Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

alkemy

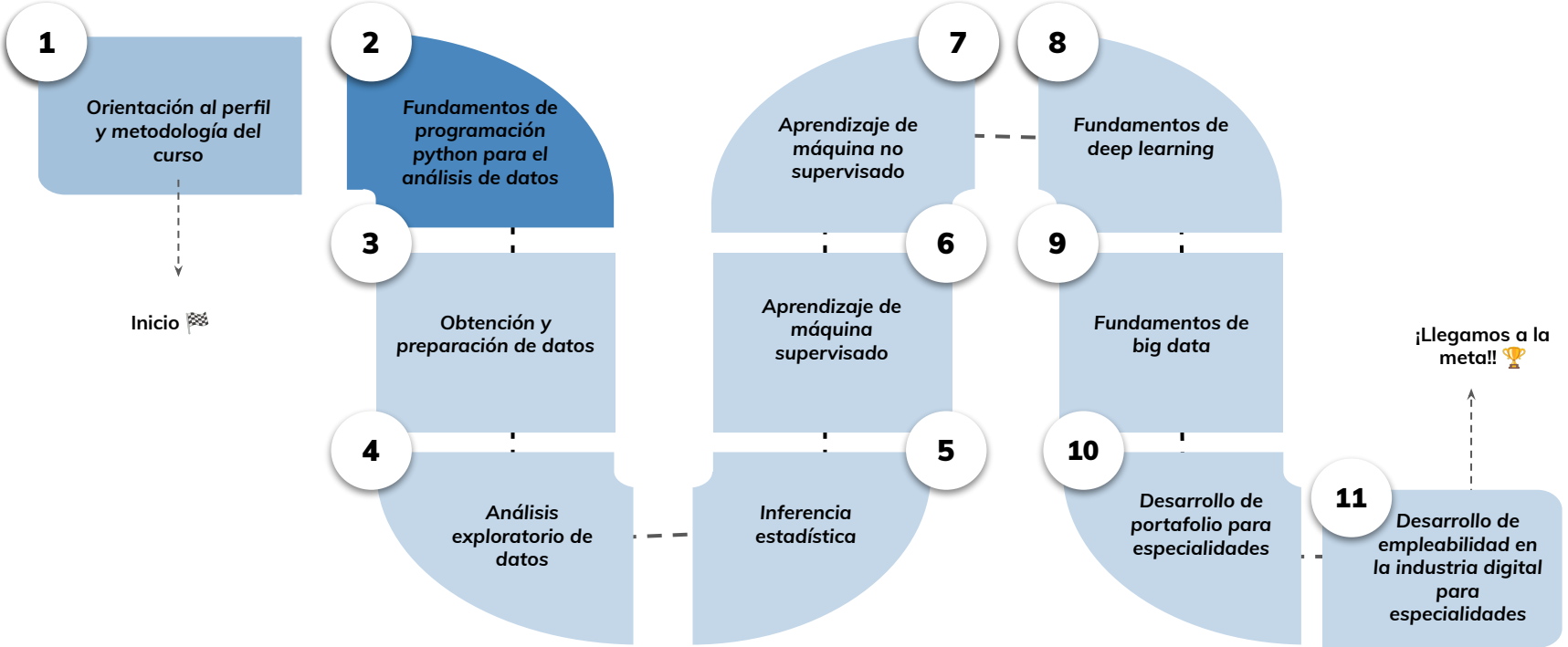
Clase grabada

El lenguaje Python- Parte 2

Aprendizaje Esperado 1: Explicar las principales herramientas del lenguaje Python para resolver distintas problemáticas en el entorno de trabajo.

Hoja de ruta

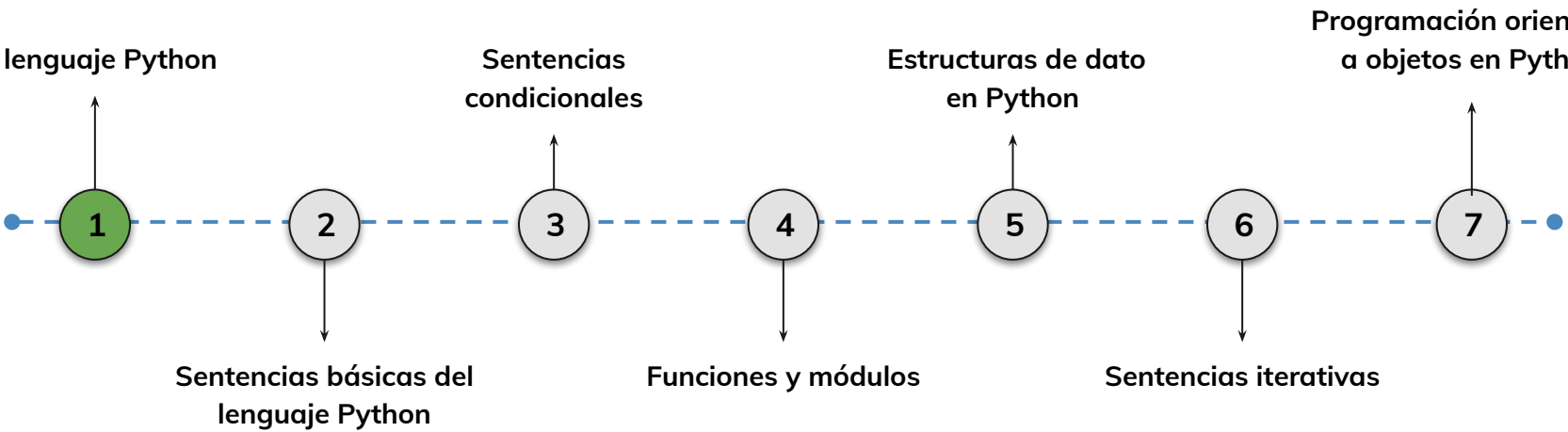
¿Cuáles skills conforman el programa? **Fundamentos de Ciencia de Datos**



alkemy

Roadmap de lecciones

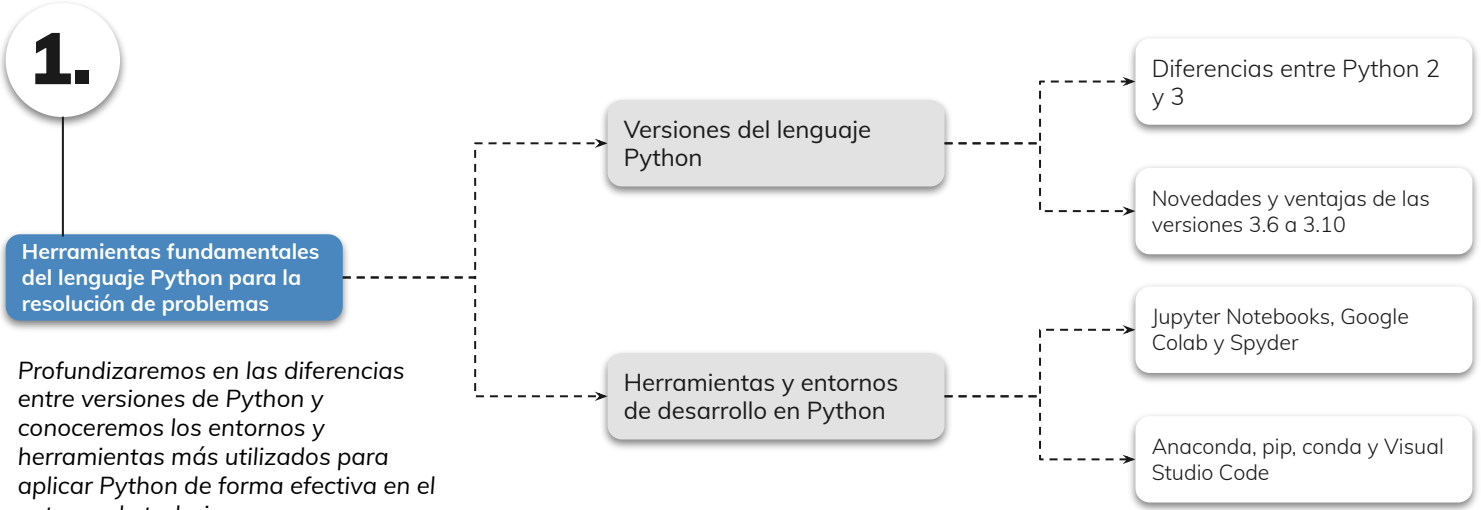
*¿Cuáles **lecciones** estaremos estudiando en este módulo?*



alkemy

Learning Path

¿Cuáles temas trabajaremos hoy?



Profundizaremos en las diferencias entre versiones de Python y conoceremos los entornos y herramientas más utilizados para aplicar Python de forma efectiva en el entorno de trabajo.

alkemy

Objetivos de aprendizaje

¿Qué aprenderás?

- Comprender las diferencias clave entre Python 2 y Python 3
- Reconocer las mejoras introducidas en las versiones más recientes del lenguaje
- Identificar las principales herramientas y entornos utilizados en proyectos con Python
- Aprender a gestionar entornos y librerías usando pip y conda
- Elegir el entorno adecuado según el tipo de problemática profesional

Repaso clase anterior

¿Quedó alguna duda?

En la clase anterior trabajamos 📚:

- → *Exploramos la historia, evolución y filosofía del lenguaje Python.*
- → *Identificamos sus ventajas como lenguaje multipropósito y de sintaxis clara.*
- → *Conocimos sus principales aplicaciones en ciencia de datos, desarrollo web y automatización.*
- → *Relacionamos problemáticas reales con herramientas específicas como Pandas, Flask o NumPy.*

El lenguaje Python

Instalación de Librerías

La facilidad para instalar librerías y módulos adicionales a través de herramientas como pip permite que Python sea un lenguaje altamente adaptable, capaz de abordar desde tareas básicas hasta proyectos complejos.

```
# Instalación de una librería con pippip install
pandas# Importar la librería en el códigoimport
pandas as pd# Usar la libreríadf =
pd.DataFrame({'A': [1, 2, 3]})print(df)
```

Portabilidad y Comunidad Activa

Python es un lenguaje multiplataforma, lo que significa que el código escrito en Python puede ejecutarse en diferentes sistemas operativos sin cambios. Esto es posible gracias a la implementación de Python en diferentes entornos, lo que permite una transición sencilla entre sistemas.

Además, la comunidad activa y global de Python contribuye constantemente con mejoras, soporte y documentación, lo que asegura que el lenguaje se mantenga actualizado y adaptable a nuevas tendencias tecnológicas.

Python 2 y Python 3: Diferencias Clave

Python 2 y Python 3 son las dos versiones principales del lenguaje, y aunque Python 2 dejó de recibir soporte en 2020, muchas aplicaciones heredadas aún se ejecutan en esta versión. Python 3 introdujo cambios significativos para mejorar la consistencia y simplicidad del lenguaje, aunque esto resultó en incompatibilidad con el código de Python 2.

Print

Python 2: print "Hola"

Python 3: print("Hola")

División

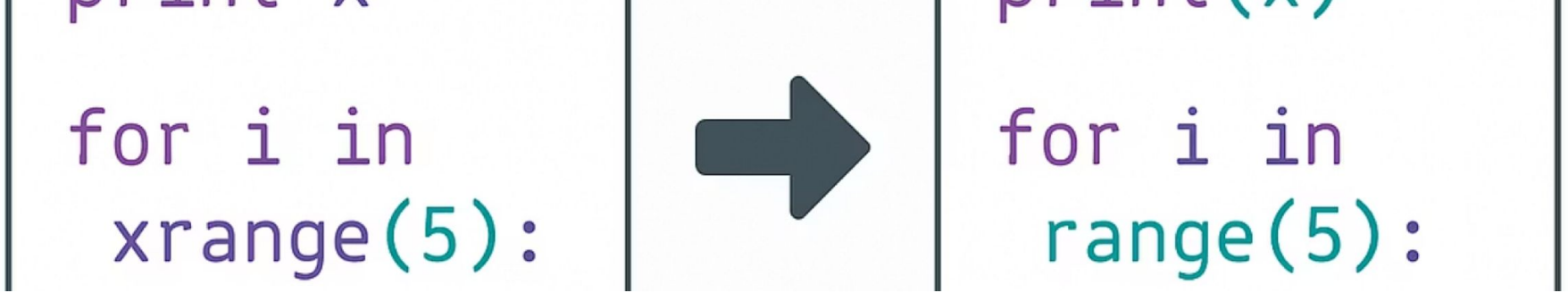
Python 2: 5/2 = 2

Python 3: 5/2 = 2.5

Unicode

Python 2: Strings ASCII por defecto

Python 3: Strings Unicode por defecto



Transición a Python 3

Python 3 se ha convertido en el estándar para nuevos desarrollos, mientras que la comunidad y las empresas han migrado progresivamente sus aplicaciones a esta versión. La transición fue importante para mejorar la escalabilidad, la eficiencia y la seguridad del lenguaje.

Versiones Actuales y Compatibilidad

Python 3 ha continuado evolucionando con versiones como 3.6, 3.7, 3.8 y posteriores, cada una de las cuales ha introducido mejoras de rendimiento, nuevas funcionalidades y corrección de errores. Cada nueva versión ha sido compatible con las anteriores, lo que facilita la actualización de proyectos sin cambios significativos en el código.

1. **Python 3.6 (2016)**
   Introdujo f-strings, variables de tipo anotado y mejoras en diccionarios
2. **Python 3.7 (2018)**
   Añadió módulos de datos, mejoras en tiempo de ejecución y optimizaciones
3. **Python 3.8 (2019)**
   Introdujo la expresión de asignación (walrus operator) y mejoras en typing
4. **Python 3.9 (2020)**
   Añadió operadores de fusión de diccionarios y mejoras en el sistema de tipos
5. **Python 3.10 (2021)**
   Introdujo match case y mejoras en el manejo de errores

Python 3.9 y 3.10: Innovaciones Recientes

Las versiones recientes de Python, como 3.9 y 3.10, han introducido características innovadoras, como la sintaxis de unión de tipos (tipo hinting), operadores mejorados para manipulación de diccionarios, y mejoras en el sistema de gestión de memoria.

La introducción de características avanzadas como los operadores de fusión en diccionarios (|) y las anotaciones de tipo mejoradas hacen que Python sea aún más eficiente y legible.

```
# Python 3.9: Operador de fusión de diccionariosdict1 = {"a": 1, "b": 2}dict2 = {"c": 3, "d": 4}combined = dict1 | dict2print(combined) # {"a": 1, "b": 2, "c": 3, "d": 4}# Python 3.10: Match casedef describe(status): match status: case 200: return "OK" case 404: return "Not Found" case _: return "Unknown"
```

Match Case en Python 3.10

La versión 3.10, en particular, introdujo match case, una estructura similar al switch case de otros lenguajes, lo que permite escribir código más organizado y controlado. Estas innovaciones son una muestra de cómo Python sigue adaptándose y mejorando.

```
# Ejemplo de match case en Python 3.10def analizar_comando(comando): match comando.split(): case ["salir"]: return "Saliendo del programa" case ["guardar", nombre]: return f"Guardando archivo como {nombre}" case ["abrir", nombre]: return f"Abriendo archivo {nombre}" case ["ayuda"]: return "Comandos disponibles: salir, guardar, abrir, ayuda" case _: return "Comando no reconocido"print(analizar_comando("guardar documento.txt"))
```

Beneficios de Actualizar a Nuevas Versiones

Rendimiento mejorado

Las versiones más recientes están optimizadas para un mejor uso de recursos y mayor velocidad de ejecución

Nuevas funcionalidades

Cada versión añade características que facilitan el desarrollo y mejoran la legibilidad del código

Seguridad

Las actualizaciones incluyen parches de seguridad y correcciones de vulnerabilidades

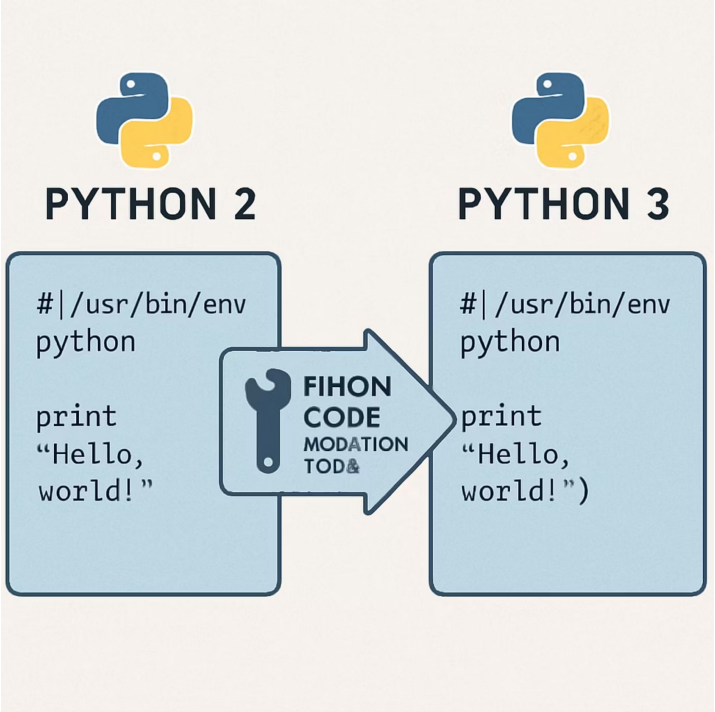
Compatibilidad con librerías

Las librerías modernas suelen requerir versiones recientes de Python, especialmente en áreas como machine learning

Estrategias para la Migración de Python 2 a Python 3

La migración de Python 2 a Python 3 puede ser compleja, especialmente en aplicaciones grandes. Sin embargo, existen herramientas y estrategias que facilitan esta transición.

La herramienta 2to3, por ejemplo, permite convertir automáticamente el código de Python 2 a Python 3, identificando y ajustando las diferencias sintácticas. Además, la comunidad proporciona guías y recursos para hacer el cambio de manera estructurada y segura.



alkemy

Pruebas en la Migración

Es importante realizar pruebas exhaustivas en el código migrado para asegurar su funcionalidad y rendimiento. Muchas organizaciones han migrado a Python 3 debido a las ventajas que ofrece en cuanto a compatibilidad, seguridad y eficiencia.

Análisis de código

Identificar áreas problemáticas que necesitarán cambios

Conversión automatizada

Utilizar herramientas como 2to3 para convertir el código

Revisión manual

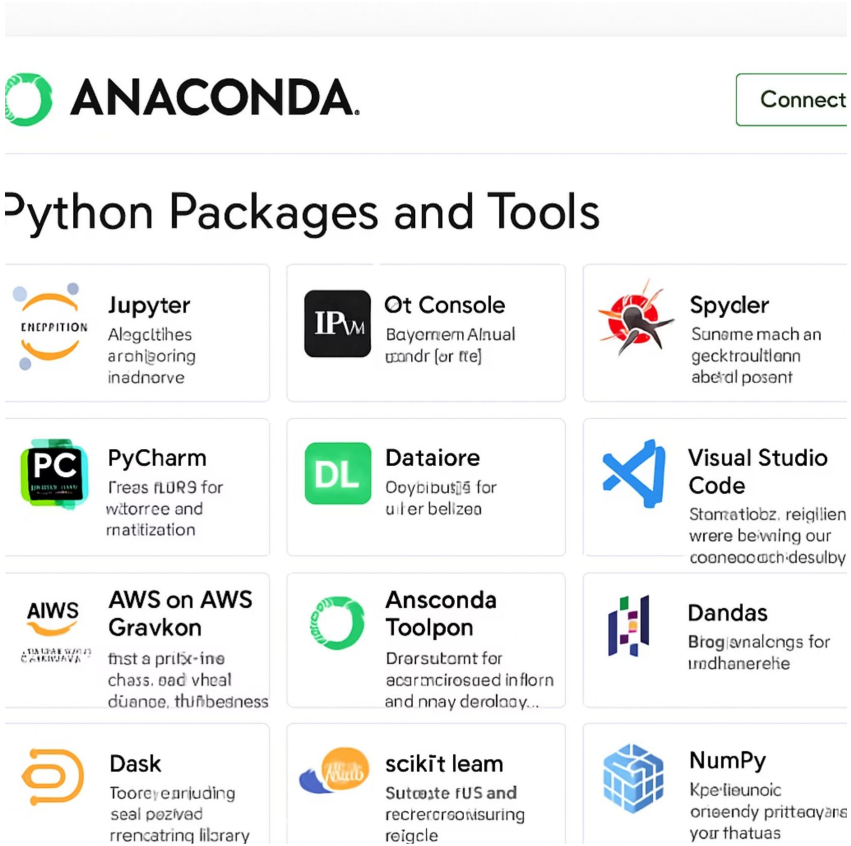
Verificar y ajustar el código convertido

Pruebas exhaustivas

Asegurar que la funcionalidad se mantiene después de la migración

El Entorno Anaconda

Anaconda es una distribución de Python ampliamente utilizada en ciencia de datos, análisis de datos y machine learning. Anaconda incluye herramientas esenciales como Jupyter Notebooks, Spyder y un administrador de paquetes que facilita la instalación de librerías y la gestión de entornos de desarrollo.



alkemy

Ventajas de Anaconda

Este entorno simplifica el proceso de configuración y es ideal para quienes trabajan con múltiples proyectos en Python, ya que permite administrar dependencias específicas para cada uno.

Gestión de paquetes

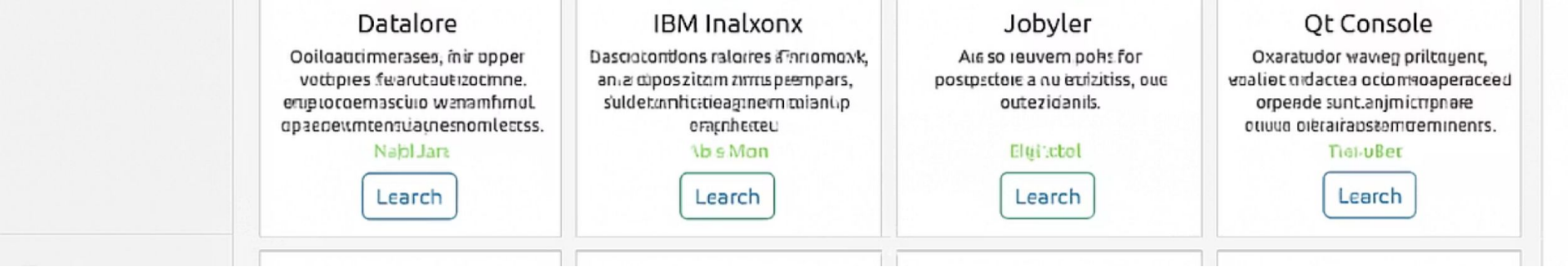
Instalación y actualización sencilla de librerías con conda

Entornos virtuales

Creación de entornos aislados para diferentes proyectos

Herramientas integradas

Incluye Jupyter, Spyder y otras herramientas populares



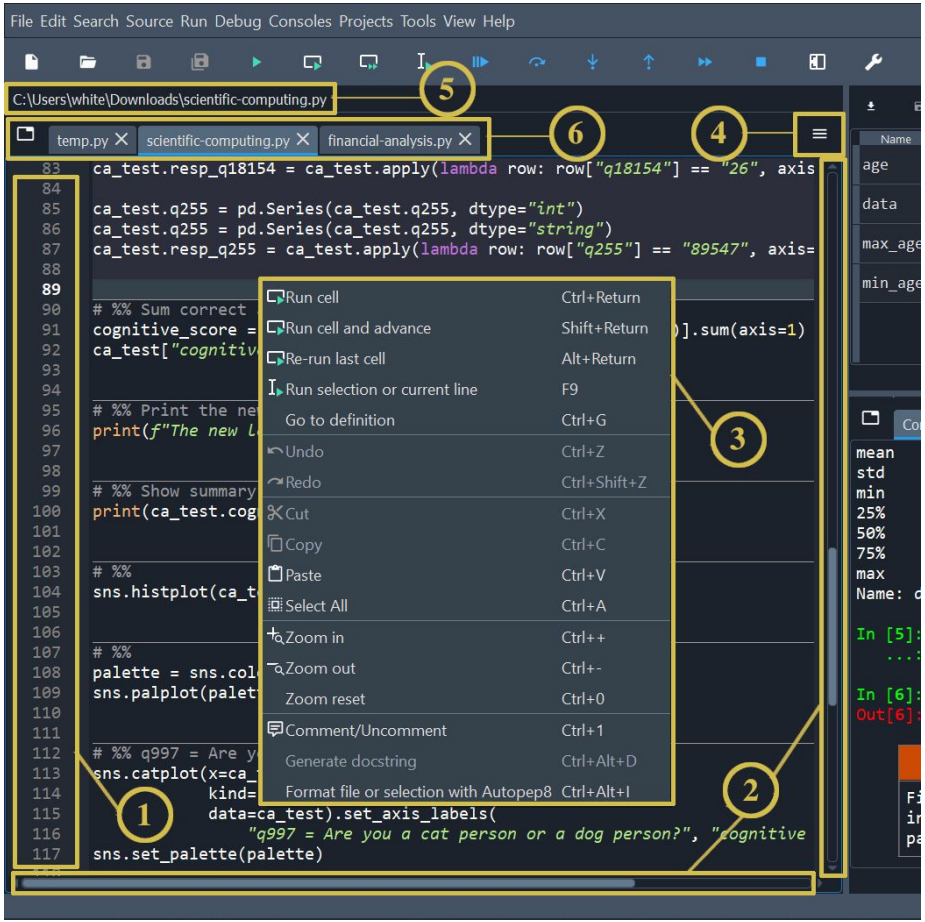
Anaconda Navigator

Anaconda también ofrece un entorno gráfico llamado Anaconda Navigator, que facilita el acceso a herramientas y configuraciones sin necesidad de usar la terminal. Esta accesibilidad lo convierte en una opción popular entre quienes desean una experiencia de instalación y administración simplificada.

El Editor Spyder

Spyder es un editor de Python especialmente diseñado para ciencia de datos y análisis científico. Parte de la distribución de Anaconda, Spyder incluye una interfaz intuitiva y herramientas integradas de depuración, autocompletado y ejecución de código en tiempo real.

Su diseño se asemeja al de MATLAB, lo que facilita su adopción para científicos y matemáticos familiarizados con este software.



alkemy

Características de Spyder

Entorno de desarrollo integrado

Editor de código, consola interactiva y explorador de variables en una sola interfaz

Explorador de variables

Permite examinar y modificar variables en tiempo de ejecución

Depurador integrado

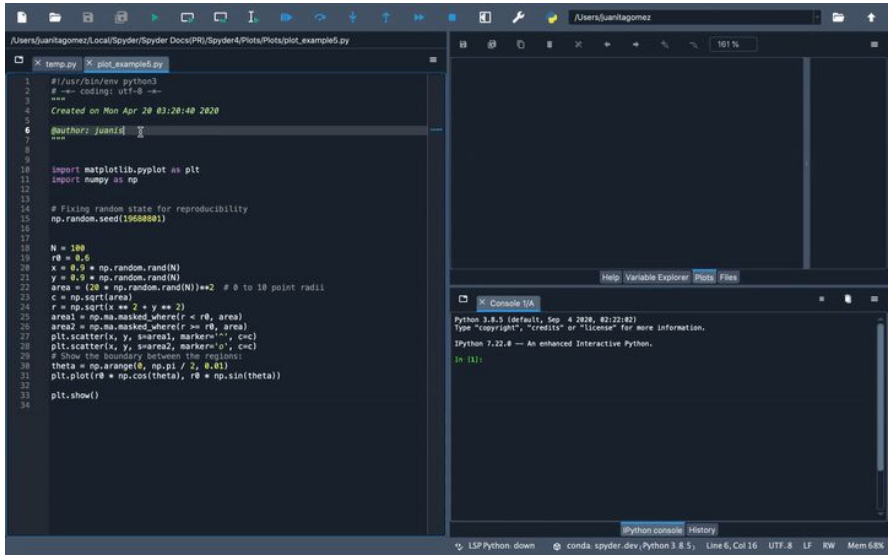
Facilita la identificación y corrección de errores en el código

Integración con librerías científicas

Soporte nativo para NumPy, Pandas, Matplotlib y otras librerías de análisis de datos

Paneles de Spyder

Spyder permite dividir el entorno de trabajo en múltiples paneles, lo que facilita la visualización y el análisis de datos en tiempo real. La integración con librerías como NumPy, Pandas y Matplotlib convierte a Spyder en una herramienta versátil para realizar cálculos y generar visualizaciones.

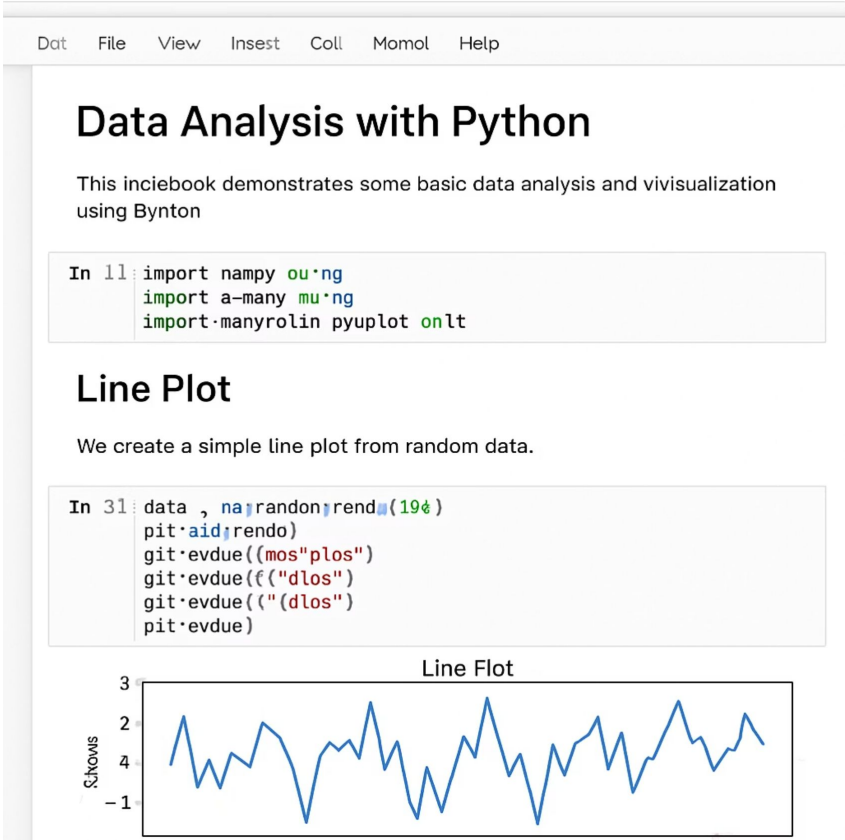


alkemy

Jupyter Notebooks

Jupyter Notebooks es una herramienta interactiva que permite escribir y ejecutar código Python en celdas independientes, lo que facilita la experimentación, el análisis y la visualización de datos.

Los Notebooks son ampliamente utilizados en ciencia de datos, machine learning e investigación académica, ya que permiten documentar el proceso y los resultados en un solo archivo que incluye código, visualizaciones y anotaciones.



alkemy

Ventajas de Jupyter Notebooks

Ejecución interactiva

Permite ejecutar bloques de código independientes y ver los resultados inmediatamente

Documentación integrada

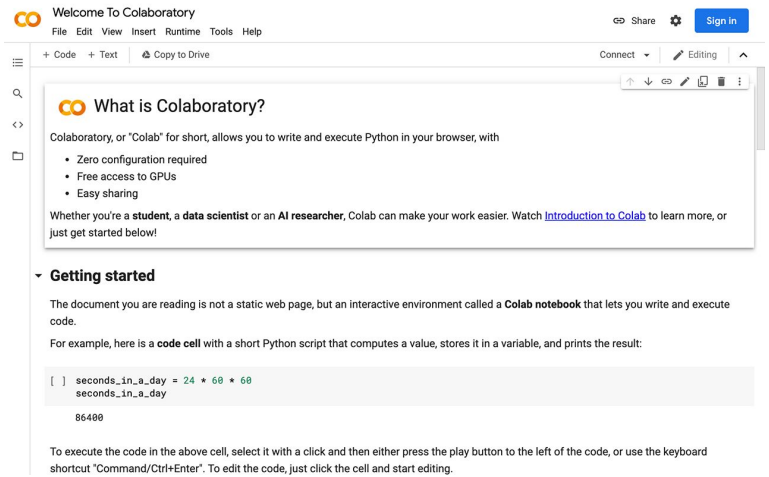
Combina código con texto explicativo en formato Markdown

Visualización en línea

Muestra gráficos y visualizaciones directamente en el documento

Compartible

Fácil de compartir como archivo .ipynb o exportar a otros formatos como HTML o PDF



Google Colab

Google Colab es una plataforma gratuita basada en Jupyter Notebooks, que permite ejecutar código Python en la nube. Esto es especialmente útil para proyectos de machine learning que requieren una gran cantidad de procesamiento, ya que Colab ofrece acceso a GPUs y TPUs, lo que permite realizar cálculos intensivos sin necesidad de un equipo de alto rendimiento.

alkemy

Características de Google Colab

Computación en la nube

Ejecuta código Python en servidores de Google sin necesidad de configuración local

Aceleración por hardware

Acceso gratuito a GPUs y TPUs para acelerar tareas de machine learning

Integración con Google Drive

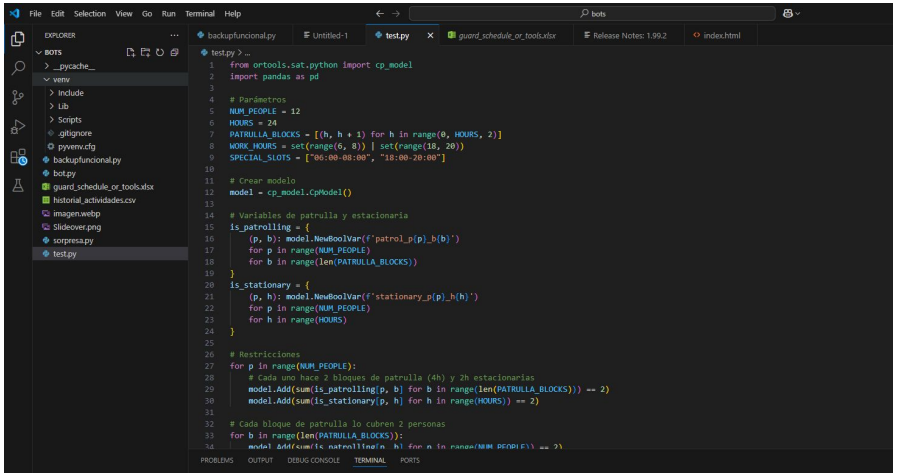
Permite almacenar y acceder a datos desde la nube fácilmente

Colaboración en tiempo real

Múltiples usuarios pueden editar y ejecutar el mismo notebook simultáneamente

Visual Studio Code

Visual Studio Code (VS Code) es un editor de código popular, desarrollado por Microsoft, que ofrece soporte para múltiples lenguajes de programación, incluida Python. Con la extensión de Python, VS Code proporciona herramientas avanzadas como depuración, autocompletado y análisis de código en tiempo real.



alkemy

Personalización de VS Code

VS Code es altamente personalizable, permitiendo a los usuarios instalar extensiones y configurar su entorno de desarrollo según sus necesidades. VS Code es una excelente opción para quienes trabajan en proyectos complejos que requieren una integración continua y control de versiones.

Extensiones

Miles de extensiones disponibles para personalizar la funcionalidad

Configuración

Ajustes detallados para adaptar el editor a tu flujo de trabajo

Temas

Personalización visual con diferentes temas y esquemas de colores

Control de versiones

Integración nativa con Git y otros sistemas de control de versiones

Extensiones de Python para VS Code

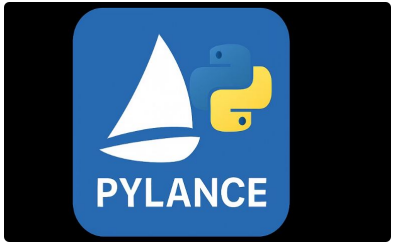
Gracias a su flexibilidad y potencia, VS Code es uno de los editores preferidos en el desarrollo profesional con Python.

Python

Soporte básico para Python con IntelliSense, linting y depuración

Pylance

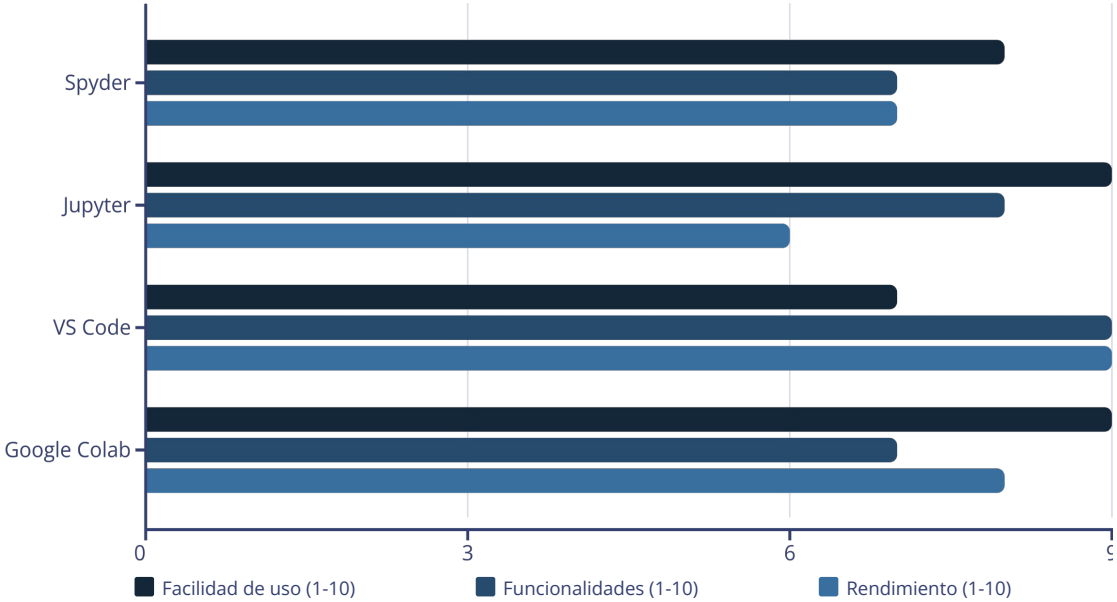
Servidor de lenguaje que mejora el autocompletado y análisis de tipos



Jupyter

Soporte para notebooks de Jupyter directamente en VS Code

Comparación de Entornos de Desarrollo



Cada entorno tiene sus fortalezas y es adecuado para diferentes tipos de proyectos y usuarios.
La elección dependerá de tus necesidades específicas y preferencias personales.

alkemy

Elección del Entorno Adecuado

Para principiantes

Anaconda con Spyder o Jupyter Notebooks ofrece una experiencia integrada y fácil de configurar

Para ciencia de datos

Jupyter Notebooks o Google Colab son ideales para análisis exploratorio y visualización

Para desarrollo profesional

VS Code proporciona un entorno completo con herramientas avanzadas para proyectos complejos

Para recursos limitados

Google Colab permite ejecutar código intensivo sin necesidad de hardware potente

Resumen:
El Lenguaje Python

Python es un lenguaje de programación que ha revolucionado el desarrollo de software, tanto por su simplicidad como por su versatilidad y potencia. Desde sus inicios en la década de 1980 hasta su posición actual como uno de los lenguajes más populares, Python ha evolucionado para satisfacer las necesidades de múltiples industrias y aplicaciones, desde la web y la automatización hasta la ciencia de datos y la inteligencia artificial.

alkemy

Comunidad y Recursos

Con una comunidad activa y recursos educativos disponibles, Python sigue siendo accesible y adaptable para desarrolladores de todos los niveles de experiencia. La elección del entorno de trabajo adecuado es fundamental para aprovechar al máximo las capacidades de Python.

Herramientas como Anaconda, Spyder, Jupyter Notebooks, Google Colab y VS Code facilitan el desarrollo, prueba y optimización de aplicaciones, permitiendo que los desarrolladores elijan el entorno que mejor se adapte a su flujo de trabajo. Estas herramientas no solo mejoran la productividad, sino que también ofrecen una experiencia de desarrollo rica y flexible.

#Momentode Preguntas...

¿?

¿Qué diferencias existen entre Python 2 y Python 3 y por qué son relevantes hoy?

¿?

¿Qué mejoras trajo Python 3.10 respecto a versiones anteriores?

¿?

¿Qué ventajas ofrece Google Colab frente a otros entornos?

¿?

¿Cuándo conviene usar Jupyter, VS Code o Spyder?

Live Coding

¿En qué consistirá la Demo?

Breve descripción

1. *Descripción*
2. *Descripción*
3. *Descripción*
4. *Descripción*
5. *Descripción*
6. *Descripción*
7. *Descripción*
8. *Descripción*

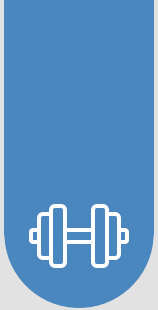
Tiempo:

alkemy

Momento:

Time-out!

⏰5 -10 min.



Ejercicio N° 1

Creando desde 0

Creando desde 0

Contexto: 🙌

Estás en un equipo de trabajo que resuelve problemas diversos: análisis de datos, automatización, visualización y desarrollo de notebooks colaborativos. Cada integrante necesita una herramienta distinta para abordar su problemática con Python.

Consigna: ✍️

Explicar las principales herramientas del lenguaje Python para resolver distintas problemáticas en el entorno de trabajo.

Para cada una de las **siguientes problemáticas** , debes indicar:

1. ¿Qué herramienta usarías?
2. ¿Por qué la elegirías?
3. ¿Cómo la instalarías o accederías?
4. ¿Qué ventajas tiene frente a otras opciones?

Problemáticas a resolver:

a) Automatización de tareas repetitivas

b) Visualización de datos y análisis exploratorio

c) Trabajo colaborativo con notebooks

d) Prototipado de scripts con acceso a múltiples entornos

Tiempo 🕐**:** *35 Minutos*

Creando desde 0

Paso a paso: ⚙️

- Identificar la problemática y su necesidad técnica
- Asociarla con la herramienta de Python más adecuada
- Justificar con fundamentos técnicos claros
- Escribir un ejemplo concreto de uso o instalación

¿Alguna consulta?

alkemy

Resumen

¿Qué logramos en esta clase?

- ✓ ***Entendimos las diferencias fundamentales entre Python 2 y 3***
- ✓ ***Exploramos las novedades y mejoras desde Python 3.6 hasta 3.10***
- ✓ ***Conocimos herramientas clave como Jupyter, Colab, Anaconda y VS Code***
- ✓ ***Aprendimos a elegir herramientas según el problema a resolver***

¡Ponte a prueba!

Momento de ejercitación

Te invitamos a aprovechar esta última sección del espacio sincrónico para realizar de manera individual las ***actividades disponibles en la plataforma.*** *Estas propuestas son claves para afianzar lo trabajado y* ***forman parte obligatoria del recorrido de aprendizaje.***

💪 Análisis de caso ——— 💪 Selección Múltiple

💪 Comprensión lectora

Si al resolverlas surge alguna duda, compartela o tráela al próximo encuentro sincrónico.

alkemy

¡Muchas gracias!

alkemy