

# Introducción a Python

Héctor Chamorro Álvarez

Department of Computer Science and Automation, University of Salamanca

Plaza de la Merced, s/n, 37008, Salamanca, Spain

Xamo1998@usal.es

## 1. Introducción

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional.

La definición anterior nos da una idea de que es Python, pero hay varias maneras de definirlo, personalmente mi favorita es la que aparece en el libro *Learning Python* de O'reilly :

*Python is probably better known as a general-purpose  
programming language that blends procedural, functional,  
and object-oriented paradigms*

## 2. Como ejecutar programas en Python

En python tenemos dos posibilidades de ejecutar programas dependiendo del uso que le vayamos a dar:

1. Prompt interactiva
2. Ficheros de código

En la primera opción, Python nos permite iniciar una sesión interactiva para ejecutar comandos simples sin necesidad de usar archivos. Para esto, tan solo debes abrir una terminal si te encuentras en un dispositivo Unix o un Command prompt si nos encontramos en Windows y escribir la palabra Python o py.

Una vez realizado esto (si hemos instalado Python correctamente) se iniciará una sesión interactiva (Figura 1) en la que podemos usar algunos comandos básicos para ver su funcionamiento.

```

C:\Users\Hector>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 2+3
5
>>> print('Hello World!')
Hello World!
>>> print(2 ** 100)
1267650600228229401496703205376
>>> x = 'Spam'
>>> print(x * 8)
SpamSpamSpamSpamSpamSpamSpamSpam
>>> import os
>>> os.getcwd()
'C:\\Users\\Hector'
>>>

```

**Figura 1** Sesión interactiva en Python

Como vemos en el ejemplo, es muy sencillo de usar, también se aprecia en la figura como se importan los módulos y se hace referencia a ellos para, en este caso, mostrar la ruta donde nos encontramos.

En el segundo caso lo único que necesitamos es un editor de texto donde codificar el programa, este debe acabar en **.py**. Veamos un breve ejemplo:

```

1. #A first Python script
2. import sys #Load a library module.
3. print(sys.platform)
4. print(2**100)
5. x= 'Spam'
6. print(x*8)

```

Al ejecutar el programa obtenemos la siguiente salida:

```

E:\Documents>script1.py
win32
1267650600228229401496703205376
SpamSpamSpamSpamSpamSpamSpamSpam
E:\Documents>

```

**Figura 2** Resultado de Script1.py

Para ejecutar el script que hemos realizado tan solo es necesario poner el nombre del fichero, también es posible usar el comando Python antes del nombre del fichero.

### 3. Tipos de datos

En Python, los datos toman la forma de objetos ya sean objetos que Python nos proporciona o los que creemos nosotros. Como veremos, en Python **todo** es un objeto.

#### 3.1 Tipos de datos principales

En la siguiente tabla veremos los tipos de datos principales que podemos encontrar en Python.

| <i>Tipo de objeto</i>                | <i>Ejemplo</i>                                   |
|--------------------------------------|--|
| Números                              | 1234, 3.1415, 3+4j, 0b111, Decimal(), Fraction() |
| Strings                              | 'spam', "Bob's", b'a\x01c', u'sp\xc4m'           |
| Listas                               | [1, [2, 'three'], 4.5], list(range(10))          |
| Diccionarios                         | {'food': 'spam', 'taste': 'yum'}, dict(hours=10) |
| Tuplas                               | (1, 'spam', 4, 'U'), tuple('spam'), namedtuple   |
| Ficheros                             | open('eggs.txt'), open(r'C:\ham.bin', 'wb')      |
| Conjuntos                            | set('abc'), {'a', 'b', 'c'}                      |
| Otros tipos                          | Booleans   |
| Unidades de programa                 | Funciones, módulos, clases                       |
| Tipos de implementación relacionados | Código compilado                                 |

Centraremos nuestra atención solo aquellos que presenten algún tipo de dificultad.

##### 3.1.1 Listas

En Python, las listas de objetos son la secuencia más general que nos vamos a encontrar proporcionada por el lenguaje. Las listas son colecciones de objetos arbitrarios ordenados que no tienen un tamaño fijo y son mutables. Veamos un breve ejemplo:

```

C:\Users\Hector>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> Array=[123, 'Spam', 1.23]
>>> len(Array)
3
>>> Array[0]
123
>>> Array[-1]
1.23
>>> Array + [4, 5, 6]
[123, 'Spam', 1.23, 4, 5, 6]
>>> Array * 2
[123, 'Spam', 1.23, 123, 'Spam', 1.23]
>>> Array
[123, 'Spam', 1.23]
>>>

```

**Figura 3** Ejemplo uso de listas

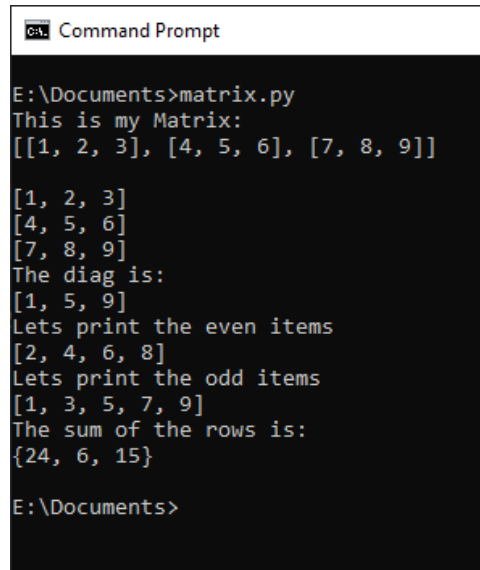
Como podemos ver, trabajar con listas en Python es extremadamente sencillo e intuitivo. Otro uso para las listas es la creación de matrices que no son más que una lista de listas. Veamos un ejemplo más complejo que involucre resolver problemas simples con matrices:

```

1. M= [[1,2,3],
2.     [4,5,6],
3.     [7,8,9]]
4. print("This is my Matrix:")
5. print(M)
6. print("")
7. for i in (0,1,2):
8.     print(M[i])
9. print("The diag is:")
10. diag=[M[i][i] for i in (0,1,2)]
11. print(diag)
12. print("Lets print the even items")
13. evens=[row[i] for row in M for i in (0,1,2) if row[i]%2==0]
14. print(evens)
15. print("Lets print the odd items")
16. odds=[row[i] for row in M for i in (0,1,2) if row[i]%2!=0]
17. print(odds)
18. print("The sum of the rows is:")
19. print({sum(row) for row in M}) #This is a set

```

Al ejecutar el programa obtenemos la siguiente salida:



```

C:\> Command Prompt

E:\Documents>matrix.py
This is my Matrix:
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]

[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
The diag is:
[1, 5, 9]
Lets print the even items
[2, 4, 6, 8]
Lets print the odd items
[1, 3, 5, 7, 9]
The sum of the rows is:
{24, 6, 15}

E:\Documents>

```

**Figura 4** Resultado de Matrix.py

Este es un buen ejemplo para familiarizarse con la nomenclatura de Python y en el que podemos apreciar la gran potencia de este lenguaje ya que somos capaces de imprimir matrices, calcular la diagonal, la suma de los numeros pares o impares en tan solo una linea de código. Analicemos como calcular los numeros pares:

```
evens=[row[i] for row in M for i in (0,1,2) if row[i]%2==0]
```

Vemos que en la variable *evens* vamos a guardar una lista ya que vemos que a la derecha del igual se encuentran unos corchetes. Nos dice que coja *row[i]* para cada fila en *M* lo que nos devolveria los elementos de cada fila, pero antes debemos definir el rango de *i*, lo cual se hace en *for i in (0,1,2)* ya que nuestra matriz tiene 3 elementos por fila y columna. Por último nos queda especificar la condición de selección la cual nos dice que si el elemento es divisible por 2 lo añada.

### 3.1.2 Diccionarios

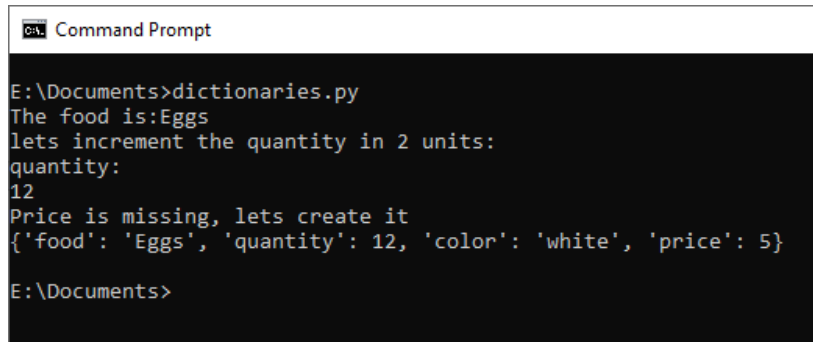
Otro tipo de dato importante en Python son los diccionarios, estos no son secuencias, normalmente son conocidos como mappings. Los mappings son colecciones de objetos, pero estos se guardan por clave en vez de en una posición relativa. Veamos un ejemplo para comprender mejor su funcionamiento:

```

1. Dic={'food': 'Eggs', 'quantity': 10, 'color': 'white'}
2. print("The food is:"+ Dic['food'])
3. print("lets increment the quantity in 2 units:")
4. Dic['quantity']+=2
5. print("quantity: ")
6. print(Dic['quantity'])
7. if not 'price' in Dic:
8.     print("Price is missing, lets create it")
9.     Dic['price']=5
10. print(Dic)

```

Al ejecutar el programa obtenemos la siguiente salida:



```

C:\ Command Prompt

E:\Documents>dictionaries.py
The food is:Eggs
lets increment the quantity in 2 units:
quantity:
12
Price is missing, lets create it
{'food': 'Eggs', 'quantity': 12, 'color': 'white', 'price': 5}

E:\Documents>

```

**Figura 5** Resultado de Dictionaries.py

Como vemos en el ejemplo, los objetos se almacenan en forma clave/valor y para buscar el valor de un objeto o dato en un diccionario solo debemos hacer referencia a la clave que tenga asignada.

### 3.1.3 Ficheros

En Python el manejo de ficheros es extremadamente fácil, tan solo son necesarias 3 líneas para abrir un fichero, leerlo y cerrarlo:

```

1. fp=fopen('data.csv')
2. data=fp.read()
3. fp.close()

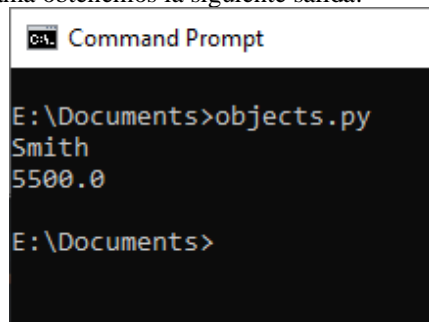
```

### 3.1.4 Clases

Por último, veremos cómo crear nuestra propia clase. Crearemos una clase Trabajador con un nombre y un salario como atributos y 2 métodos: conseguir apellido y realizar aumento. Finalmente, también tenemos el constructor.

```
1. class Worker:
2.     def __init__(self, name, pay):
3.         self.name=name
4.         self.pay=pay
5.     def lastName(self):
6.         return self.name.split()[-1]
7.     def giveRaise(self, percent):
8.         self.pay*=(1.0+percent)
9.
10. bob= Worker('Bob Smith', 4000)
11. sue= Worker('Sue Jones', 5000)
12. print(bob.lastName())
13. sue.giveRaise(.10)
14. print(sue.pay)
```

Al ejecutar el programa obtenemos la siguiente salida:



```
Command Prompt

E:\Documents>objects.py
Smith
5500.0

E:\Documents>
```

**Figura 6** Resultado de Objects.py

## 4. Python en la administración de sistemas

En la administración de sistemas es muy común encontrarse varios lenguajes como Perl, Bash, ksh, Ruby...

Pero la pregunta que debemos hacernos es: ¿Es Python mejor que Perl, Bash, Ruby o cualquier lenguaje? Es muy complicado poder afirmar que un lenguaje es mejor que otro. Sin embargo, Python tiene varios puntos positivos que hacen que sea una opción muy recomendada:

- Es fácil de aprender
- En Python podrás escribir scripts en horas mientras que en otros lenguajes tardarías días o semanas literalmente.
- Puede realizar tareas complejas (Leer un fichero de Log línea a línea y extraer información deseada para analizarla)
- Legibilidad

Veamos varios ejemplos de tareas que podríamos realizar tanto con Python, Perl o Bash para notar la diferencia en cuanto a legibilidad:

```
1. #!/bin/bash
2. if [ -d "/tmp" ] ; then
3.     echo "/tmp is a directory"
4. else
5.     echo "/tmp is not a directory"
6. fi
```

Así sería el script equivalente en Perl:

```
1. #!/usr/bin/perl
2. if (-d "/tmp") {
3.     print "/tmp is a directory\n";
4. }
5. else {
6.     print "/tmp is not a directory\n";
7. }
```

Y así sería el equivalente en Python:

```
1. #!/usr/bin/env python
2. import os
3. if os.path.isdir("/tmp"):
4.     print "/tmp is a directory"
5. else:
6.     print "/tmp is not a directory"
```



#### 4.1 Ejemplo de conexión SSH automatizada

Veamos un ejemplo de cómo ejecutar un comando o varios comandos en una máquina remota a través de SSH. Vamos a utilizar el módulo *paramiko*. Para poder utilizar este módulo debemos instalarlo a través de la herramienta *pip*. Para instalar *pip* debemos seguir los siguientes pasos:

1. Instalar *curl* (Si este no está instalado) con la siguiente instrucción `apt install curl`
2. Instalamos *pip*: `curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py`
3. Ejecutamos lo siguiente: `python get-pip.py`
4. Por último, escribimos lo siguiente para instalar el módulo: `pip install paramiko`

Primero vamos a ver el código y posteriormente procederemos a explicarlo:

```
1. #!/usr/bin/env python
2. import paramiko
3. import socket
4. import getpass
5. host=raw_input('Enter the host:')
6. hostname= socket.getaddrinfo(host, None)[0][4][0]
7. username = raw_input('Enter the user:')
8. password = getpass.getpass('Password:')
9.
10. paramiko.util.log_to_file('paramiko.log')
11. s= paramiko.SSHClient()
12. s.load_system_host_keys()
13. s.set_missing_host_key_policy(paramiko.AutoAddPolicy())
14. s.connect(hostname, username=username, password=password)
15. stdin, stdout, stderr = s.exec_command('ls -l')
16. print stdout.read()
17. s.close()
```

Para conectar al host deseado debemos conocer la ip ya que *paramiko* no resuelve los nombres de host. Para obtener la ip de un host dado usamos el módulo *socket*, `socket.getaddrinfo('encia.usal.es', None)` esto nos devuelve la siguiente información:

```
C:\Users\xamo>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import socket
>>> info=socket.getaddrinfo('encia.usal.es',None)
>>> info
[(<AddressFamily.AF_INET: 2>, 0, 0, '', ('212.128.144.28', 0))]
>>> info[0][4]
('212.128.144.28', 0)
>>> info[0][4][0]
'212.128.144.28'
>>>
```

**Figura 7** Valor de `socket.getaddrinfo()`

Como vemos en la figura, para acceder al valor de la ip tenemos que usar [0][4][0].

Una vez tenemos el *hostname* rellenamos los campos *username* y *password* con los valores de nuestro id y contraseña.

Para realizar la conexión usamos la función *connect* pasándole como argumentos el nombre del host, del usuario y de la contraseña.

Una vez conectados ejecutaremos el comando *ls -l* para listar el contenido de la carpeta de inicio.

Podemos realizar cualquier comando y así automatizar tareas que hagamos varias veces.

## 4.2 Ejemplo de descarga de ficheros por sftp

Este ejemplo es muy parecido al anterior ya que usaremos el mismo módulo *paramiko* y descargaremos todos los elementos de una carpeta en un servidor. Veamos el código y después procederemos a la explicación:

```
1. #!/usr/bin/env python
2. import paramiko
3. import os
4. import socket
5. import getpass
6. host=raw_input('Enter the host:')
7. hostname = socket.getaddrinfo(host, None)[0][4][0]
8. port = 22
9. username = raw_input('Enter the user: ')
10. password = getpass.getpass('Password:')
11. dir_path = '/home/'+username+'/folder_to_retrieve'
12.
13. t = paramiko.Transport((hostname, port))
14. t.connect(username=username, password=password)
15. sftp = paramiko.SFTPClient.from_transport(t)
16. files = sftp.listdir(dir_path)
17. for f in files:
18.     print 'Retrieving', f
19.     sftp.get(os.path.join(dir_path, f), f)
20. t.close()
```

Utilizamos la misma táctica para obtener la ip del servidor, después creamos un objeto transporte con el que haremos todas las operaciones necesarias.

### 4.3 Ejemplo avanzado de ftp con ssh

Imaginemos por ejemplo que queremos descargar una carpeta en concreto de un servidor, en el siguiente ejemplo, nos conectaremos por ssh al servidor, mostraremos todas las carpetas de la carpeta `/home/user` y le preguntaremos al usuario por la carpeta que quiere descargar. Después procederá a descargar todos los archivos de dicha carpeta.

```
1. #!/usr/bin/env python
2. import paramiko
3. import os
4. import socket
5. import getpass
6. host=raw_input('Enter the host:')
7. hostname = socket.getaddrinfo(host, None)[0][4][0]
8. port = 22
9. username = raw_input('Enter the user:')
10. password = getpass.getpass('Password:')
11.
12. paramiko.util.log_to_file('paramiko.log')
13. s= paramiko.SSHClient()
14. s.load_system_host_keys()
15. s.set_missing_host_key_policy(paramiko.AutoAddPolicy())
16. s.connect(hostname, username=username, password=password)
17. stdin, stdout, stderr = s.exec_command('ls -d */')
18. print(stdout.read())
19. folder=raw_input('Type the folder you want to retrieve...')
20. dir_path = '/home/'+username+'/'+folder
21.
22. t = paramiko.Transport((hostname, port))
23. t.connect(username=username, password=password)
24. sftp = paramiko.SFTPClient.from_transport(t)
25. files = sftp.listdir(dir_path)
26. for f in files:
27.     print 'Retrieving', f
28.     sftp.get(os.path.join(dir_path, f), f)
29. t.close()
30. s.close()
```

#### 4.4 Ejemplos del uso de OS

*Os* es un módulo que nos permite hacer una gran cantidad de cosas en la administración de sistemas ya que combinamos el poder de Python con todas las ordenes que podemos hacer desde un terminal.

Si queremos crear una carpeta y copiar el fichero */etc/shadow* en la carpeta actual podremos hacerlo todo desde Python:

```
1. import os
2. folder=raw_input('Introduce el nombre de la carpeta:')
3. os.system('mkdir '+folder)
4. os.system('cp /etc/shadow '+os.getcwd()+ '/' +folder+ '/shadow')
```

## 5. Referencias

1. (Lutz, 2013) Learning Python. 5<sup>th</sup> Edition, O'Reilly
2. (Noah Gift, Jeremy M. Jones, 2008) Python for Unix and Linux System Administration, O'Reilly