python™

Python

- Creado por Guido van Rossum en **1991**
- Lenguaje **interpretado,** interactivo, orientado a objetos, **multiplataforma**
- **Modo interactivo**: IDLE (Shell de Python)
- El Zen de Python (en modo interactivo):

```
>>> import this
```

Python 2 vs Python 3

- **Python 2** es más usado, **Python 3** es el futuro
- Muchas más librerías para Python 2 pero morirá en 2020
- Diferencias principales:

Comportamiento	**Python 2**	**Python 3**
Imprimir por pantalla	*print* "Hola"	*print* ("Hola")
Entrada de teclado	*raw_input* ()	*input* ()
División entera	**/**	**//**
Tipos enteros	*int long*	*int*

Comentarios

```
# Comentario de una línea

'''
Comentario
de varias
líneas
'''
```

Tipos de datos

- **Números**
  - *int*, *float*, *complex*
- **String** (*str*)
- **Lista** (*list*)
  - Puedo contener tipos de dato distintos
  - Longitud variable (**mutable**)
- **Tupla** (*tuple*)
  - Lista **inmutable**
- **Diccionario** (*dict*)
  - Lista con clave / valor
  - En otros lenguajes mapa o tabla hash

Entrada / salida por consola

```
print('Hola') # Imprime: Hola

nombre = 'Monty'
print('Hola', nombre) # Hola Monty

numero = 5
print(numero) # Imprime: 5

# Obtiene un texto del teclado
texto = input()
nombre = input('Dime tu nombre:')
# Evalúa la entrada por consola
numero = eval(input('Dame un número:'))
suma = 2 + numero
```

Operadores

```
+ - * / %              // Aritméticos
**                     // Exponente
//                     // División entera
= += -= *= /=          // De asignación
== != < > <= >=        // Relacionales
not and or             // Lógicos
```

Strings

```
texto = 'hola'          # "hola" también vale
texto[0]                # "h"
texto[1:3]              # "ol"
texto[2:]               # "la"
texto.capitalize()      # "Hola"

texto2 = 'adiós "entre comillas"'

# 'hola y adiós "entre comillas"'
texto = texto + ' y ' + texto2

texto.replace('adiós', 'bye')
lista_strings = texto.split(' ')
```

Listas y tuplas

```
lista = [1, 'dos', 2.5]          # Declaración
lista.append(3)                  # Añadir
lista.insert(1, 1.5)             # Insertar
lista.remove(3)                  # Eliminar
lista[2]                         # Acceder
lista[2] = 54                    # Modificar
lista[0:2]                       # Sub lista
len(lista)                       # Tamaño
min(lista)                       # Mínimo
max(lista)                       # Máximo
lista.sort()                     # Ordenar
lista.reverse()                  # Invertir
lista = lista + lista2           # Concatenación

# Tuplas: Mismas operaciones que las listas
# Menos operaciones de modificación y declaración
tupla = (3, 2.0, 'uno')          # Declaración
```

Diccionarios (mapa, tabla hash…)

```
numerosRomanos = {1: 'I', 2: 'II', 3: 'III', 4:
'IV', 5: 'V'}                    # Declaración
numerosRomanos[6] = 'VI'         # Añadir
del numerosRomanos[2]            # Eliminar
numerosRomanos[1]                # Acceder
numerosRomanos.get(1)            # Acceder
numerosRomanos[4] = 'IIII'       # Modificar
numerosRomanos.keys()            # Claves
numerosRomanos.values()          # Valores
numerosRomanos.clear()           # Borrar todo
```

Conjuntos (sin repetidos)

```
conjunto = {1, 2, 3, 4}          # Declaración
conjunto.add(6)                  # Añadir un 6
conjunto.remove(2)               # Eliminar el 2
4 in conjunto                    # ¿Está el 4?
```

Conversión de tipos (casting)

```
int(2.4)          # Convierte a entero: 2
float(2)          # Convierte a float: 2.0

# Evalua el string (en este caso se
obtiene un float: 6.3)
eval('6.3')

list((2, 3, 4))       # De tupla a lista

# De lista a tupla
tuple(['a', 'e', 'i', 'o', 'u'])
```

If

```
if 2 < 3:
    print('correcto')
```

Else if

```
if 5 == 4:
    print('que va')
elif 3 <= 25:
    print('sí')
else:
    print('no')
```

Else

```
if 5 > 8:
    print('no creo')
else:
    print('claro')
```

No hay switch / case

For

```
for i in range(1,8):
    print(i)

vocales = ['a', 'e', 'i', 'o', 'u']
for letra in vocales:
    print(letra)
```

While

```
i = 0
while i < 10:
    print(i)
    i += 1
```

Continue y break

```
j = 1
while j < 20:
    if j == 10:
        continue  # salta a la condición del while
    if j == 15:
        break     # para el bucle
    j += 2
```

Funciones

```
def suma(op1, op2):
    return op1 + op2

resultado = suma(4, 5) # resultado vale 9

def diHola():
    print('Hola')

diHola()                    # Imprime: Hola

def despedida(nombre='humano'):
    print('Adiós', nombre)

despedida()                 # Imprime: Adiós humano
despedida('Flint')          # Imprime: Adiós Flint
```

Algunas librerías

```
import math
math.pi                 # El número pi
math.cos(math.pi)       # Coseno de pi
math.log(math.e)        # Logaritmo del número e

import random
random.random()         # Aleatorio entre 0 y 1
random.randint(1,10)    # Aleatorio entre 1 y 10

import time
time.time()             # Segundos desde Epoch
localtime = time.localtime(time.time())
print(localtime.tm_hour)    # Hora actual
print(localtime.tm_min)     # Minuto actual
time.sleep(3)               # Parar 3 segundos
```

Más cosas

```
def esPar(x): # Devuelve si un número es par
    return x % 2 == 0

nums = list(range(5))    # Crea la lista: [0,1,2,3,4]
sum(nums)                # Suma todo: 10

map(lambda x: x**2, nums) # Eleva todo al cuadrado
filter(esPar, nums)       # Se queda con los pares

# En Python 3 map y filter devuelven un iterador
# Por ejemplo, para obtener una lista:
lista = list(filter(esPar, nums))

# Eleva al cuadrado sólo los pares
[x**2 for x in range(5) if x % 2 == 0]
```

Ficheros

```
# Abrir / crear fichero para escritura
fichero = open('spam.txt', 'w') # "a" (añadir)
fichero.name # Nombre del fichero
fichero.mode # Modo: "w" (sobreescribir)
fichero.closed # Cerrado? False
fichero.write('spam spam spam') # Escribir
fichero.close() # Cerrar

# Abrir / crear fichero para lectura
fichero = open('spam.txt', 'r')
contenido = fichero.read() # Leer

import os
os.rename('spam.txt', 'otro.txt') # Renombrar
os.remove('otro.txt') # Borrar
```

Raspberry Pi - Pines de salida

```
import RPi.GPIO as GPIO

# Elegir modo de numeración de pines (BCM / BOARD)
GPIO.setmode(GPIO.BCM)          # Numeración lógicos

# Pin de salida
GPIO.setup(17, GPIO.OUT)        # Pin 17 como salida
GPIO.output(17, GPIO.HIGH)      # Encender pin

# Salida analógica (PWM)
pwm = GPIO.PWM(18, 100)         # Pin 18 a 100Hz
pwm.start(0)                    # Valor inicial: 0%
pwm.ChangeDutyCycle(50)         # 50% (2.5V)
pwm.stop()                      # Apagado

# Reseteo de pines
GPIO.cleanup()
```

Raspberry Pi - Pines de entrada

```
# Pin 23 como entrada
GPIO.setup(23, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

while True: # Bucle infinito
    if GPIO.input(23) == 1: # Comprobar pin 23
        print('Botón pulsado')

# Interrupciones
def evento(channel): # Llamada a la función al pulsar
    print('Botón pulsado')

# Asignar interrupción al pin 23
GPIO.add_event_detect(23, GPIO.RISING,
callback=evento, bouncetime=300)

GPIO.remove_event_detect(23) # Quitar interrupción
```

Servidor

```python
import socket

socket_servidor = socket.socket() # Crea un socket
host = socket.gethostname() # Obtiene el hostname
port = 12345 # Seleccionar un puerto
# Asociar el socket al host y puerto
socket_servidor.bind((host, port))
# Espera a peticiones de cliente (máximo 5)
socket_servidor.listen(5)

while True:
    # Establece conexión con cliente
    conexion, direccion_ip = socket_servidor.accept()
    print('Petición de:', direccion_ip)
    mensaje = 'Soy el servidor!'
    conexion.send(mensaje.encode('ascii'))
    conexion.close() # Cierra la conexión
```

Cliente

```python
import socket

# Crea un socket (TCP, otra opción: UDP)
socket_cliente = socket.socket()
# Obtiene el hostname
host = socket.gethostname()
port = 12345 # Seleccionar un puerto

# Inicia conexión
socket_cliente.connect((host, port))
# Recibe 1024 bytes
mensaje = socket_cliente.recv(1024)
print(mensaje.decode('ascii'))

socket_cliente.close() # Cierra el socket
```

Herramientas

- Documentación oficial:
  - Web de Python
- IDEs:
  - **IDLE** (consola oficia de Python)
  - Repl.it (online)
  - **PyCharm** (de JetBrains)
  - **PyDev** (plugin de Eclipse)
  - **Thonny** (para Raspberry)