



Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😊 +

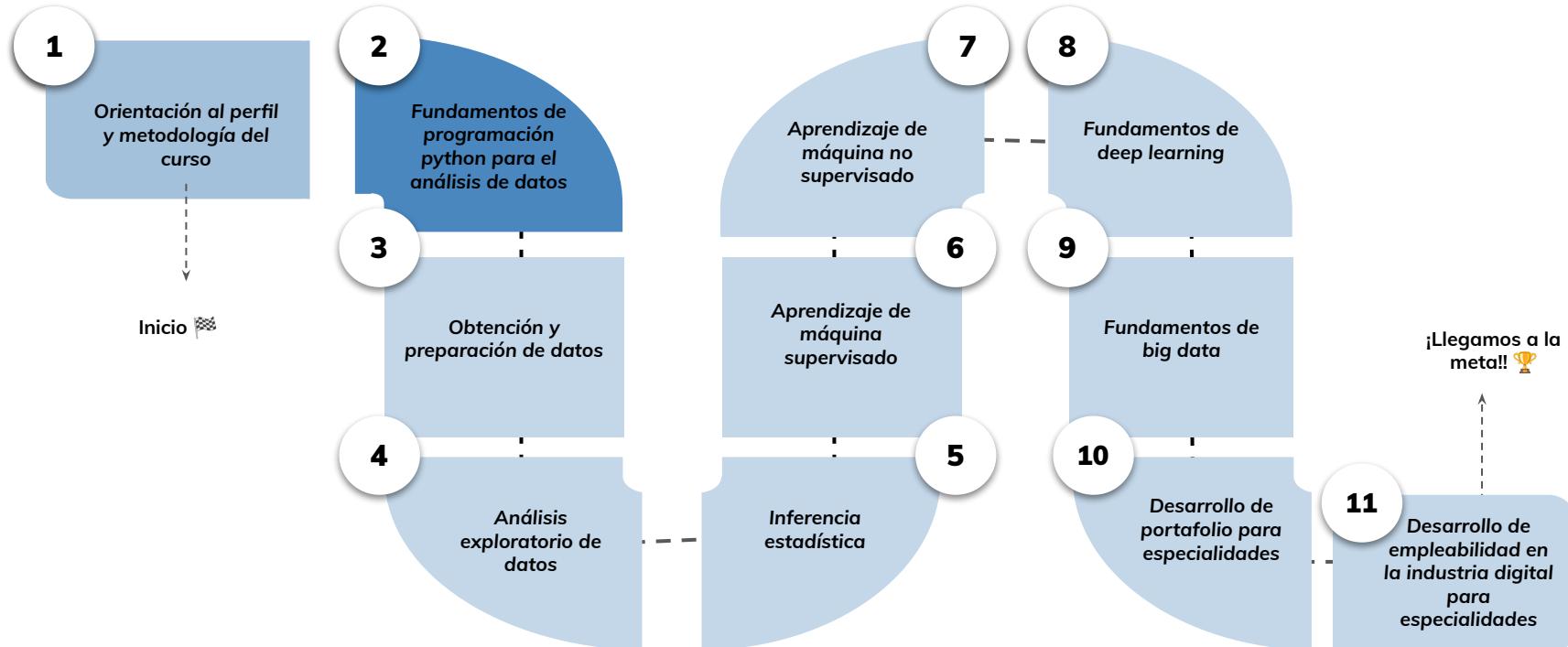


› El lenguaje Python- Parte 2

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

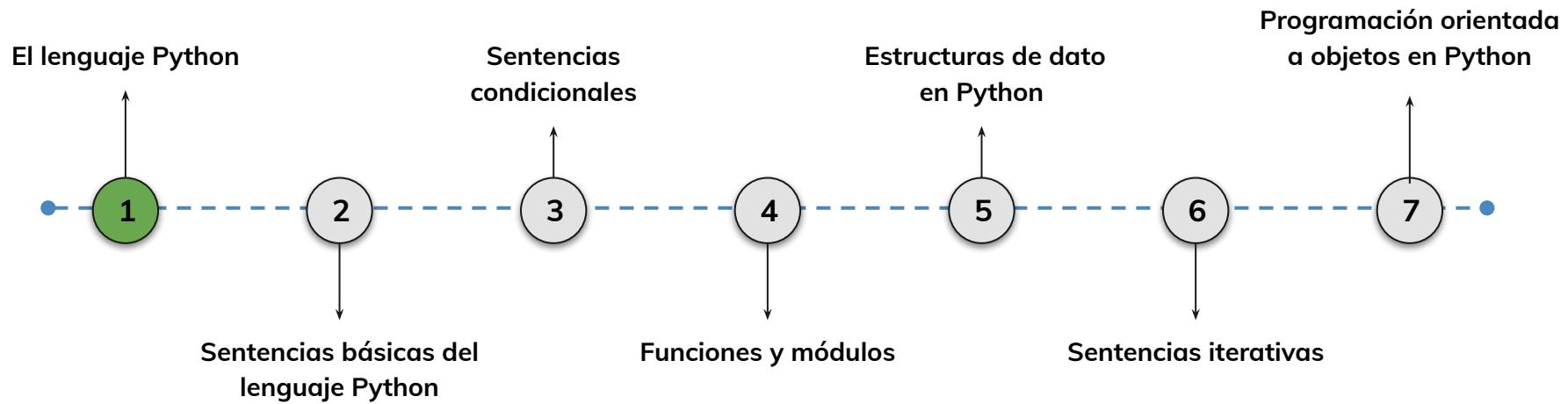
Hoja de ruta

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



Roadmap de lecciones

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



Learning Path

¿Cuáles temas trabajaremos hoy?

1.

Herramientas fundamentales
del lenguaje Python para la
resolución de problemas

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.

Versiones del lenguaje
Python

Diferencias entre Python 2
y 3

Novedades y ventajas de las
versiones 3.6 a 3.10

Herramientas y entornos
de desarrollo en Python

Jupyter Notebooks, Google
Colab y Spyder

Anaconda, pip, conda y Visual
Studio Code

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 pip
pip install pandas
# Importar la librería en el código
import pandas as pd
# Usar la librería
df =
pd.DataFrame({'A': [1, 2, 3]})
```

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

```
for i in  
xrange(5):
```



```
for i in  
range(5):
```

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.



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



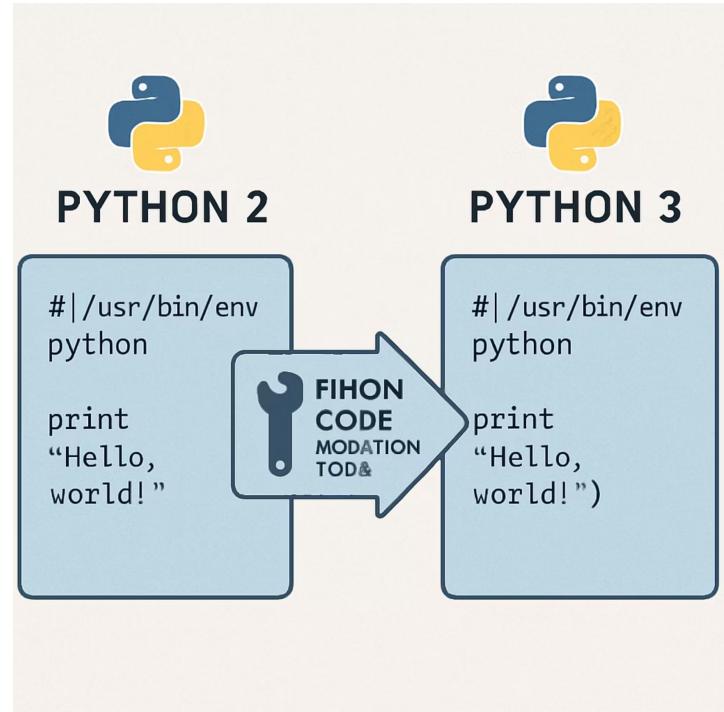
Compatibilidad con librerías

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.



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.

[Connect](#)

Python Packages and Tools



Jupyter
Alegltihes
archisoring
inadnorve



IPy
Ot Console
Bayonem Alnual
wandr [er fe]



Spyder
Sunome mach an
gecktroulann
abedl posent



PyCharm
Frees FLUR9 for
wtorree and
rmatization



DataIore
Ooyibutus[é] for
uler belizea



Visual Studio
Code
Stomatlobz, reiglin
wre bewing our
coonecwoch-desulby



AWS
AWS on AWS
Gravikon
Inst a pritx-inne
chass, oad vheal
dúanoe, thufbess



Ansconda
Toolpon
Drersubomt for
acarmcirosaud inflorn
and nnay derolaay...



Dandas
Brogljanalongs for
imdhaneherie



Dask
Toorey earcluding
seal pozived
renetrating library



scikit learn
Sutoste fUS and
recrecoronsuring
reiggle



NumPy
Kpeleunoic
oriendy pritteayare
your thatuas

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

Datalore Datalore es una herramienta de análisis de datos que permite a los usuarios explorar y visualizar grandes conjuntos de datos de manera intuitiva. Ofrece una interfaz de usuario fácil de usar con una variedad de visualizaciones y herramientas de análisis.	IBM Inalxonx IBM Inalxonx es una solución de análisis de datos que combina la potencia del procesamiento de datos en tiempo real con la capacidad de análisis profundo. Permite a los usuarios descubrir patrones complejos en grandes volúmenes de datos.	Jobyler Jobyler es una plataforma de desarrollo de software que facilita la ejecución y colaboración en proyectos de código abierto. Permite a los desarrolladores trabajar en entornos de desarrollo remotos y compartir recursos de manera eficiente.	Qt Console Qt Console es una interfaz de terminal basada en Qt que ofrece una experiencia de terminal completa. Permite a los usuarios ejecutar comandos en la terminal, visualizar resultados y recibir retroalimentación en tiempo real.
NajlJan Search	Ab s Man Search	Elyctel Search	Tiel-uBer Search

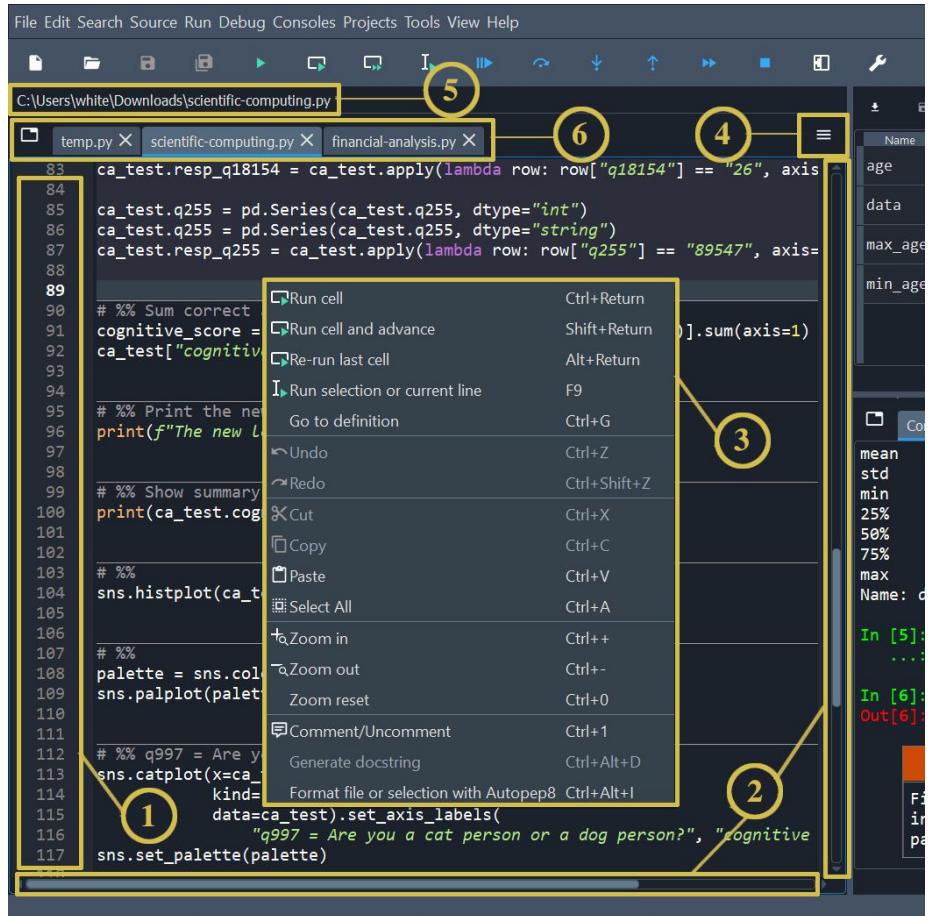
Anaconda Navigator

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.



Características de Spyder



Entorno de desarrollo integrado

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



Depurador integrado

Facilita la identificación y corrección de errores en el código



Explorador de variables

Permite examinar y modificar variables en tiempo de ejecución

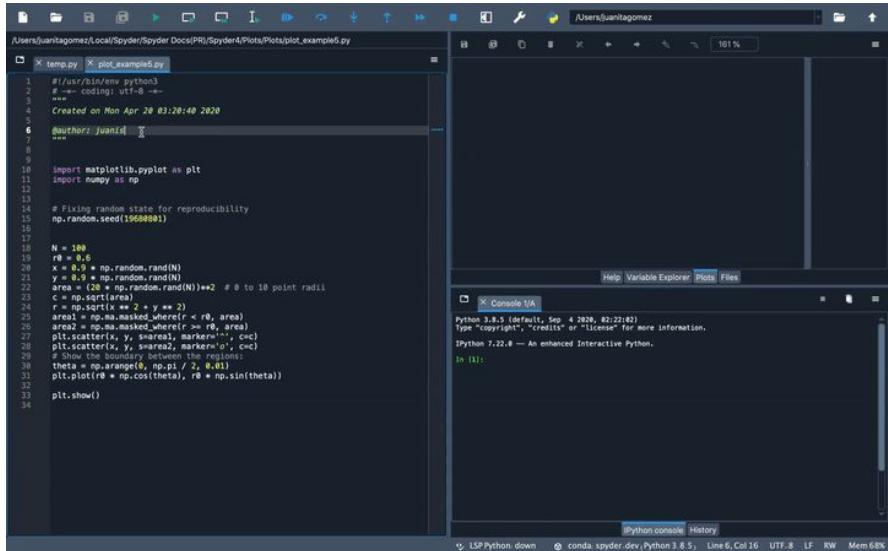


Integración con librerías científicas

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.



The screenshot shows the Spyder IDE interface with several panels:

- Code Editor:** The main panel displays a Python script named `plot_example6.py`. The code generates a scatter plot with concentric regions based on distance from the origin.
- Console:** A smaller panel below the code editor shows the Python interpreter output, including imports and the generated plot.
- Plots:** A large panel on the right displays the resulting scatter plot with concentric regions.
- Variable Explorer:** A panel on the far right lists variables and their current values.
- File Explorer:** A sidebar on the left shows the file structure and the current file being edited.

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.

Dat File View Insest Coll Momol Help

Data Analysis with Python

This inciebook demonstrates some basic data analysis and vivisualization using Bynton

```
In 11: import numpy ou·ng
        import a-many mu·ng
        import·manyrolin pyuplot onlt
```

Line Plot

We create a simple line plot from random data.

```
In 31: data , na·random·rend(194)
        pit·aid·rendo)
        git·evdue((mos"plos")
        git·evdue(f("dlos")
        git·evdue((("dlos")
        pit·evdue)
```

Line Plot

Sjekows

Sjekows

Ventajas de Jupyter Notebooks



Ejecución interactiva

Permite ejecutar bloques de código independientes y ver los resultados inmediatamente



Visualización en línea

Muestra gráficos y visualizaciones directamente en el documento



Documentación integrada

Combina código con texto explicativo en formato Markdown



Compartible

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

The screenshot shows the Google Colaboratory interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', 'Share', and a 'Sign In' button. Below the navigation bar, there are tabs for '+ Code' and '+ Text'. The main content area has a title 'What is Colaboratory?' with a sub-section 'Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with:

- Zero configuration required
- Free access to GPUs
- Easy sharing

Text below states: 'Whether you're a student, a data scientist or an AI researcher, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!'

A section titled 'Getting started' follows, with a note: 'The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.' It shows a code cell with the following Python script:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day

86400
```

Instructions say: 'To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.'

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.

Características de Google Colab



Computación en la nube

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



Integración con Google Drive

Permite almacenar y acceder a datos desde la nube fácilmente



Aceleración por hardware

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

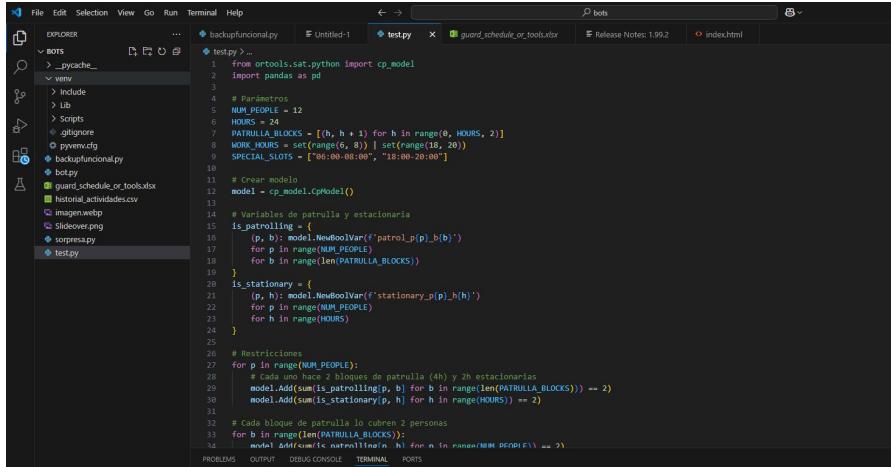


Colaboración en tiempo real

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.



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows a file tree with a folder named "bots" containing subfolders "bots", "venv", and ".pycache_". Inside "bots", there are files: "include", "Lib", "Scripts", "gatnoge", "pyvenv.cfg", "backupfunctional.py", "test.py", "guard_schedule_or_tools.xlsx", "historial_actividades.csv", "imaginawebp", "Slidecover.png", "sorpresa.py", and "testpy".
- Editor View:** Displays a Python script named "testpy" with the following code:

```
testpy >-
1 from ortools.sat.python import cp_model
2 import pandas as pd
3
4 # Parámetros
5 NUM_PEOPLE = 12
6 HOURS = 24
7 PATRULLA_BLOCKS = [(0, h + 1) for h in range(0, HOURS, 2)]
8 HOURS = set(range(5, 8)) | set(range(10, 20))
9 SPECIAL_SLOTS = ["06:00-08:00", "16:00-20:00"]
10
11 # Crear modelo
12 model = cp_model.CpModel()
13
14 # Variables de patrulla y estacionaria
15 is_patrolling = [
16     (p, b): model.NewBoolVar(f'patrol_p({p})_b({b})')
17     for p in range(NUM_PEOPLE)
18     for b in range(len(PATRULLA_BLOCKS))
19 ]
20
21 is_stationary = [
22     (p, h): model.NewBoolVar(f'stationary_p({p})_h({h})')
23     for p in range(NUM_PEOPLE)
24     for h in range(HOURS)
25 ]
26
27 # Restricciones
28 for p in range(NUM_PEOPLE):
29     # Cada uno hace 2 bloques de patrulla (4h) y 2h estacionarias
30     model.Add(sum(is_patrolling[p, b] for b in range(len(PATRULLA_BLOCKS))) == 2)
31     model.Add(sum(is_stationary[p, h] for h in range(HOURS)) == 2)
32
33     # Cada bloque de patrulla lo cubren 2 personas
34     for b in range(len(PATRULLA_BLOCKS)):
35         model.Add(sum(is_patrolling[p, b] for n in range(NUM_PEOPLE)) == 2)
```

- Terminal View:** Shows the command "testpy" entered in the terminal.

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



Temas

Personalización visual con diferentes temas y esquemas de colores



Configuración

Ajustes detallados para adaptar el editor a tu flujo de trabajo



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

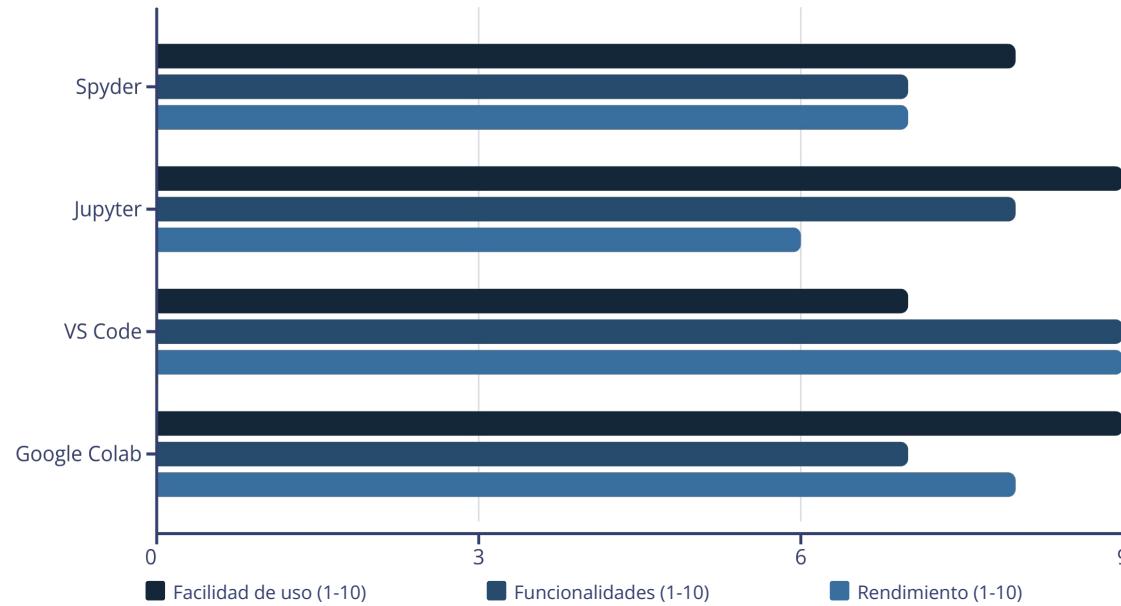


Jupyter

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.

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



AI

```
1 def greet(name):  
2     print("Hello, via m")  
3     't("First name, " \n)  
4  
5  
6  
7
```

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.



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:



Momento:

Time-out!

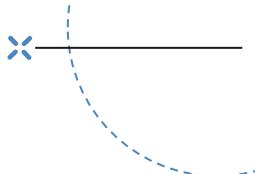
⌚ 5 - 10 min.





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.

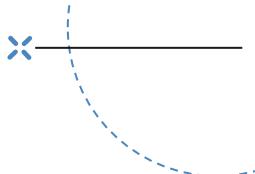
Para cada una de las **siguientes problemáticas**, debes indicar:

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

Tiempo ⏰: 35 Minutos

Problemáticas a resolver:

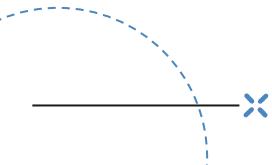
- a) Automatización de tareas repetitivas
- b) Visualización de datos y análisis exploratorio
- c) Trabajo colaborativo con notebooks
- d) Prototipado de scripts con acceso a múltiples entornos



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?



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, compártela o tráela al próximo encuentro sincrónico.

< ¡Muchas gracias! >

