



# Python para el cálculo científico

---

Daniel Lubián Arenillas

12 de febrero de 2018



Presentando Python

Librerías importantes

Programación orientada a objetos

Sintaxis básica de Python

Numpy: el `ndarray`

Scipy

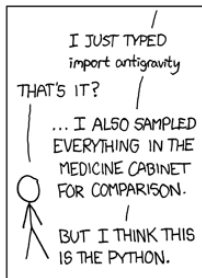
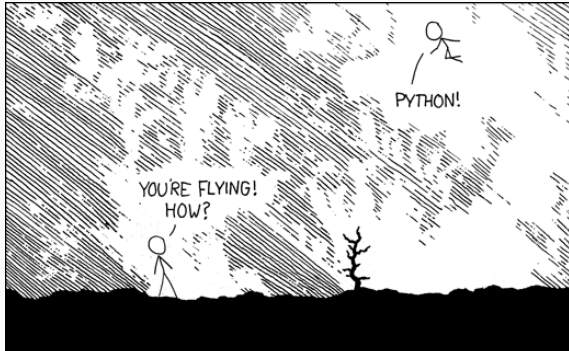
Matplotlib.pyplot

# Presentando Python

---



- Creado por Guido van Rossum en 1991.
- Lenguaje de propósito general:
  - Cálculo científico
  - Desarrollo web
  - Administración de sistemas
  - GUIs
  - Inteligencia artificial
  - Todo es posible
- Lenguaje multiparadigma, permite programación estructurada, orientada a objetos, funcional,...
- Lenguaje interpretado, no compilado.





- Dos versiones: 2.7 y **3.6**
- Libre, abierto y gratuito, con una comunidad enorme → todo a golpe de Google.
- Mil y una librerías abiertas y gratuitas.
- Rápido y fácil de escribir, puede ser lento de ejecutar.

Para escribirlo: Spyder, cuadernos Jupyter, Pycharm, VS Code, Atom, Geany, Notepad++...



IP[y]:  
IPython

**Numpy:** cálculo numérico

**Scipy:** cálculo científico

**Matplotlib:** graficado

**Pandas:** análisis de datos

**IPython:** consola interactiva



Un **objeto** tiene **métodos** (“funciones”) y **atributos** (“variables”) que lo constituyen.

- `A.shape`
- `z.conjugate()`
- `v.reshape((3,4))`

Todo Python trabaja con objetos<sup>1</sup>

---

<sup>1</sup>(hasta donde yo sé)



# Sintaxis básica de Python

---



Entero	integer	1
Con coma flotante	float	1.
Complejo	complex	1. + 2j
Booleano	boolean	True

## División de enteros

```
1 print(3 / 2)
2 print(3 / 2.)
3 print(3.0 // 2)
4
5 ## SALIDA
6 # 1.5
7 # 1.5
8 # 1
```



**Cadenas**  $\rightarrow s = \text{"perro"}$  inmutable

**Listas**  $\rightarrow l = [1, \text{'perro'}, \text{True}]$

**Tuplas**  $\rightarrow t = (1, \text{'perro'}, \text{True})$  inmutable

**Diccionarios**  $\rightarrow d = \{ \text{'O': 16, 'H2O': 18} \}$

**Ojo: los índices van de 0 a n-1, como en C**

- $s[0]$  devuelve 'p'
- $l[-1]$  devuelve True
- $t[3]$  no existe
- $d[\text{'H2O'}]$  devuelve 18
- $l[0:2]$  devuelve  $[1, \text{'perro'}]$  ( $0 \leq i < 2$ )



## Condición: if-elif-else

```
1 if 1 in x:  
2     haz esto  
3 elif 2 < 7 and 2 > 1:  
4     haz esto otro  
5 else:  
6     o esto
```



## Bucle: for

```
1 x = [2, "muse"]
2 n = 2
3
4 for i in range(0, n):
5     for x_i in x:
6         print(i, x_i)
7
8 ## SALIDA:
9 # 0 2
10 # 0 muse
11 # 1 2
12 # 1 muse
```



## Bucle: while

```
1 z = 1 + 1j
2
3 while abs(z) < 100:
4     if z.imag == 0:
5         break
6     z = z**2 + 1
```



## Iteración avanzada

```
1 words = ('cool', 'powerful', 'readable')
2
3 for index, item in enumerate(words):
4     print((index, item))
5
6 # (0, 'cool')
7 # (1, 'powerful')
8 # (2, 'readable')
```



## *List comprehensions*

```
1 x = [i**2 for i in range(6)]  
2 print(x)  
3  
4 ## SALIDA  
5 # [0, 1, 4, 9, 16, 25]
```





## Definición

```
1 # funciones normales
2 def mi_funcion(x, z=2):
3     return x**2 + z
4
5 # funciones anonimas
6 f = lambda x: x**3 + 5
```

## Paso por referencia

Los argumentos de las funciones se pasan por referencia, no por parámetro. Se mete la variable, no una copia, por lo que si se hace alguna modificación a `x` en el ámbito de la función, `x` cambiará fuera de la función.



## Importar

```
1 import function
2 import numpy as np
3 from scipy.linalg import inv as hazme_inversa
4 import matplotlib.pyplot as plt
5 from math import *
6
7 e = function.mi_funcion(5, z=3)
8 z = np.linspace(0, 1, 3)
9 i = hazme_inversa(np.eye(3))
10 plt.plot(z)
11 e = sqrt(e)
```

## **Numpy: el ndarray**

---

# Scipy

---

# Matplotlib.pyplot

---