]*5: Define el tamaño de los puntos en el gráfico. Se multiplican los totales de facturas por 5 para hacer que los puntos sean más visibles.

En la línea 8: plt.scatter(x, y, s=sizes, c=’yellow’, alpha=0.75, edgecolor=’black’, linewidth=0.5): Crea un gráfico de dispersión donde los puntos representan las salidas. Los puntos son de color amarillo con una transparencia de 0.75 y tienen un borde negro. El tamaño de los puntos está determinado por la variable ‘sizes‘.

En las líneas 9 y 10: hora = range(0, 13) y plt.yticks(hora): Establecen las marcas en el eje y para representar los 12 meses del año.

En la línea 11: plt.grid(True): Añade una cuadrícula al gráfico.

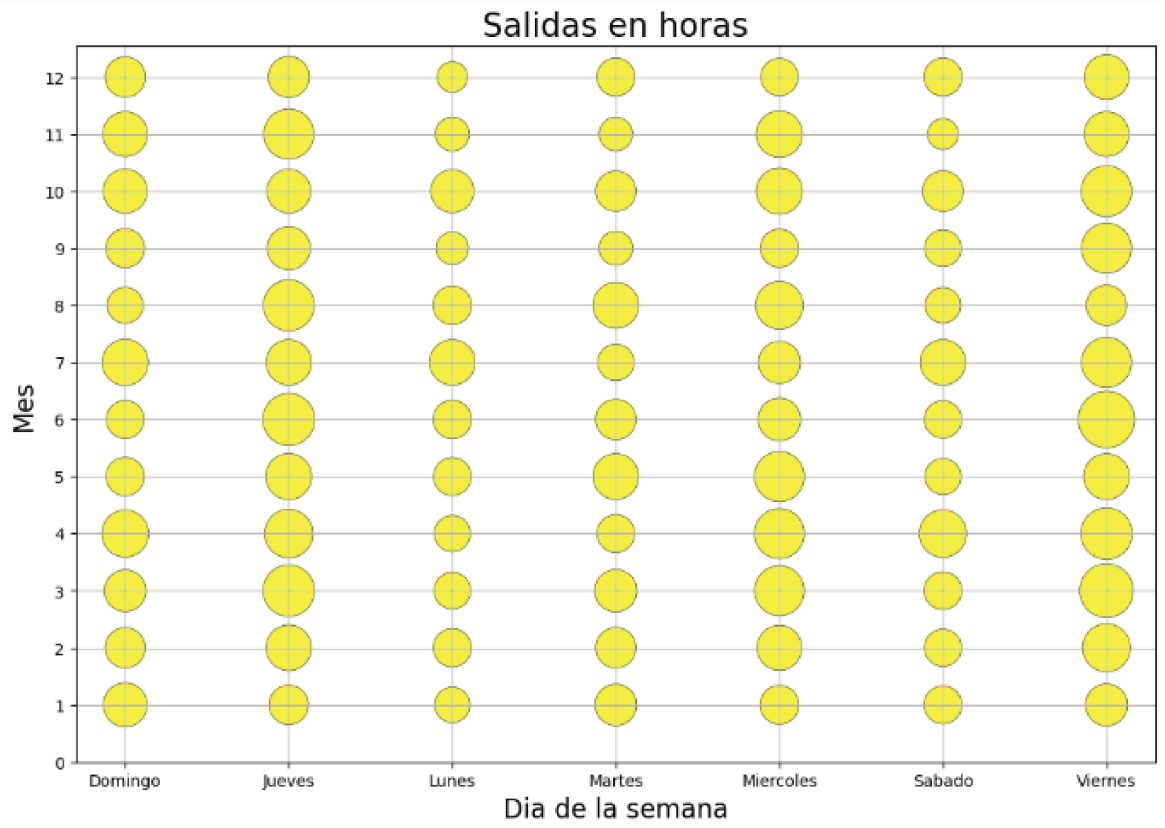
En la línea 12 : plt.title(‘Salidas en horas’, fontsize=20): Establece el título del gráfico.

En las líneas 13 y 14: plt.xlabel(‘Dia de la semana’, fontsize=16) y plt.ylabel(’Mes’, fontsize=16): Establecen las etiquetas de los ejes x e y.

En la línea 15: plt.savefig(’La Posada 2023 Salidas por dia y mes.png’, bbox_inches=’tight’): Guarda la figura como un archivo PNG llamado ‘La Posada 2023 Salidas por dia y mes.png’. El parámetro bbox_inches=‘tight’ ayuda a recortar el espacio en blanco no deseado alrededor del gráfico.

Para resumir, este gráfico de dispersión proporciona una visualización de las salidas en diferentes meses para cada día de la semana, lo que puede ayudar a identificar patrones de comportamiento a lo largo del tiempo.

Out[20]:



In [21]:
```
1 ventas_por_mes = df.groupby(df['Fecha'].dt.to_period('M'))['Importe'].sum()
2 plt.figure(figsize=(10, 6))
3 ventas_por_mes.plot(kind='bar', color='skyblue')
4 plt.title('Total de Ventas por Mes')
5 plt.xlabel('Mes')
6 plt.ylabel('Total de Ventas')
7 plt.xticks(rotation=45, ha='right')
8 plt.tight_layout()
```

Con estas las líneas de código se crea un gráfico de barras que muestra el total de ventas por mes. Aquí está el desglose del código:

En la línea 1:

ventas_por_mes = df.groupby(df['Fecha'].dt.to_period('M'))['Importe'].sum(): Se agrupan los datos por mes utilizando la función 'groupby' y calcula la suma total de importes para cada

mes. ‘df[’Fecha’].dt.to_period(’M’) convierte la columna de fechas en periodos mensuales.

En la línea 2: plt.figure(figsize=(10, 6)): Se crea una nueva figura de 10 pulgadas ancho y 6 pulgadas de alto.

En la línea 3: ventas_por_mes.plot(kind=’bar’, color=’skyblue’): Crea un gráfico de barras donde el eje x representa los meses y el eje y representa el total de ventas para cada mes. El parámetro ‘color=’skyblue’‘ establece el color de las barras.

En la línea 4: plt.title(‘Total de Ventas por Mes’): Establece el título del gráfico.

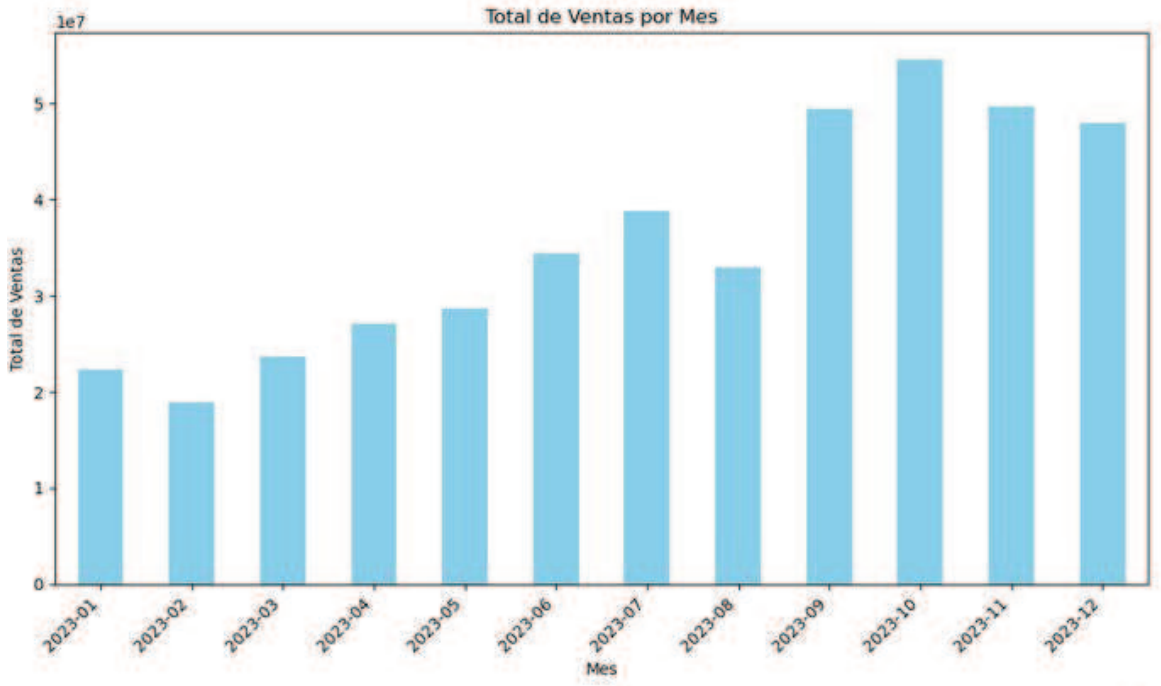
En las líneas 5 y 6: plt.xlabel(‘Mes’) y plt.ylabel(’Total de Ventas’): Establecen las etiquetas de los ejes x e y, respectivamente.

En la línea 6: plt.xticks(rotation=45, ha=‘right’): Rota las etiquetas del eje x 45 grados y alinea las etiquetas a la derecha para una mejor legibilidad.

En la línea 8: plt.tight_layout(): Ajusta automáticamente el diseño del gráfico para que quepa correctamente en la figura.

En síntesis, este gráfico de barras proporciona una visualización clara del total de ventas por mes, lo que facilita la comparación de las ventas a lo largo del tiempo. El color skyblue es utilizado para mejorar la estética del gráfico.

Out[21]:



In [22]:
```
1 from matplotlib.ticker import FuncFormatter
2
3 ventas_por_dia = df.groupby(df['Fecha'].dt.date)['Importe'].sum()
4
5 plt.figure(figsize=(10, 6))
6 plt.plot(ventas_por_dia.index, ventas_por_dia.values, marker='o', color='
    skyblue', linestyle='-')
7
8 formatter = FuncFormatter(lambda x, _: '{:.0f}M'.format(x / 1e6))
9 plt.gca().yaxis.set_major_formatter(formatter)
10
11 plt.title('Ventas Totales por Dia')
12 plt.xlabel('Dia')
13 plt.ylabel('Total de Ventas')
14 plt.xticks(rotation=45, ha='right')
15 plt.tight_layout()
16 plt.show()
```

Este código genera un gráfico de líneas que muestra las ventas totales por día. Aquí está el desglose del código:

En la línea 3: ventas_por_dia = df.groupby(df['Fecha'].dt.date)['Importe'].sum(): Se agrupan los datos por fecha, calcula la suma total de importes para cada día y crea una Serie llamada 'ventas_por_dia' que contiene las fechas en el índice y el total de ventas en los valores.

En la línea 4: plt.figure(figsize=(10, 6)): Se crea un nuevo gráfico con una anchura de

10 pulgadas y una altura de 6 pulgadas.

En la línea 5: plt.plot(ventas_por_dia.index, ventas_por_dia.values, marker='o', color='skyblue', linestyle='-'): Traza el gráfico de líneas utilizando las fechas en el eje x y los totales de ventas en el eje y. Se utilizan círculos como marcadores, el color skyblue y un estilo de línea sólida.

En la línea 8: formatter = FuncFormatter(lambda x, _: ':.0fM'.format(x / 1e6)): Crea un formateador personalizado para el eje y que muestra los valores en millones con el sufijo 'M'.

En la línea 9: plt.gca().yaxis.set_major_formatter(formatter): Establece el formateador personalizado en el eje y.

En la línea 11: plt.title('Ventas Totales por Dia'): Establece el título del gráfico.

En las líneas 12 y 13: plt.xlabel('Dia') y plt.ylabel('Total de Ventas'): Establecen las etiquetas de los ejes x e y, respectivamente.

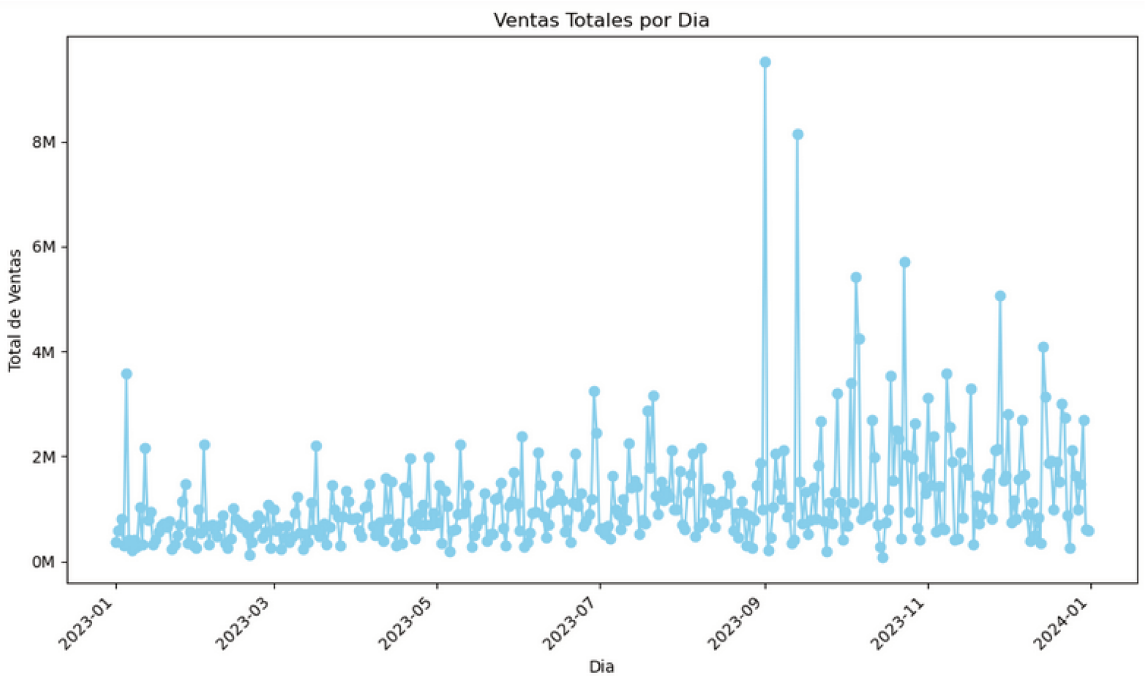
En la línea 14: plt.xticks(rotation=45, ha='right'): Rota las etiquetas del eje x 45 grados y alinea las etiquetas a la derecha para una mejor legibilidad.

En la línea 15: plt.tight_layout(): Ajusta automáticamente el diseño del gráfico para que quepa correctamente en la figura.

En la línea 16: plt.show(): Muestra el gráfico.

Resumiendo, este gráfico de líneas proporciona una visualización clara de las ventas totales por día, lo que permite identificar tendencias y patrones en las ventas a lo largo del tiempo. Además, el formateador personalizado en el eje y mejora la legibilidad al mostrar los valores en millones.

Out[22]:



In [23]:
```
nuevo_df = df[df['Habitacion'] != 0]
nuevo_df = nuevo_df[(nuevo_df['Habitacion'] > 100) & (nuevo_df['Habitacion']
    < 199)]
facturas_por_habitacion = nuevo_df.groupby('Habitacion')['Importe'].count()
facturas_por_habitacion.info

plt.figure(figsize=(9, 9))
plt.pie(facturas_por_habitacion, labels=facturas_por_habitacion.index,
   autopct='%1.1f%%', startangle=140)
plt.title('Distribucion de Facturas Emitidas por Habitacion (Primer Piso)')
plt.axis('equal')
plt.tight_layout()
plt.show()
```

El código bajo expuesto realiza lo siguiente:

En la línea 1: nuevo_df = df[df[‘Habitacion’] != 0]: Filtra el DataFrame original (‘df’) para excluir las filas donde el número de habitación es igual a 0.

En la línea 2:

nuevo_df = nuevo_df[(nuevo_df[‘Habitacion’] 100) & (nuevo_df[‘Habitacion’] 199)]: Filtra el DataFrame ‘nuevo_df’ para incluir solo las filas donde el número de habitación está en el rango de 101 a 198.

En la línea 3:

facturas_por_habitacion = nuevo_df.groupby('Habitacion')['Importe'].count(): Se agrupan los datos del DataFrame 'nuevo_df' por número de habitación y cuenta el número de facturas emitidas para cada habitación.

En la línea 6: plt.figure(figsize=(9, 9)): Se crea una nueva figura para el gráfico de torta de nueve por nueve pulgadas.

En la línea 7: plt.pie(facturas_por_habitacion, labels=facturas_por_habitacion.index, autopct='%1.1f%%', startangle=140): Crea el gráfico de pastel con las frecuencias de facturas por habitación. Las etiquetas del gráfico son los números de habitación y el porcentaje de cada porción se muestra con una precisión de un decimal.

En la línea 8: plt.title('Distribucion de Facturas Emitidas por Habitacion (Primer Piso)'): Establece el título del gráfico.

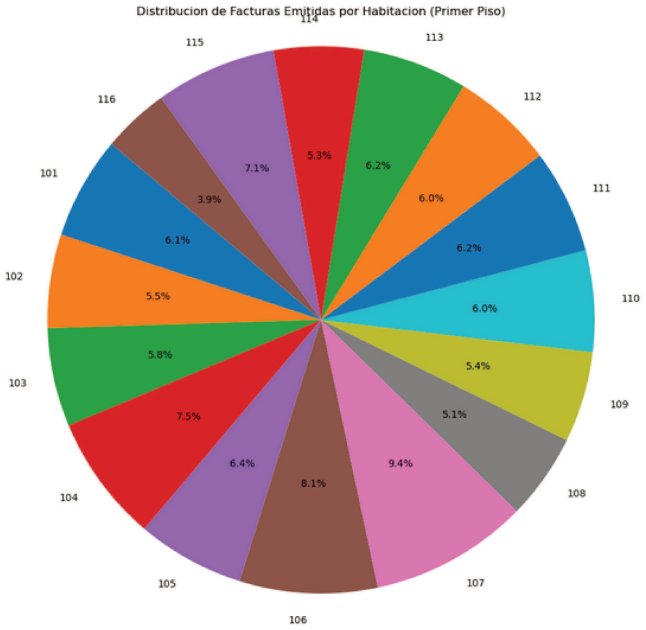
En la línea 9: plt.axis('equal'): Se ajusta el aspecto del gráfico para que el pastel se dibuje como un círculo perfecto.

En la línea 10: plt.tight_layout(): Ajusta automáticamente el diseño del gráfico para que quepa correctamente en la figura.

En la línea 11: plt.show(): Muestra el gráfico de torta.

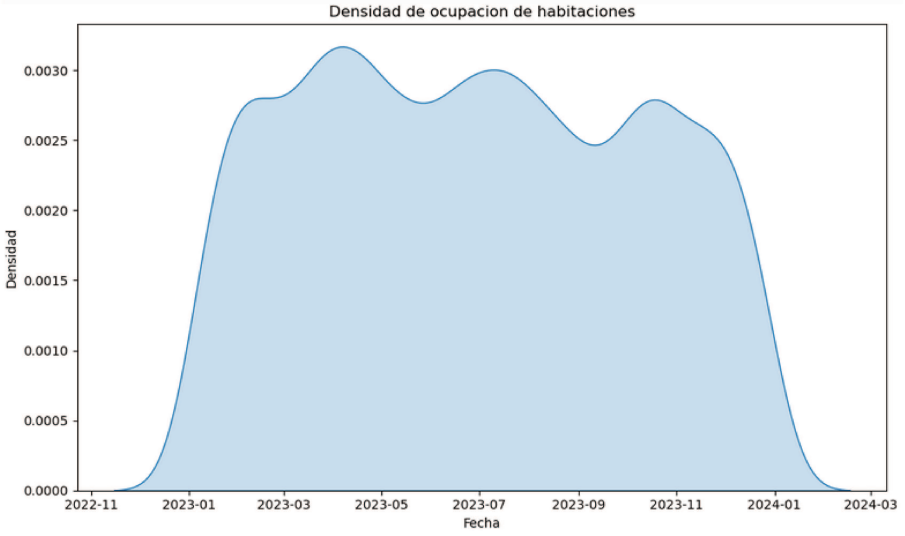
Este código generará un gráfico de pastel que muestra la distribución del número de facturas emitidas por habitación en el primer piso. Cada porción del pastel representará el porcentaje de facturas emitidas para una habitación específica en relación con el total.

Out[23]:



In [24]:
```
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.kdeplot(data=df, x='Fecha', fill=True)
plt.title('Densidad de ocupacion de habitaciones')
plt.xlabel('Fecha')
plt.ylabel('Densidad')
plt.tight_layout()
plt.show()
```

Out[24]:



El código precedente utiliza la biblioteca Seaborn para crear un gráfico de densidad de kernel (KDE) que muestra la densidad de ocupación de habitaciones a lo largo del tiempo. Aquí está el desglose del código:

En la línea 1: import seaborn as sns: Importa la biblioteca Seaborn para trazar gráficos estadísticos más avanzados.

En la línea 2: plt.figure(figsize=(10, 6)): Se crea una nueva figura para el gráfico con una anchura de 10 pulgadas y una altura de 6 pulgadas.

En la línea 3: sns.kdeplot(data=df, x='Fecha', fill=True): Crea un gráfico de densidad de kernel (KDE) utilizando los datos del DataFrame 'df'. La variable en el eje x es 'Fecha', que representa el tiempo. El parámetro 'fill=True' rellena el área bajo la curva de densidad.

En la línea 4: plt.title('Densidad de ocupacion de habitaciones'): Establece el título del gráfico.

En la líneas 5 y 6: plt.xlabel('Fecha') y plt.ylabel('Densidad'): Establecen las etiquetas de los ejes x e y, respectivamente.

En la línea 7: plt.tight_layout(): Ajusta automáticamente el diseño del gráfico para que quepa correctamente en la figura.

En la línea 8: plt.show(): Muestra el gráfico de densidad de kernel.

Sintetizando, este gráfico proporciona una representación visual de la densidad de ocupación de habitaciones a lo largo del tiempo, lo que puede ayudar a identificar patrones y fluctuaciones en la ocupación a lo largo de diferentes fechas. La curva de densidad muestra la distribución de la ocupación de habitaciones en función de la fecha.

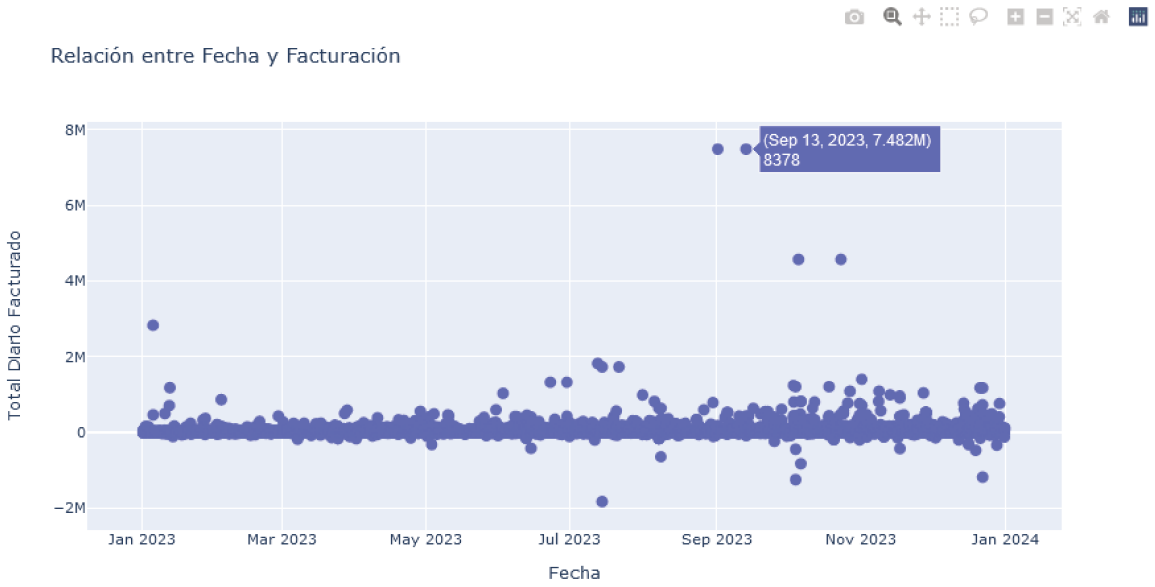
In [25]:
```
import plotly.graph_objects as go
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=df['Fecha'],
    y=df['Importe'],
    mode='markers',
    marker=dict(size=10),
    text=df.index
))

fig.update_layout(
    title='Relación entre Fecha y Facturación',
    xaxis_title='Fecha',
    yaxis_title= 'Total Diario Facturado'
```

```
16 )
17
18 fig.show()
```

Out[25]:



Estas líneas de código se sirven de la librería Plotly para crear un gráfico de dispersión que muestra la relación entre la fecha y el total diario facturado. Aquí está el desglose del código:

En la línea 1: import plotly.graph_objects as go: Importa la clase ‘Figure‘ de Plotly para crear visualizaciones 1interactivas.

En la línea 2: fig = go.Figure(): Se crea una nueva figura en la variable ’fig’ donde se representará el gráfico.

Desde la línea 4 hasta la línea 10: Agrega un trazado de dispersión al gráfico.

fig.add_trace(go.Scatter(

x=df[’Fecha’], y=df[’Importe’]: Define los valores de las coordenadas x e y para el gráfico. En este caso, ’Fecha’ se usa para el eje x, e ’Importe’ para el eje y.

mode=’markers’: Establece el modo del trazado como ’markers’, lo que indica que se usarán puntos para representar los datos.

marker=dict(size=10): Define el tamaño de los marcadores en el gráfico.

text=df.index: Agrega texto que aparecerá al pasar el mouse sobre los puntos del gráfico.

Desde la línea 12 hasta la línea 16: Actualiza el diseño del gráfico.

fig.update_layout(‘

title=‘Relación entre Fecha y Facturación’: Establece el título del gráfico.

xaxis_title=‘Fecha’: Etiqueta el eje x como ‘Fecha’.

yaxis_title=‘Total Diario Facturado’: Etiqueta el eje y como ‘Total Diario Facturado’.

fig.show(): Muestra el gráfico interactivo.

Este gráfico de dispersión interactivo permite explorar la relación entre la fecha y el total diario facturado al pasar el mouse sobre los puntos. Es una herramienta útil para identificar tendencias y patrones en los datos de facturación a lo largo del tiempo.

23.5. Ejercicios Práctico N° 5: Automatizar el envío de correos electrónicos a clientes de un estudio contable

Presentación del problema: Ud. tiene asignadas entre sus tareas dentro de un estudio contable realizar la facturación de los servicios profesionales desde la página de AFIP y enviarles el comprobante emitido a cada cliente vía correo electrónico con un texto de estilo dentro de los primeros cinco días del mes siguiente al de la facturación.

Analice una forma en que este proceso, que se realiza todos los meses, pueda ser automatizado para disminuir las horas de recursos humanos asignada y disminuyendo errores de carga manual. Los datos correspondientes a cada cliente se encuentran registrados en un archivo de Excel que acumula mensualmente en diferentes hojas la facturación del mes correspondiente, según se observa en la siguiente imagen.

	A	B	C	D	E	F	G
1	Cliente	CUIT	Condición fiscal	Email	Importe	Aclaración en el email	Adjuntar
2	Cuevas Angelina	27-11111111-1	SUJETO EXENTO	cuevas@gmail.com	$ 50.000,00	Aclaración: Debe honorarios febrero 2024	801
3	Espino Alejandro	20-11111111-2	IVA INSCRIPTO	espino@live.com	$ 60.500,00	Aclaración: Su cuenta corriente no presenta facturas adeudadas.	802
4	Estrada Isabel	27-11111111-3	MONOTRIBUTO	estrada@gmail.com	$ 40.000,00	Aclaración: Debe honorarios enero y febrero de 2024	803
5	Fernandez Ana	27-11111111-4	MONOTRIBUTO	anafer@gmail.com	$ 30.000,00	Aclaración: Su cuenta corriente no presenta facturas adeudadas.	804
6	Heredia Agustin	20-11111111-5	MONOTRIBUTO	agus.h@hotmail.com	$ 30.000,00	Aclaración: Su cuenta corriente no presenta facturas adeudadas.	805
7	Marulo Manuel	27-11111111-5	IVA INSCRPTO	mm2020@gmail.com	$ 30.000,00	Aclaración: Su cuenta corriente no presenta facturas adeudadas.	806
8							
9							
10							

11-2023 | 12-2023 | 1-2024 | 2-2024 | 3-2024

Figura 30: Imagen del archivo origen de los datos en Excel.

In [1]:
```
archivo = "Detalle honorarios clientes.xlsx"
```

In [2]:
```
from datetime import date
hoy = date.today()
mes = int(today.strftime("%m"))
mes_anterior = mes-1
if mes_anterior==0:
    mes_anterior=12
año = int(today.strftime("%Y"))
if mes_anterior==12:
    año=año-1
hoja = (f"0{mes_anterior}-{año}")
if len(hoja)>6:
    hoja = hoja[1:]
print(f"El periodo facturado es {hoja}")
```

Este código en Python se utiliza para determinar el período de facturación anterior en función de la fecha actual. Aquí está el significado línea por línea:

En la línea 1: from datetime import date se utiliza para importar la clase 'date' del módulo 'datetime' en Python. Esto permite trabajar con fechas que se utilizarán para obtener la hoja de cálculo a utilizar en el mes en curso.

En la línea 2: hoy = date.today(): Crea un objeto de fecha llamado 'hoy' con la fecha actual del sistema.

En la línea 3: mes = int(today.strftime(" %m")): Obtiene el mes actual de la fecha actual y lo convierte en un número entero de uno a doce.

En la línea 4: mes_anterior = mes-1: Calcula el mes anterior al que corresponden los honorarios vencidos restando 1 al mes actual, para proceder con cálculos mediante un ciclo IF, debido a que puede surgir un inconveniente en el mes de enero ya que este valor mes_anterior será igual a cero.

En la línea 5: if mes_anterior==0:: Si el mes anterior resulta ser 0 (enero), se ajusta a diciembre (mes 12).

En la línea 7: año = int(today.strftime(" %Y")): Obtiene el año actual de la fecha actual y lo convierte en un número entero.

En la línea 8: if mes_anterior==12:: Si el mes anterior es diciembre (mes 12), se ajusta el año al año anterior.

En la línea 10: hoja = (f'0mes_anterior-año"): Crea una cadena de texto que representa el período de facturación en el formato "mm-yyyy". El objetivo es formatear el mes con un 0 delante si su longitud es menor a dos dígitos.

En la línea 11: if len(hoja) 6:: Verifica si la longitud de la cadena de texto almacenada en 'hoja' es mayor que 6. por ejemplo, si el mes anterior es octubre, noviembre o diciembre).

En la línea 12: hoja = hoja[1:]: Si la longitud de la cadena hoja es mayor que 6, se elimina el primer caracter (que sería el 0 adicional agregado anteriormente).

En la línea 13: print(f."El periodo facturado es hoja"): Imprime el período de facturación determinado, como verificación del período con el que se va a conntinuar trabajando..

```
In [3]: import pandas as  pd
        diccionario_honorarios=pd.read_excel(archivo,
                                             usecols=[0,3,4,5,6],
```

```
4                                          sheet_name=[hoja])
5 dataframe_honorarios = diccionario_honorarios.get(hoja).fillna("")
6 dataframe_honorarios.head()
```

En la línea 1: import pandas as pd: Importa la biblioteca Pandas y la nombra como pd para facilitar su uso en el código.

Desde la línea 2 hasta la línea 4: diccionario_honorarios=pd.read_excel(archivo, usecols=[0,3,4,5,6], sheet_name=[hoja]): Lee el archivo Excel especificado por archivo. Solo lee las columnas 0, 3, 4, 5 y 6 (tener presente que las columnas están indexadas desde cero). Lee solo la hoja especificada por hoja y carga los datos en un diccionario de DataFrames. Cada clave del diccionario corresponde al nombre de la hoja y el valor es un DataFrame que contiene los datos de esa hoja.

En la línea 5: dataframe_honorarios = diccionario_honorarios.get(hoja).fillna(): Obtiene el DataFrame correspondiente a la hoja especificada por hoja del diccionario_honorarios y lo guarda en la variable 'dataframe_honorarios'. Luego, se rellenan los valores faltantes en este DataFrame con una cadena vacía.

En la línea 6: dataframe_honorarios.head(): Muestra las primeras filas del DataFrame dataframe_honorarios. Este método muestra por defecto las primeras cinco filas del DataFrame, proporcionando una vista previa de los datos cargados desde el archivo Excel.

	Cliente	Email	Importe	Aclaración en el email	Adjuntar
0	Cuevas Angelina	cuevas@gmail.com	50000	Aclaración: Debe honorarios febrero 2024	801
1	Espino Alejandro	espino@live.com	60500	Aclaración: Su cuenta corriente no presenta fa...	802
2	Estrada Isabel	estrada@gmail.com	40000	Aclaración: Debe honorarios enero y febrero de...	803
3	Fernandez Ana	anafer@gmail.com	30000	Aclaración: Su cuenta corriente no presenta fa...	804
4	Heredia Agustin	agus.h@hotmail.com	30000	Aclaración: Su cuenta corriente no presenta fa...	805
5	Marulo Manuel	mm2020@gmail.com	30000	Aclaración: Su cuenta corriente no presenta fa...	806

```
In [4]: 1 import smtplib, ssl, email, time
        2 from email import encoders
        3 from email.mime.base import MIMEBase
        4 from email.mime.text import MIMEText
        5 from email.mime.multipart import MIMEMultipart
```

Con estas instrucciones se procede a la importación de la biblioteca smtplib, ssl, y el módulo email para enviar correos electrónicos desde una cuenta de correo electrónico. Este conjunto de líneas es lo básico para poder generar un script para el envío de correos e:

En la línea 1: import smtplib: Este módulo proporciona una interfaz para enviar mensajes de correo electrónico. Se utiliza para establecer una conexión SMTP con el servidor de correo

saliente.

En la línea 1: import ssl: Este módulo proporciona soporte para SSL (Secure Sockets Layer), que se utiliza para encriptar la conexión con el servidor SMTP, lo que garantiza la seguridad de la comunicación entre el cliente y el servidor.

En la línea 1: import email: Este módulo proporciona las clases y métodos necesarios para la creación y manipulación de mensajes de correo electrónico.

En la línea 1: import time: Este módulo proporciona funciones relacionadas con el tiempo, aunque no se utiliza explícitamente en este fragmento de código, podría ser útil para realizar pausas o gestión del tiempo en aplicaciones más complejas.

En la línea 2: from email import encoders: Importa la función encoders desde el módulo email. Esta función se utiliza para codificar adjuntos de correo electrónico antes de enviarlos.

En la línea 3: from email.mime.base import MIMEBase: Importa la clase MIMEBase desde el módulo email.mime.base. MIMEBase es una clase base para todos los tipos de partes MIME.

En la línea 4: from email.mime.text import MIMEText: Importa la clase MIMEText desde el módulo email.mime.text. Esta clase se utiliza para representar partes de texto de un mensaje de correo electrónico.

En la línea 5: from email.mime.multipart import MIMEMultipart: Importa la clase MIMEMultipart desde el módulo email.mime.multipart. MIMEMultipart se utiliza para representar un mensaje de correo electrónico multipart, que puede contener tanto texto como archivos adjuntos.

En resumen, este código importa las bibliotecas y módulos necesarios para enviar correos electrónicos con Python, establecer una conexión segura con el servidor SMTP utilizando SSL, y construir mensajes de correo electrónico con texto y archivos adjuntos utilizando la biblioteca email.

In [5]:
```
asunto = "Honorarios profesionales Estudio Contable"
mail_cco = ["sfumis@gmail.com"]
puerto = 465
smtp = 'smtp.gmail.com'
remitente = 'EstudioContable@gmail.com'
password = 'xcw34gghdfrhss'
```

El código de la entrada N° 5 está relacionado con el envío de correos electrónicos a través de un servidor SMTP (Protocolo simple de transferencia de correo) utilizando la biblioteca

'smtplib' de Python. En síntesis, lo que se está definiendo es:

En la línea 1: asunto: Es el asunto del correo electrónico que deseas enviar, en este caso, "Honorarios profesionales Estudio Contable".

En la línea 2: mail_cco: Es una lista de direcciones de correo electrónico a las que se enviará una copia oculta (CCO) del correo electrónico.

En la línea 3: puerto: Es el número de puerto del servidor SMTP que se utilizará para el envío del correo electrónico. En este caso es el puerto 463.

En la línea 4: smtp: Es la dirección del servidor SMTP que se utilizará para el envío del correo electrónico. En este caso, es 'smtp.gmail.com', que es el servidor SMTP de Gmail.

En la línea 5: remitente: Es la dirección de correo electrónico del remitente del correo electrónico. En este caso, es 'EstudioContablegmail.com'.

En la línea 6: password: Es la contraseña de la cuenta de correo electrónico del remitente. Ten en cuenta que este código contiene la contraseña en texto plano, lo cual puede ser un riesgo de seguridad. Es recomendable utilizar métodos más seguros para manejar contraseñas, como variables de entorno o sistemas de gestión de secretos.

Con este código se inicia un script que enviará correos electrónicos automatizados utilizando la configuración proporcionada. Es importante tener en cuenta que necesitarás importar la biblioteca 'smtplib' y posiblemente otras bibliotecas necesarias para el envío de correos electrónicos y trabajar con el protocolo SMTP. Además, se tendrá que configurar adecuadamente la autenticación SMTP utilizando el remitente y la contraseña definidos previamente.

In [6]:
```
for index, row in dataframe_honorarios.iterrows():
    nombre_cliente = row["Cliente"]
    destinatario = row["Email"]
    filename = row["Adjuntar"]
    filename1 = f'2022961217_011_0002_0000000{filename}.pdf'

    importe_sin_formato = row["Importe"]
    import locale
    locale.setlocale(locale.LC_ALL , 'es_AR')
    honorario = locale.currency(importe_sin_formato, grouping=True)

    aclaraciones = row["Aclaración en el email"]

    msg = MIMEMultipart("mixed")
    msg["Subject"] = asunto
    msg["From"] = remitente
    msg["To"] = destinatario

```

```
    cuerpo_mail_html = f"""\
                    <html><body>
                    Estimado/a {nombre_cliente}
                    <br>
                    Me dirijo a Ud. a fin de enviarle la facturación correspondiente a los honorarios profesionales brindados en el mes de {hoja} por un total de {honorario}.
                    <br><br>
                    Los datos para el abono de dicha factura son los siguientes:
                        <b>Razón Social:</b> SEBASTIAN FUMIS<br>
                        <b>CUIT Nro.:</b> 20 229612175 5<br>
                        <b>CBU:</b> 3658269014682682827 4<br>
                        <b>Alias: </b> estudio.contable.honorarios<br>
                    <br>
                    {aclaraciones}
                    <br><br>
                    Por favor, enviar el comprobante de pago una vez realizado el mismo al siguiente email: avisospagos@gmail.com.
                    <br><br>
                    Se adjunta factura con los honorarios correspondientes.
                    Quedo a disposición, por cualquier duda o consulta.
                    Saludos
                    <br><br>
                    CPN Sebastián Fumis
                    </body></html>
                    """
    part = MIMEText(cuerpo_mail_html, "html")
    msg.attach(part)

    if filename:
        with open(filename1, "rb") as attachment:
            part = MIMEBase("application", "octet-stream")
            part.set_payload(attachment.read())

    encoders.encode_base64(part)

    if filename1:
        part.add_header(
            "Content-Disposition",
            "attachment", filename= filename1
        )
        msg.attach(part)

    context = ssl.create_default_context()

    with smtplib.SMTP_SSL(smtp, puerto, context=context) as server:
        server.login(remitente, password)
        grupo_destinatarios = [destinatario] + mail_cco
        server.sendmail(remitente, grupo_destinatarios, msg.as_string())

```

```
66      import time
67      time.sleep(3)
68
69      print(f"Se envió facturación a: {nombre_cliente}")
```

Este es un típico caso de código tipo bucle, que itera sobre el DataFrame llamado 'dataframe_honorarios' y realiza las siguientes acciones en cada fila:

En la línea 2: Obtiene de cada uno de los registros –filas– del archivo base en formato Excel el nombre del cliente ubicado en la columna 'Cliente'.

En la línea 3: A continuación extrae del mismo registro la dirección de correo electrónico del destinatario ubicado cada en la columna 'Email'.

En las líneas 4 y 5 se configura el nombre del archivo a adjuntar teniendo en cuenta que en el archivo Excel por una cuestión de comodidad sólo se indican los últimos tres dígitos de la factura correspondiente. Mediante la quinta línea se logra configurar que el nombre del archivo a adjuntar sea coincidente con el formato predeterminado con el que se descarga originalmente del servidor de la AFIP.

Desde la línea 7 a 10: Se formatea el importe sin formato del honorario a cobrar utilizando el módulo 'locale' para representarlo en formato monetario.

En la línea 12: Se toma de la columna 'Aclaración' el texto que pueda tener la base de datos en la hoja del archivo Excel utilizado.

Desde la línea 19 hasta la 41: Se construye el cuerpo del correo electrónico en formato HTML, que incluye un saludo personalizado al cliente, con campos específicos obtenidos del archivo de origen y culmina con la firma del remitente.

Desde la línea 42 a 57 se relaciona el nombre del archivo a adjuntar obtenido del archivo base con los respectivos archivos descargados de la página de la AFIP guardados en la misma carpeta donde se encuentra el archivo con código Python.

Entre la línea 61 y 64 se desarrolla el proceso de logueo al servidor saliente del correo relacionando todas la variables determinadas con el código preexistente.

En la línea 66 se importa la librería 'time' que habilita el código 'time.sleep(3)' que genera una pausa de tres segundos para que el bucle FOR continúe con el siguiente registros a efectos de no sobresaturar el proceso de envío de correo electrónico.

En la línea 69 se plantea dentro del ciclo FOR que se imprima la leyenda de cada correo

electrónico enviado para control de funcionamiento.

Referencias

Baraldi, V. (2016). Enseñanza, investigación y extensión: tres prácticas articuladas en la cátedra de Didáctica General de la Universidad Nacional del Litoral. *+ E: Revista de Extensión Universitaria*, (6), 306-313.

Blanco, M. E., D. y Cazeneuve. (2023). La matemática argentina que enseña al mundo cómo los cálculos ayudan a entender la realidad [Último acceso 2 de abril de 2024]. https://www.infobae.com/america/ciencia-america/2023/11/15/la-matematica-argentina-que-ensena-al-mundo-como-los-calculos-ayudan-a-entender-la-realidad/

Bota, H., & Gosa, A. (2021). *Python for accounting*. Venta Online.

Bouhuijs, P. A. (2011). Implementing problem based learning: Why is it so hard? *REDU. Revista de Docencia Universitaria*, *9*(1).

Carrera, L. (2013). El aprendizaje basado en problemas como fundamento de una propuesta curricular innovadora. El caso de la carrera de medicina de la UNL. En E. UNL. (Ed.), *Tensiones entre disciplinas y competencias en el currículum universitario.* (pp. 159-166). Stubrin Adolfo; Diaz Natalia.

Clarín, D. (2022). Los lenguajes de programación más buscados por los unicornios argentinos [Último acceso 2 de abril de 2024]. https://www.clarin.com/tecnologia/lenguajes-programacion-buscados-unicornios-argentinos_0_IA9AVggs0G.html

Dewey, J. (1897). *My Pedagogic Creed, by Prof. John Dewey; Also, The Demands of Sociology Upon Pedagogy, by Prof. Albion W. Small*. New York,, Chicago,: EL Kellogg & Company.

Downey, A., Elkner, J., & Meyers, C. (2002). Aprenda a Pensar como un Programador con Python. *Aprenda a pensar como programador con python*.

Harari, Y. N. (2014). *Sapiens. De animales a dioses: Una breve historia de la humanidad*. Debate.

Harari, Y. N. (2018). *21 lecciones para el siglo XXI*. Debate.

Iglesias, J. (2002). El aprendizaje basado en problemas en la formación inicial de docentes. *Revista Perspectivas*, *32*(3), 79-95.

Lion, C. (2020). Baricco, A.(2019). The game. Buenos Aires: Anagrama. *InterCambios. Dilemas y transiciones de la Educación Superior*, *7*(1).

Marcipar Katz, S., & Zanabria, C. (2007). ¿De qué hablamos cuando hablamos de cambios curriculares? *Revista Ciencias Económicas*, *1*(6), 79-88. https://doi.org/https://doi.org/10.14409/ce.v1i6.1105

Melnyk, N., Trachova, D., Kolesnikova, O., Demchuk, O., & Golub, N. (2020). Accounting trends in the modern world. *Independent Journal of Management & Production*, *11*(9), 2403-2416.

Muñoz, M. L. (2022). Chat GPT, un "monstruo""" tecnológico que "se metió por la ventana" y revolucionó el mundo [Último acceso 2 de abril de 2024]. https://www.ellitoral.com/educacion/chatgpt-inteligencia-artificial-escuelas-uso-santiago-bilinkis-entrevista_0_5InJ5lPEEg.html

Oppenheimer, A. (2013). *Cuentos Chinos: El engaño de Washington, la mentira populista y la esperanza de América Latina*. Debolsillo.

Ostengo, H. C. (2015). El Sistema de Información Contable.

Pahlen Acuña, R. J., Campo, A. M., Chaves, O. A., Chyrikins, H., Fronti de Gracıa, L., Helouani, R., & Viegas, J. C. (2011). Contabilidad: sistemas y procesos.

Roca, G. ((2015). *Las nuevas tecnololgías en niños y adolescentes. Guía para educar saludablemente en una sociedad digital*. Hospital Sant Joan de Déu (ed.)

Serres, M. (2014). *Pulgarcita*. Editorial Gedisa.

Summit, W. G. (2024). A Conversation with the Founder of NVIDIA: Who Will Shape the Future of AI [Último acceso 2 de abril de 2024]. https://www.youtube.com/watch?t=1109&v=8Pm2xEViNIo&feature=youtu.be

Sweigart, A. (2019). *Automate the boring stuff with Python: practical programming for total beginners*. No Starch Press.

Zumstein, F. (2021). *Python for Excel*. O'Reilly Media, Inc.