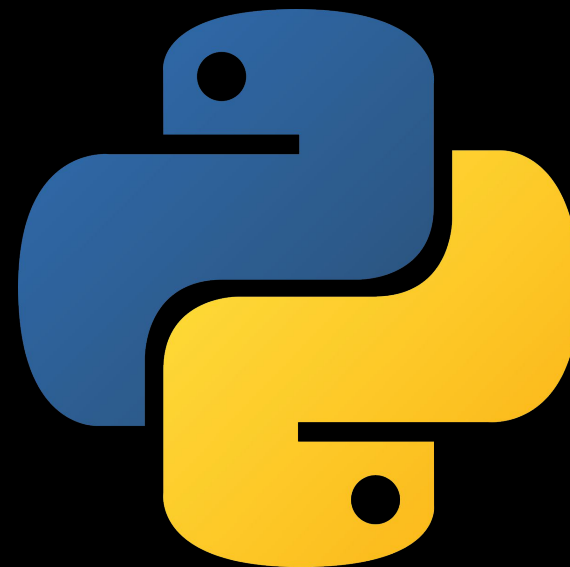


¿Qué es Python?

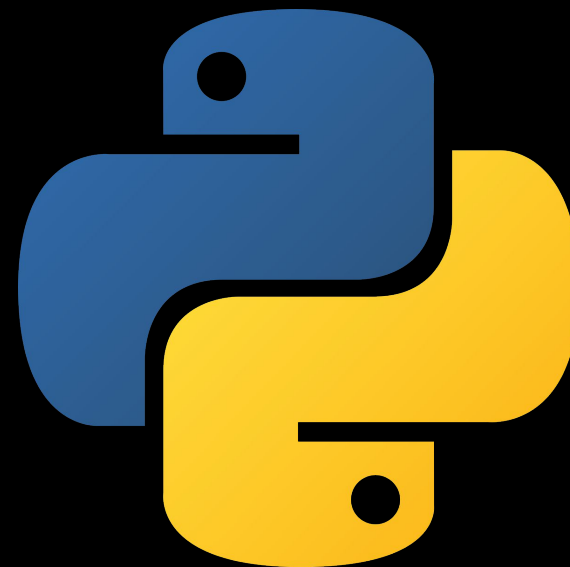
# ¿Qué es Python?

**Python** es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código (Wikipedia)



# ¿Qué es Python?

1. Lenguaje de programación flexible
2. Diseñado para ser fácil de leer





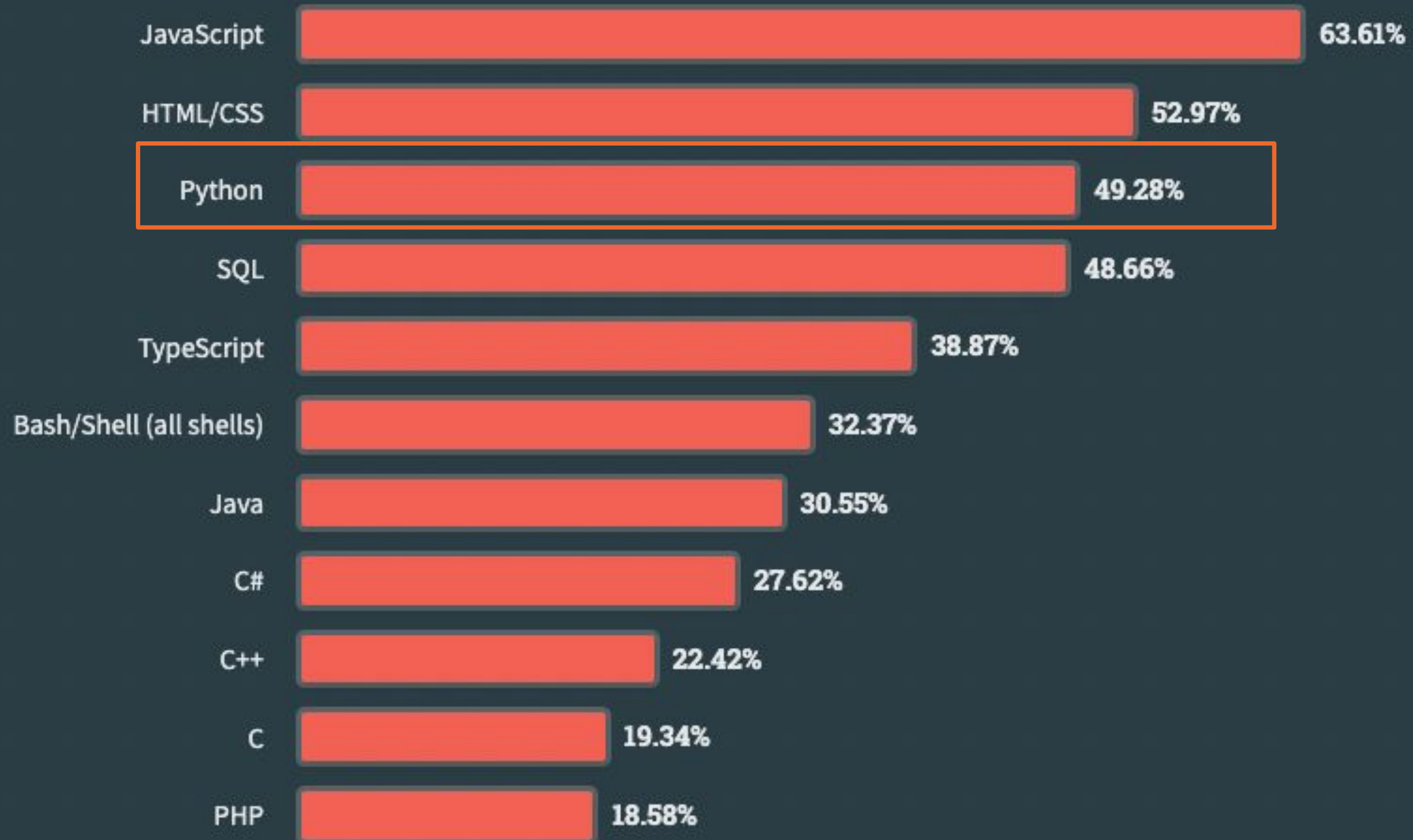
**Creado por Guido van  
Rossum en 1991**

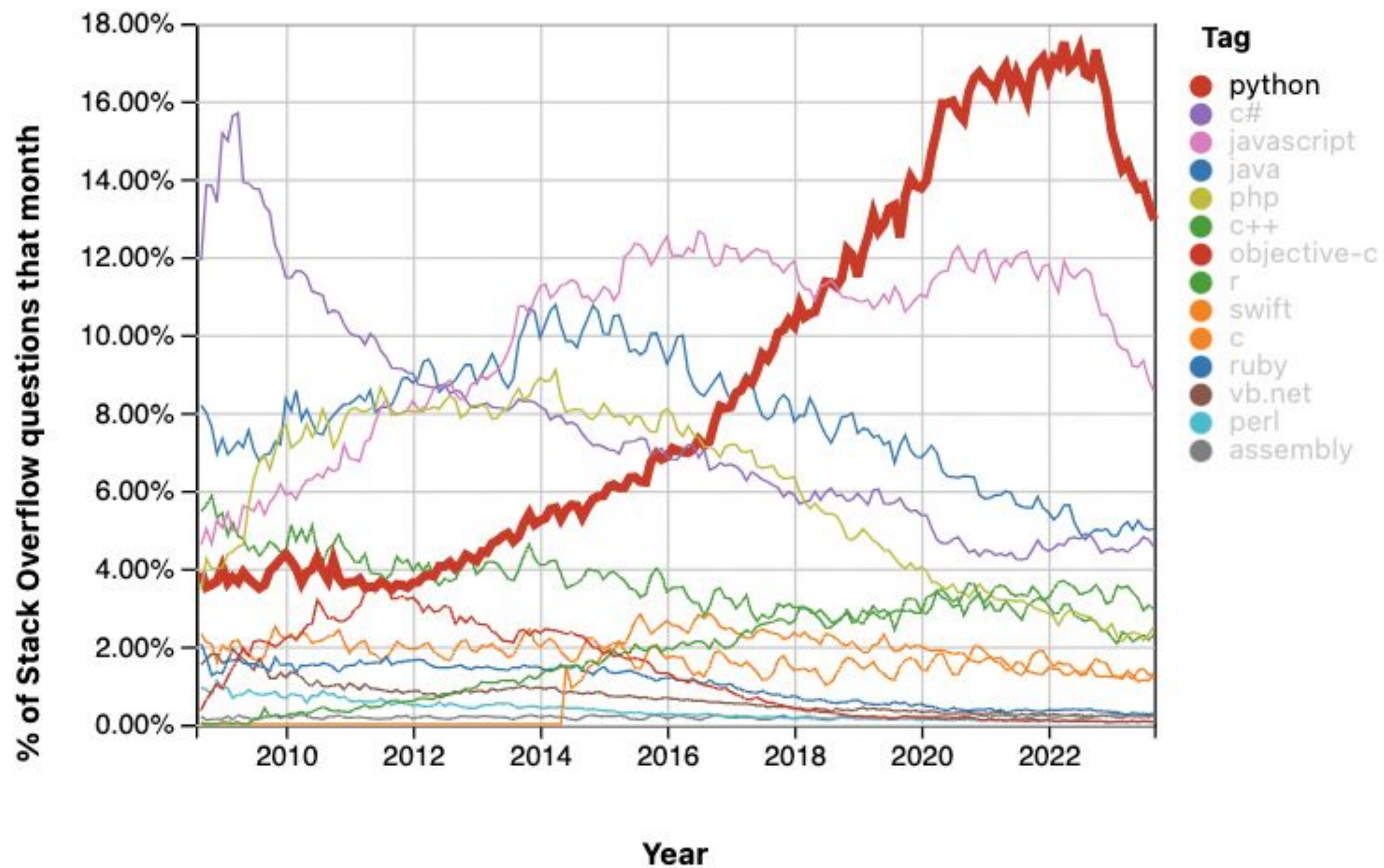
**Nombrado en honor al grupo de comedia  
británico Monty Python.**






# Stackoverflow 2023 Developer Survey





# ChatGPT usa Python

 Trabajo finalizado Ocultar detalles de ejecución ^

```
python

# Importing necessary libraries for data analysis and visualization
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Reading the dataset
file_path = '/mnt/data/iris.csv'
df = pd.read_csv(file_path)

# Displaying the first few rows of the dataset
df.head()
```

Copy code





# ¿Por qué Python?



# ¿Por qué **Python**?

- 1. Fácil de aprender para principiantes**
- 2. Amplia comunidad y soporte**
- 3. Gran cantidad de librerías y frameworks**
- 4. Versátil y multiplataforma**

¿Para qué se usa **Python**?



# ¿Para qué se usa **Python**?

1. Desarrollo web (Django, Flask)
2. Análisis de datos (Pandas, NumPy y más)
3. Inteligencia Artificial y Machine Learning
4. Automatización de tareas y scripting





## Python 2

```
print 'Hola, Mundo!'
```

## Python 3

```
print('Hola, Mundo!')
```

# La filosofía de Python

## El Zen de Python

Bello es mejor que feo.  
Explícito es mejor que implícito.  
Simple es mejor que complejo.  
Complejo es mejor que complicado.  
Plano es mejor que anidado.  
Espaciado es mejor que denso.  
La legibilidad es importante.



```
sebastiantunnell — python — 86x27
(base) Sebs-MacBook-Air-2:~ sebastiantunnell$ python
Python 3.9.16 (main, Mar  8 2023, 04:29:24)
[Clang 14.0.6 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>> █
```

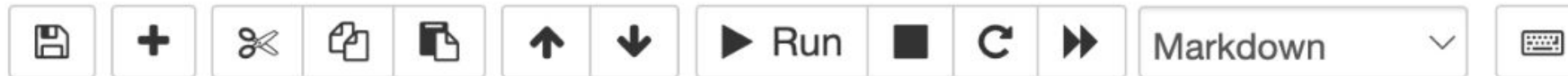


# ¿Qué es un Jupyter Notebook?

# ¿Qué es un **Jupyter Notebook**?

Un **Jupyter Notebook** es una aplicación web de código abierto que te permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo.





# My first Jupyter Notebook

This is Markdown.


In [1]: `print("Hello World!")`

Hello World!

In [2]: `import time`  
`time.sleep(10)`

Cell colour:

- Green outline — cell is in “edit mode”
- Blue outline — cell is in “command mode”

Welcome To Colaboratory

FileEditViewInsertRuntimeToolsHelp



Share

Table of contents

Getting started

{x}Data science

Machine learning

More Resources

Featured examples

+ Section


+ Code+ Text

Copy to Drive

Connect

# Welcome to Colab!

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette.



## What is Colab?

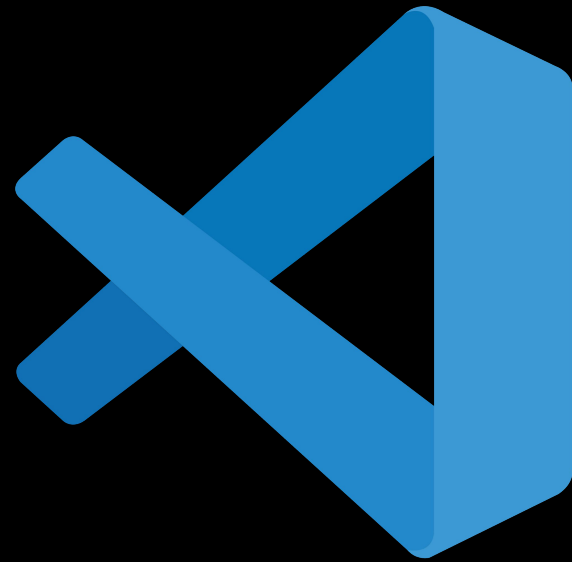
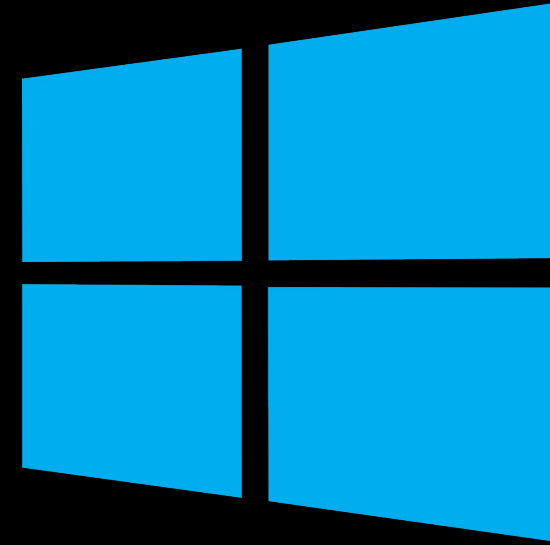
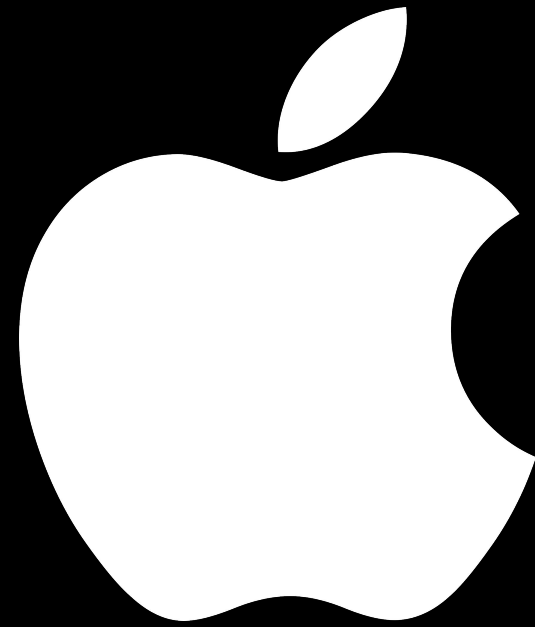
Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

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



# .ipynb



kaggle

# Atajos de teclado

- Insertar celda arriba - **A**
- Insertar celda a continuación - **B**
- Cambiar celda a Markdown - **M**
- Eliminar - **X** o **DD**
- Deshacer - **Z**



# Variables y Tipos de Datos en Python





¿Qué es una **Variable**?

# ¿Qué es una **Variable**?

Una **variable** en programación es como un contenedor en la memoria que almacena datos.

# ¿Qué es una **Variable**?

Una **variable** en programación es como un contenedor en la memoria que almacena datos.

$$x = 5$$



# Tipos de datos en Python

# Tipos de datos en Python

- **String** (Cadena)
- **Integer** (Número entero)
- **Float** (Número decimal)
- **Boolean** (True, False)



# Listas en Python



# Listas en Python

Una **lista** es una colección ordenada y modificable de elementos.

```
python
```

```
lista = [1, "a", True]
```

# Listas en Python

- Las **listas** pueden contener diferentes tipos de datos
- Los elementos en una **lista** están indexados y empiezan por 0

```
python
```

```
lista = [1, "a", True]
```



# Listas vs Tuplas vs Conjuntos

<b>Lista (List)</b>	<b>Conjunto (Set)</b>	<b>Tupla (Tuple)</b>
<b>Las listas son Mutables</b>	<b>El conjunto es Mutable</b>	<b>La tupla es Inmutable</b>
<b>Es una colección Ordenada de elementos</b>	<b>Es una colección Desordenada de elementos</b>	<b>Es una colección Ordenada de elementos</b>
<b>Los elementos en la lista pueden ser reemplazados o cambiados</b>	<b>Los elementos en el conjunto no pueden ser cambiados o reemplazados</b>	<b>Los elementos en la tupla no pueden ser cambiados o reemplazados</b>



# Diccionarios

# Diccionarios en Python

Los **diccionarios** se utilizan para almacenar valores de datos en pares clave:valor.

Un **diccionario** es una colección que es ordenada\*, modificable y no permite duplicados.

*\*A partir de la versión 3.7 de Python, los **diccionarios** son ordenados.*





# Errores en Python

# Errores en Python

**SyntaxError:** Errores de sintaxis en el código.

**NameError:** Uso de una variable o función no definida.

**TypeError:** Operación con un tipo de dato incorrecto.

# Errores en Python

**Try y Except:** Utiliza bloques try y except para manejar errores. Permite que el programa continúe incluso si ocurre un error.



# Bucles en Python

# Bucles en Python

Un **bucle** (loop) es una estructura de control que repite un bloque de código mientras se cumple una condición.

Permite ejecutar repetidamente operaciones similares de manera eficiente.

# Bucles en Python

**For:** Iterar sobre una secuencia (como una lista, una cadena de texto, un rango, etc.).

```
python
```

```
for elemento in secuencia:  
    # Código a repetir
```

```
python
```

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



# Bucles en Python

**While:** Repetir un bloque de código mientras una condición sea verdadera.

```
python
```

```
while condicion:  
    # Código a repetir
```

```
python
```

```
i = 0  
while i < 5:  
    print(i)  
    i += 1
```

# Bucles en Python

## Control dentro de Bucles

**break:** Termina el bucle antes.

**continue:** Salta al siguiente ciclo de iteración.

**else:** Se ejecuta si el bucle termina normalmente (sin break).



¿Qué es una **función** en **Python**?

# ¿Qué es una **función** en **Python**?

Ya hemos utilizado **funciones**:

```
type()  
print()  
str()
```

# ¿Qué es una **función** en **Python**?

Una **función** es un bloque de código reutilizable que se ejecuta solo cuando es llamado.

Se define con la palabra clave `def` y puede recibir datos (conocidos como parámetros) y devolver un resultado.

# ¿Qué es una **Función** en **Python**?

Las **funciones** permiten agrupar código que realiza una tarea específica y reutilizarlo sin duplicaciones.

Dividen el código en partes más pequeñas y manejables, facilitando el mantenimiento y la depuración y creando modularidad en el código.





# ¿Cómo crear una **función**?

# ¿Cómo crear una función?

```
python
```

```
def saludar(nombre):  
    return f"Hola, {nombre}!"
```



¿Qué es un paquete?

# ¿Qué es un paquete?

Los **paquetes** son un componente esencial en la programación para importar funciones escritas previamente.

Un **paquete** en Python es un directorio que contiene archivos de Python. Los paquetes contienen módulos que se pueden importar y utilizar en diferentes programas.

```
python
```

```
import random  
numero_aleatorio = random.randint(1, 10)
```

# ¿Qué es un paquete?

Algunos paquetes muy utilizados son:

Numpy

Pandas

Matplotlib

Seaborn

¿Qué es un paquete?

```
import seaborn as sns
```



# ¿Qué es NumPy?

**NumPy** es una librería de Python que soporta grandes matrices y matrices multidimensionales. Las matrices están optimizadas para operaciones matemáticas y científicas.

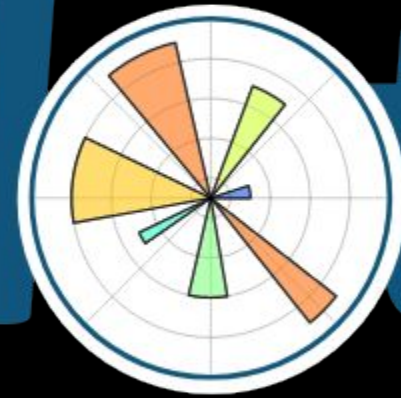
El nombre viene de **N**umerical **P**ython.







*matplotlib*



# ¿Qué es **Matplotlib**?

**Matplotlib** es una librería de Python que produce gráficos con calidad de publicación en una variedad de formatos impresos y entornos interactivos.



# ¿Qué es Matplotlib?

## matplotlib

Cheat sheet Version 3.5.0

### Quick start

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X, Y, color='green')

fig.savefig("figure.pdf")
plt.show()
```

### Anatomy of a figure

### Subplots layout

### Basic plots

- `plot([X], Y, [fmt], ...)`  
X, Y, fmt, color, marker, linestyle
- `scatter(X, Y, ...)`  
X, Y, [s]izes, [c]olors, marker, cmap
- `bar[h](x, height, ...)`  
x, height, width, bottom, align, color
- `imshow(Z, ...)`  
Z, cmap, interpolation, extent, origin
- `contour[f]([X], [Y], Z, ...)`  
X, Y, Z, levels, colors, extent, origin
- `pcolormesh([X], [Y], Z, ...)`  
X, Y, Z, vmin, vmax, cmap
- `quiver([X], [Y], U, V, ...)`  
X, Y, U, V, C, units, angles
- `pie(X, ...)`  
Z, explode, labels, colors, radius
- `text(x, y, text, ...)`  
x, y, text, va, ha, size, weight, transform
- `fill[_between](x)(...)`  
X, Y1, Y2, color, where

### Advanced plots

- `step(X, Y, [fmt], ...)`  
X, Y, fmt, color, marker, where

### Scales

`ax.set_[xy]scale(scale, ...)`

- `linear`: any values
- `log`: values > 0
- `symlog`: any values
- `logit`: 0 < values < 1

### Projections

`subplot(..., projection=p)`

- `p='polar'`
- `p='3d'`
- `p=ccrs.Orthographic()`  
import cartopy.crs as ccrs

### Lines

`linestyle or ls`

`capstyle or dash_capstyle`

`marker`

### Markers

`marker`

### Colors



# ¿Qué es **Pandas**?

La librería **Pandas** está construida sobre **NumPy** y proporciona un uso fácil para estructuras de datos y herramientas de análisis de datos con Python.



# ¿Qué es Pandas?

**Series:** Matriz unidimensional con etiquetas, similar a un diccionario.

a	3
b	-5
c	7
d	4



# ¿Qué es Pandas?

**DataFrame:** Estructura de datos tabular con columnas de diferentes tipos.



	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasília	207847528

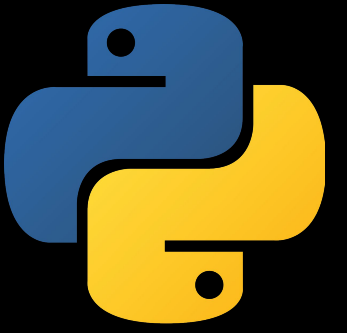
# ¿Qué es **Pandas**?

Permite cargar muchos tipos de archivos con

- `pd.read_csv()`
- `pd.read_excel()`
- `pd.read_json()`







seaborn

# ¿Qué es **Seaborn**?

La librería de visualización **Seaborn** se basa en **Matplotlib** y proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos.





# Proyecto **final**

 **Datademia**  
*Tu academia de **datos** en español*



# Dataset de **Titanic**

El dataset del **Titanic** contiene información sobre los pasajeros del famoso barco Titanic, que se hundió en 1912.

Incluye detalles como edad, género, clase del billete, si sobrevivieron o no, y más.



# Dataset de **Titanic**

- **survived** - Supervivencia: Indica si el pasajero sobrevivió o no. 0 = No, 1 = Sí.
- **pclass** - Clase del Billeto: 1 = Primera clase, 2 = Segunda clase, 3 = Tercera clase.
- **sex** - Género del pasajero.
- **age** - Edad del pasajero en años.
- **sibsp** - Hermanos/Cónyuges a Bordo
- **parch** - Padres/Hijos a Bordo
- **fare** - Tarifa pagada
- **embarked** - Puerto de Embarque - C = Cherbourg, Q = Queenstown, S = Southampton.

Otras cosas a tomar en cuenta:

- **pclass**: Sirve como un proxy para el estatus socioeconómico
- **sibsp y parch**: Estas variables definen las relaciones familiares de los pasajeros a bordo.