

# Taller de Introducción a Python

Daniel Gutiérrez Reina, [dgutierrezreina@us.es](mailto:dgutierrezreina@us.es)



## ACE-TI



# Objetivos:

## OBJETIVOS DEL TALLER:

- ▶ Aprender nociones básicas sobre programación en Python. Programar ejemplos sencillos en Python.
- ▶ Familiarizarse con Spider como IDE de programación.



# Introducción a Python



## CONSIDERACIONES PREVIAS:

- ▶ Creado en los años 90s por el holandés **Guido van Rossum**. El nombre se debe al grupo de humoristas *Monty Python*.
- ▶ Python es un lenguaje de **programación interpretado** (Lenguajes interpretados Vs Lenguajes compilados).
  - ▶ C es un lenguaje compilado → Errores en tiempo de compilación.
  - ▶ Python → Los errores saltan en tiempo de ejecución.
- ▶ Lenguaje multiplataforma (Windows, Linux, Mac) → La distribuciones de Linux suelen venir con el interprete de Python ya incorporado.
- ▶ **Open source** (gratis).
- ▶ Está ganando mucha importancia en los último años (diseños web y análisis de datos). **Machine Learning, Big Data, Deep Learning, Artificial Intellingence.**
- ▶ Nosotros vamos a trabajar con la **versión 2.7 de Python** (Aunque hay versiones superiores, ésta es aun la más utilizada). Todas las versiones de Python 2.x son compatibles, hubo un salto con Python 3.x en el que ciertos métodos no son compatibles con Python 2.x.





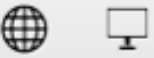





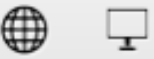
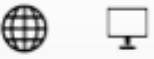
# Introducción a Python

Popularidad de los lenguajes de programación (Publicado en IEEE Spectrum 2015):



# Introducción a Python

Popularidad de los lenguajes de programación (Publicado en IEEE Spectrum 2016):

| Language Rank | Types   | Spectrum Ranking |
|---------------|---|------------------|
| 1. C          |   | 100.0            |
| 2. Java       |    | 98.1             |
| 3. Python     |    | 98.0             |
| 4. C++        |   | 95.9             |
| 5. R          |    | 87.9             |
| 6. C#         |    | 86.7             |
| 7. PHP        |  | 82.8             |
| 8. JavaScript |  | 82.2             |
| 9. Ruby       |  | 74.5             |
| 10. Go        |  | 71.9             |

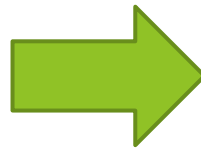


# Introducción a Python

Índice PYPL: [Popularity of Programming Language](#) (Búsquedas en Google)

PYPL Index (Worldwide)

| Mar 2016 ▲ | Change ◆ | Programming language ◆ | Share ◆ | 12 month trends ◆ |
|------------|----------|------------------------|---------|-------------------|
| 1          | -        | Java                   | 24.2%   | +0.3%             |
| 2          | ↑        | Python                 | 11.9%   | +1.2%             |
| 3          | ↓        | PHP                    | 10.7%   | -0.8%             |
| 4          | -        | C#                     | 8.9%    | +0.1%             |
| 5          | -        | C++                    | 7.6%    | -0.5%             |
| 6          | -        | C                      | 7.5%    | +0.1%             |
| 7          | -        | Javascript             | 7.3%    | +0.3%             |
| 8          | -        | Objective-C            | 5.0%    | -0.9%             |
| 9          | ↑↑       | Swift                  | 3.0%    | +0.4%             |
| 10         | -        | R                      | 2.9%    | +0.3%             |
| 11         | ↓↓       | Matlab                 | 2.8%    | -0.3%             |
| 12         | -        | Ruby                   | 2.3%    | -0.2%             |
| 13         | -        | Visual Basic           | 1.8%    | -0.4%             |
| 14         | -        | VBA                    | 1.5%    | +0.1%             |
| 15         | -        | Perl                   | 1.1%    | -0.1%             |
| 16         | -        | Scala                  | 0.9%    | +0.2%             |
| 17         | -        | Iua                    | 0.5%    | +0.0%             |



Worldwide, Mar 2017 compared to a year ago:

| Rank | Change | Language    | Share  | Trend  |
|------|--------|-------------|--------|--------|
| 1    |        | Java        | 22.7 % | -1.4 % |
| 2    |        | Python      | 15.0 % | +3.0 % |
| 3    |        | PHP         | 9.3 %  | -1.2 % |
| 4    |        | C#          | 8.3 %  | -0.4 % |
| 5    | ↑↑     | Javascript  | 7.7 %  | +0.4 % |
| 6    | ↓      | C++         | 6.9 %  | -0.5 % |
| 7    | ↓      | C           | 6.9 %  | -0.1 % |
| 8    |        | Objective-C | 4.1 %  | -0.6 % |
| 9    |        | R           | 3.5 %  | +0.4 % |
| 10   |        | Swift       | 2.9 %  | +0.0 % |



# Introducción a Python

## Consultas en stackoverflow.com

javascript × 1327647

JavaScript (not to be confused with Java) is a dynamic, weakly-typed language used for client-side as well as server-side scripting. Use

624 asked today, 6787 this week

java × 1214811

Java (not to be confused with JavaScript or JScript) is a general-purpose object-oriented programming language designed to be used in

506 asked today, 5169 this week

c# × 1062418


for questions about Microsoft's C# .NET language or if your code uses C#.

350 asked today, 3981 this week

php × 1040639

a general-purpose programming language primarily designed for server-side web development.

473 asked today, 4439 this week

 android × 953931

Android, used for programming or developing digital devices (Smartphones, Tablets, Autos, TVs, Wear, Glass), is Google's mobile OS.

455 asked today, 4137 this week

jquery × 817719

a popular cross-browser JavaScript library that facilitates DOM (Document Object Model) traversal, event handling, animations, and

270 asked today, 2695 this week

python × 706301

a dynamic and strongly typed programming language designed to emphasize usability. Two similar but mostly incompatible versions of

538 asked today, 4837 this week

html × 626509

the standard markup language used for structuring web pages and formatting content. HTML describes the structure of a website

295 asked today, 2984 this week

c++ × 499440

a general-purpose programming language. It was originally designed as an extension to C, and keeps a similar syntax, but is now a

217 asked today, 1797 this week

ios × 492804

the mobile operating system running on the Apple iPhone, iPod touch, and iPad. Use the tag [ios] for questions related to programming

170 asked today, 1833 this week

css × 451324

a style sheet language used for describing the look and formatting of HTML (Hyper Text Markup Language), XML (Extensible Markup

201 asked today, 1988 this week

mysql × 447364

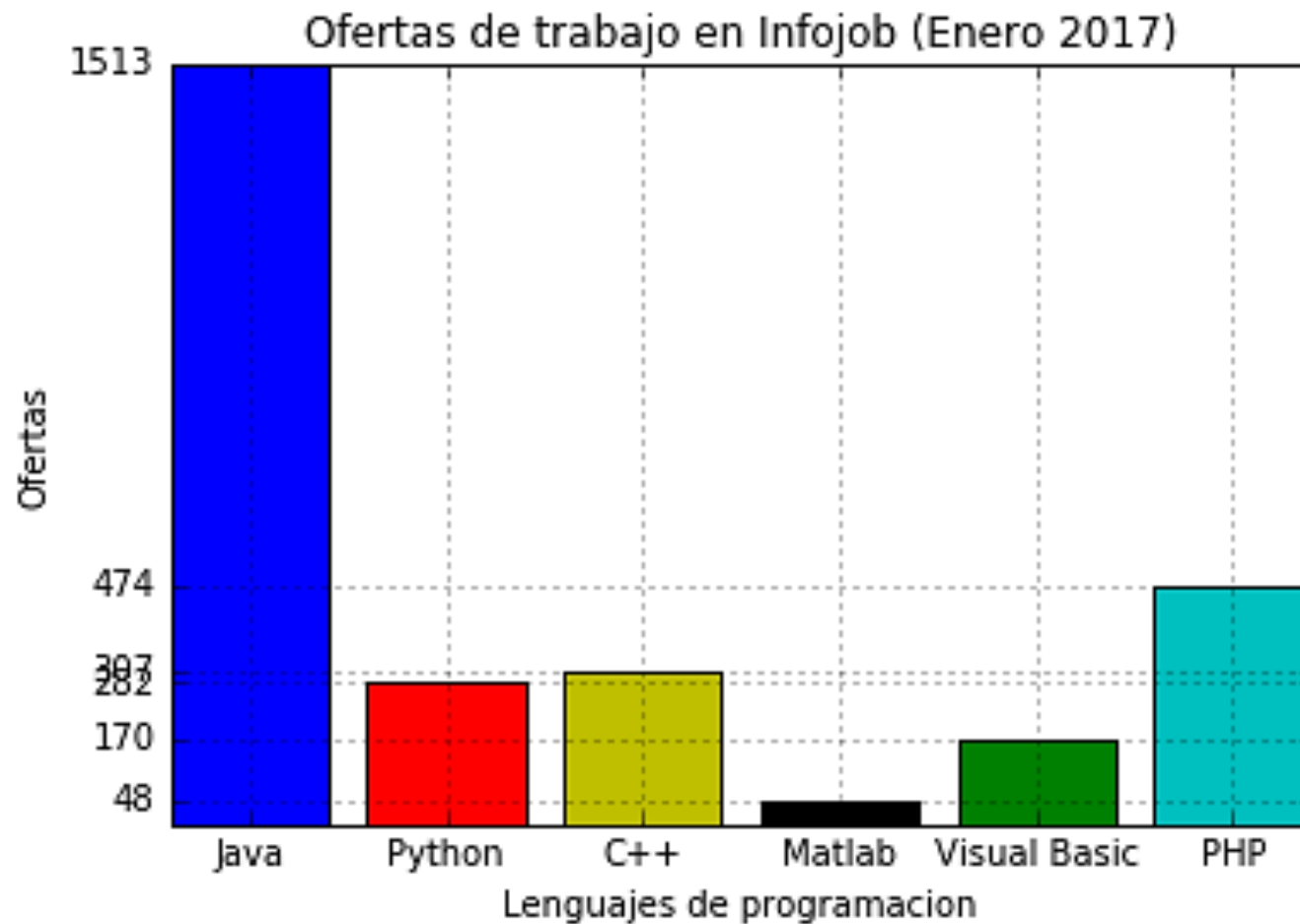
a freely available, open source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). Do not

203 asked today, 1902 this week



# Introducción a Python

## Ofertas de trabajo en Infojobs:





# Introducción a Python

## ¿CÓMO VAMOS A TRABAJAR EN EL TALLER?

- Plataforma Anaconda (Disponible en Windows, MAC y Linux) → Python 2.7.
- Transparencias para describir los contenidos teóricos más generales.
- Scripts de ejemplos.



# Introducción a Python

## Formas de trabajar en Python:

1) Con el interprete directamente:



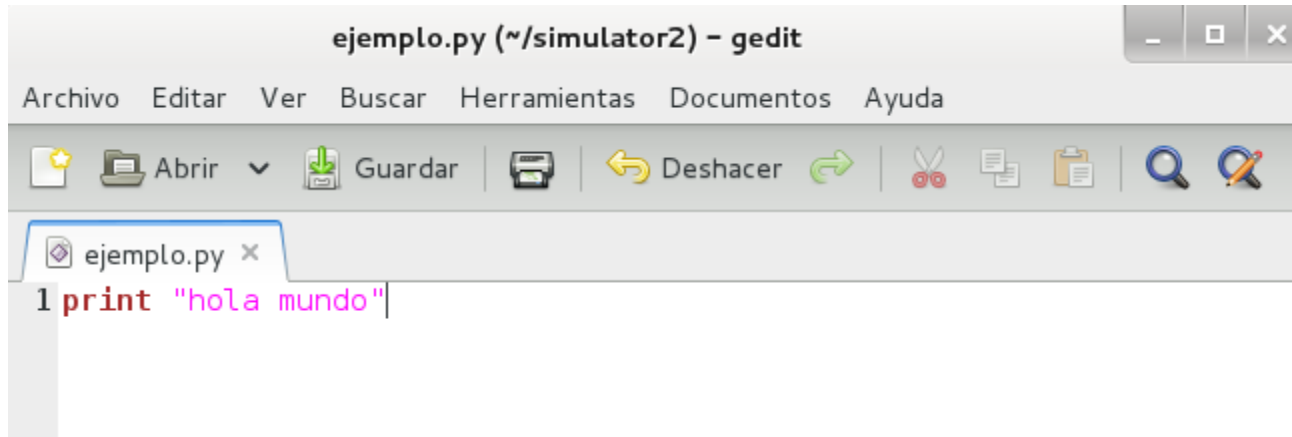
```
dany@debian: ~/simulator2
Archivo  Editar  Ver  Buscar  Terminal  Pestañas  Ayuda
dany@debian: ~/simulator2  x  dany@debian: ~/simulator2  x  dany@debian: ~/simulator2  x
dany@debian:~/simulator2$ python
Python 2.7.3 (default, Mar 14 2014, 11:57:14)
[GCC 4.7.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



# Introducción a Python

## Formas de trabajar en Python:

2) Procesador de texto, creamos scripts (.py) → Por ejemplo gedit (Linux), o bloc de notas en Windows:



Las instrucciones se interpretan una a una. Si hay un error nos saltará.

Para lanzar el script ponemos en el terminal:

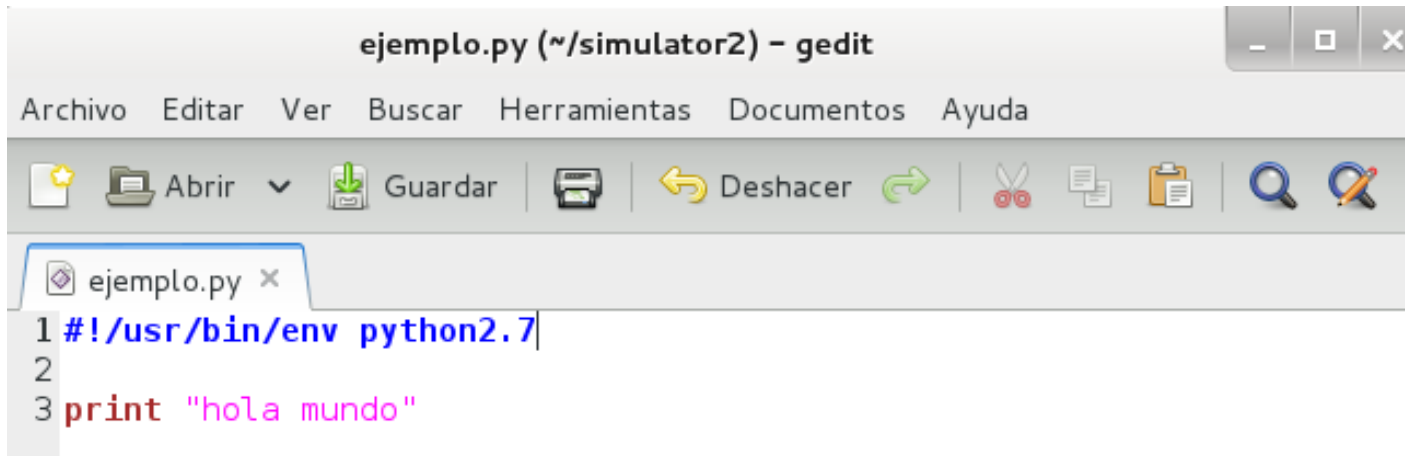
```
python ejemplo.py
```



# Introducción a Python

## Formas de trabajar en Python:

3) Lanzamos el script como un programa (sistemas operativos Linux)

A screenshot of a gedit text editor window titled "ejemplo.py (~simulator2) - gedit". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Herramientas", "Documentos", and "Ayuda". Below the menu is a toolbar with icons for "Abrir", "Guardar", "Deshacer", and other functions. A single tab labeled "ejemplo.py" is open, showing the following code:

```
1 #!/usr/bin/env python2.7
2
3 print "hola mundo"
```

Tenemos que dar permisos de ejecución al script mediante el comando:

```
chmod +x ejemplo.py
```

Ahora podemos lanzar el script de la siguiente forma:

```
./ejemplo.py
```



# Introducción a Python

## Formas de trabajar en Python: **Interprete ipython**

Es uno de los interpretes interactivos más utilizados en Python.

```
IP IPython (Py 2.7)
1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 3.2.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: print "Hola mundo"
Hola mundo

In [2]: i = 5

In [3]: print i
5

In [4]: _
```

```
Terminal de IPython
IP: Terminal 1/A x

Python 2.7.12 |Anaconda 4.1.1 (64-bit)| (default, Jun 29 2016,
11:07:13) [MSC v.1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra
details.
%gui       -> A brief reference about the graphical user interface.

In [1]:
```



# Introducción a Python

## Formas de trabajar en Python: **Anaconda**

- Distribución de Python que incluye los 100 paquetes de Python (también R) más usados.
- Entre esos 100 paquetes están los que vamos a ver en este curso → Numpy y Matplotlib.
- Se distribuye forma gratuita por la empresa Continuum Analytics.  
<https://www.continuum.io/downloads>
- Dentro de la distribución Anaconda se incluye iPython y Spyder (entorno de programación que vamos a utilizar).  
<http://pythonhosted.org/spyder/>



# Introducción a Python

## Entorno de Programación Spyder

Barra de herramientas

The screenshot shows the Spyder Python IDE interface. The main editor window on the left contains a Python script with a docstring. The top menu bar includes options like Archivo, Editar, Buscar, Código fuente, Ejecutar, Depurar, Terminales, Herramientas, Ver, and Ayuda. A toolbar with various icons is located below the menu bar. The right side of the interface features several panels: the 'Explorador de variables' (Variables Explorer) panel shows a table with one variable 'i' of type 'int' and value '5'; the 'Inspector de objetos' (Object Inspector) panel is empty; the 'Explorador de archivos' (File Explorer) panel is empty; and the 'Terminal de IPython' (IPython Terminal) panel shows the command prompt with the input 'In [5]: i = 5' and 'In [6]:'. The status bar at the bottom displays 'Permisos: RW', 'Fin de línea: CRLF', 'Codificación: UTF-8', 'Línea: 7', 'Columna: 1', and 'Memoria: 80 %'.

ZONA PARA ESCRIBIR  
LOS SCRIPTS

Visualizar variables,  
objetos y archivos

INTERPRETE IPYTHON

| Nombre | Tipo | Tamaño | Valor |
|--------|------|--------|-------|
| i      | int  | 1      | 5     |

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Jul 11 17:40:48 2016
4
5 @author: dany
6 """
7
8
```

```
In [4]: %clear
In [5]: i = 5
In [6]:
```

Permisos: RW    Fin de línea: CRLF    Codificación: UTF-8    Línea: 7    Columna: 1    Memoria: 80 %

# ¿Qué vamos a ver?



- 1) Variables (general)
- 2) Listas
- 3) Tuplas
- 4) Diccionarios
- 5) Operaciones básicas
- 6) Control de flujo
- 7) Funciones
- 8) Módulos y paquetes
- 9) Manejo de archivos
- 10) Clases





# Variables en Python

Las variables no tienen tipo → El tipo se asigna cuando se declara y el tipo se puede cambiar sobre la marcha (**TIPADO DINÁMICO**).

Todas la variables en realidad son **objetos**.

Función “type” nos indica el tipo de la variable.

## Ejemplos:

```
In [1]: var1 = 1
In [2]: var2 = 1.2
In [3]: var3 = "hola"
In [4]: type(var1)
Out[4]: int
In [5]: type(var2)
Out[5]: float
In [6]: type(var3)
Out[6]: str
```



# Variables en Python

**TENED SIEMPRE EN LA MENTE QUE TODO EN PYTHON ES UN OBJETO**

*Objects* are Python's abstraction for data. All data in a Python program is represented by objects or by relations between objects. (In a sense, and in conformance to Von Neumann's model of a "stored program computer," code is also represented by objects.)



**Los objetos tienen atributos (variables) y métodos (acciones/funciones)**



# Variables en Python

Algunas reglas para los nombres de las variables:

- Python distingue entre mayúsculas y minúsculas.
- Podemos utilizar `_`, pero se recomienda no utilizarlo al comienzo del nombre de la variable.
- Las variables no pueden empezar por números.
- Tampoco se pueden utilizar las palabras claves de Python.

## Ejemplos:

```
In [3]: 12variables = 10
File "<ipython-input-3-396548a82935>", line 1
      12variables = 10
          ^
SyntaxError: invalid syntax
```

```
In [4]: Var1 = 10
```

```
In [5]: var1 = 4
```

```
In [6]: Var1
Out[6]: 10
```

```
In [7]: var1
Out[7]: 4
```

|          |         |        |        |       |
|----------|---------|--------|--------|-------|
| and      | del     | from   | not    | while |
| as       | elif    | global | or     | with  |
| assert   | else    | if     | pass   | yield |
| break    | except  | import | print  |       |
| class    | exec    | in     | raise  |       |
| continue | finally | is     | return |       |
| def      | for     | lambda | try    |       |



# Variables en Python

Imprimir variables en Python → print (si utilizamos el interprete interactivo no hace falta).

## Ejemplos:

```
In [2]: entero=1
```

```
In [3]: print entero  
1
```

```
In [4]: print "El numero entero es %d" %entero  
El numero entero es 1
```

## Errores de Programación:

```
In [1]: print('hola' + 7)  
Traceback (most recent call last):  
  
  File "<ipython-input-1-f6289185b7c8>", line 1, in <module>  
    print('hola' + 7)  
  
TypeError: cannot concatenate 'str' and 'int' objects
```



# Listas en Python

Es una lista de datos, el tipo del contenido depende de como lo definamos → Se utilizan muchísimo en Python → **RECORDAD ES UN OBJETO!!!**

## Ejemplos:

`lista = ["nombre", "apellidos", 31, Portugal]` # pueden contener varios tipos de datos

❖ Añadir un elemento → `append()`

`lista.append("ingeniero")` # ingeniero se añade al final

❖ Insertar un elemento en una posición determinada → `insert(posición, elemento)`

`lista.insert(2, "en paro")`

# Eliminar un elemento `remove(elemento)`

`lista.remove("nombre")` # eliminamos la cadena nombre de la lista, si no está el elemento → error



# Listas en Python

## Más ejemplos:

```
In [6]: print lista1  
[1, 2, 4, 5]
```

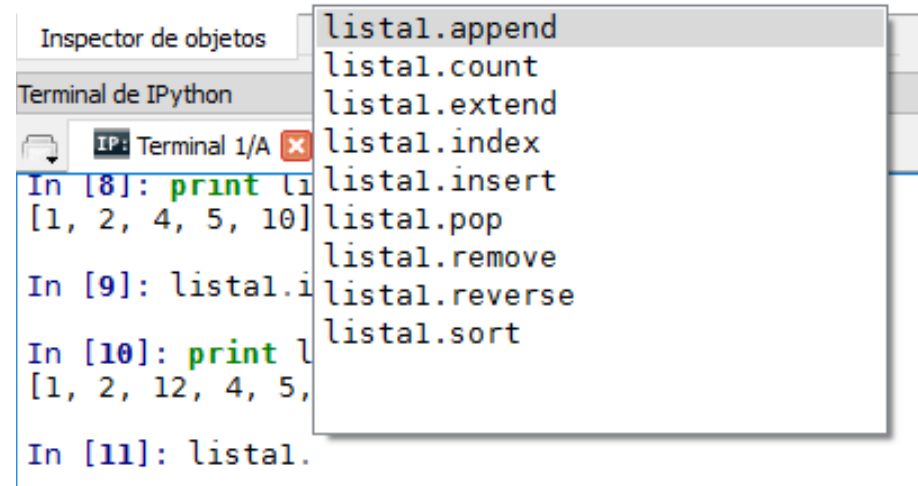
```
In [7]: lista1.append(10)
```

```
In [8]: print lista1  
[1, 2, 4, 5, 10]
```

```
In [9]: lista1.insert(2, 12)
```

```
In [10]: print lista1  
[1, 2, 12, 4, 5, 10]
```

Si pulsamos el tabulador nos indican todas las opciones que tenemos:



# Listas en Python

## Explorador de variables:

| Explorador de variables |      |        |   |
|-------------------------|------|--------|---|
| Nombre                  | Tipo | Tamaño | Valor                                     |
| lista2                  | list | 4      | ['Spain', 'Portugal', 'England', 'Italy'] |

Inspector de objetos   Explorador de variables   Explorador de archivos

Terminal de IPython

IP: Terminal 1/A

```
In [15]: lista2 = ["Spain", "Portugal", "England", "Italy"]  
In [16]:
```



# Listas en Python

## Más ejemplos:

❖ Obtener el elemento de una posición de la lista

lista[2] # obtenemos el tercer elemento de la lista

→ OJO con que no exista ningún elemento en dicha posición

→ El primer elemento de una lista es el elemento 0

```
In [15]: lista2 = ["Spain", "Portugal", "England", "Italy"]
```

```
In [16]: lista2[2]
```

```
Out[16]: 'England'
```

```
In [17]: lista2[1]
```

```
Out[17]: 'Portugal'
```

```
In [18]: lista2[0]
```

```
Out[18]: 'Spain'
```

```
In [20]: lista2[10]
```

```
Traceback (most recent call last):
```

```
File "<ipython-input-20-bf385002c90e>", line 1, in <module>
```

```
lista2[10]
```

```
IndexError: list index out of range
```

```
In [21]:
```





# Listas en Python

## Slicing: Obtener varias posiciones de una lista a la vez [:]

Sequences also support slicing: `a[i:j]` selects all items with index  $k$  such that  $i \leq k < j$ . When used as an expression, a slice is a sequence of the same type. This implies that the index set is renumbered so that it starts at 0.

```
In [27]: lista2[:]  
Out[27]: ['Spain', 'Portugal', 'England', 'Italy']
```

Todos

```
In [28]: lista2[1:]  
Out[28]: ['Portugal', 'England', 'Italy']
```

A partir del primer elemento

```
In [29]: lista2[:3]  
Out[29]: ['Spain', 'Portugal', 'England']
```

Desde el 0 al 2

```
In [30]: lista2[2:3]  
Out[30]: ['England']
```

El 3 elemento

Con el índice -1 accedemos a la última posición de la lista → Probar!



# Listas en Python

Podemos referenciar las posiciones de la lista tomando como referencia el último elemento

```
In [1]: l1 = [1, 2, 3]
```

```
In [2]: l1[-1]  
Out[2]: 3
```

```
In [3]: l1[-2]  
Out[3]: 2
```

```
In [4]: l1[:-1]  
Out[4]: [1, 2]
```



# Listas en Python

Copiar listas: **OJO LAS LISTAS SON LISTAS ENLAZADAS → Los nombres son punteros**

Explorador de variables

| Nombre   | Tipo | Tamaño | Valor              |
|----------|------|--------|--------------------|
| numeros1 | list | 5      | [1, 2, 4, 6, 1000] |
| numeros2 | list | 5      | [1, 2, 4, 6, 1000] |

Inspector de objetos   Explorador de variables   Explorador de archivos

Terminal de IPython

IP: Terminal 1/A

```
In [58]: numeros1 = [1, 2, 4, 6]
```

```
In [59]: numeros2 = numeros1
```

```
In [60]: numeros1.append(1000)
```

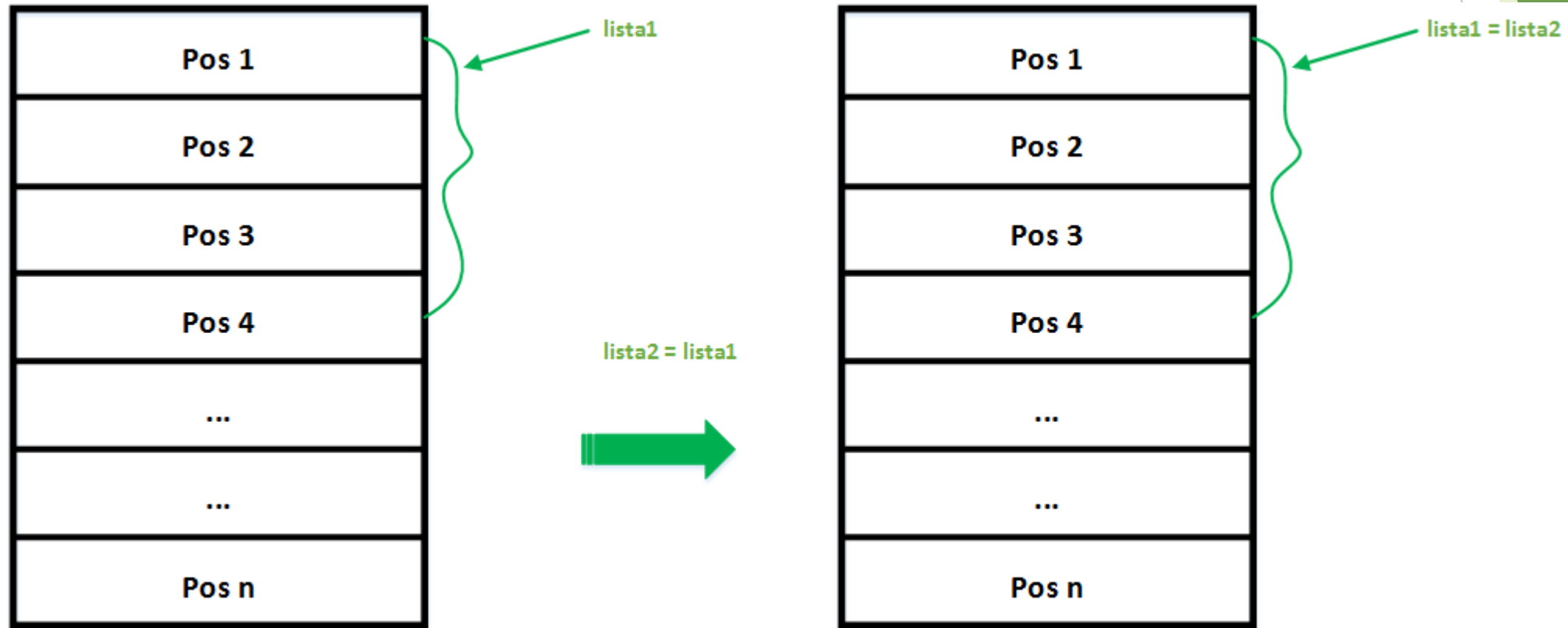
```
In [61]: print numeros1  
[1, 2, 4, 6, 1000]
```

```
In [62]: print numeros2  
[1, 2, 4, 6, 1000]
```



# Listas en Python → Concepto de Puntero

Los nombres de las listas son punteros



# Listas en Python

Copiar listas: Forma de copiar una lista sin enlazar [:]

```
In [64]: numeros3 = numeros1[:]
In [65]: numeros3.append(5000)
In [66]: numeros1
Out[66]: [1, 2, 4, 6, 1000]
In [67]: numeros3
Out[67]: [1, 2, 4, 6, 1000, 5000]
```

Otra forma: **deepcopy**

```
In [69]: import copy
In [70]: numeros4 = copy.deepcopy(numeros1)
In [71]: numeros4.append(7000)
In [72]: numeros1
Out[72]: [1, 2, 4, 6, 1000]
In [73]: numeros4
Out[73]: [1, 2, 4, 6, 1000, 7000]
```

Si dentro de una lista tenemos otra lista para desenlazar todo debemos utilizar Deepcopy

<https://docs.python.org/2/library/copy.html>



# Tuplas en Python

Es muy similar a una lista, pero con algunas diferencias:

- ❖ Van entre paréntesis (las listas van entre corchetes)
- ❖ No se pueden modificar, sólo lectura → **Objeto inmutable**
- ❖ Tiene dos métodos que se pueden utilizar (también se pueden utilizar con las listas):
  - Count: cuenta el número de veces que un valor está en la tupla.
  - Index: nos devuelve el índice de la posición en la que se encuentra un valor que se pasa como parámetro.

```
In [22]: t1 = (1, 2)
```

```
In [23]: t1  
Out[23]: (1, 2)
```

```
In [24]: type(t1)  
Out[24]: tuple
```

```
In [25]: t1.count(2)  
Out[25]: 1
```

```
In [26]: t1.index(1)  
Out[26]: 0
```

```
In [27]: t1[0]  
Out[27]: 1
```

```
In [28]: t1[0] = 10  
Traceback (most recent call last):
```

```
File "<ipython-input-28-19b36c976505>", line 1, in <module>  
    t1[0] = 10
```

```
TypeError: 'tuple' object does not support item assignment
```



# “Un paréntesis con la variables ...”

## Algunos comandos de Ipython muy útiles:

Clear: sirve para limpiar el terminal de Ipython

%Reset: sirve para borrar las variables de memoria. Nos preguntará si estamos seguros de que queremos borrar las variables (no se puede dar marcha atrás)

```
In [7]: %reset
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])?
```

Help: Ayuda en línea, le podemos pasar cualquier método.

```
In [8]: lista1 = [1, 2, 3]
```

```
In [9]: help(lista1.remove)
```

```
Help on built-in function remove:
```

```
remove(...)
```

```
L.remove(value) -- remove first occurrence of value.
```

```
Raises ValueError if the value is not present.
```



# Cadenas en Python (strings)

Sirve para almacenar cadenas de caracteres → Objetos inmutables.

Explorador de variables

| Nombre  | Tipo | Tamaño | Valor       |
|---------|------|--------|-------------|
| cadena1 | str  | 1      | Hola mundo  |
| cadena2 | str  | 1      | Adios mundo |

Inspector de objetos   Explorador de variables   Explorador de archivos

Terminal de IPython

IP: Terminal 3/A

```
In [16]: cadena1 = "Hola mundo"
In [17]: cadena2 = "Adios mundo"
```





# Cadenas en Python (strings)

Existen muchísimos métodos asociados a una cadena:

[http://www.tutorialspoint.com/python/python\\_strings.htm](http://www.tutorialspoint.com/python/python_strings.htm)

Algunos de ellos:

.capitalize() : Convierte en mayúscula el primer carácter.

.find("cad"): Devuelve la posición de un carácter o cadena "cad".

.replace(old, new): Reemplaza la cadena old por la new

Ejemplos:

```
In [26]: cadenal  
Out[26]: 'Hola mundo'
```

```
In [27]: cadenal.capitalize()  
Out[27]: 'Hola mundo'
```

```
In [28]: cadenal.find("o")  
Out[28]: 1
```

```
In [29]: cadenal.replace("Hola", "---")  
Out[29]: '--- mundo'
```

```
In [30]: len(cadenal)  
Out[30]: 10
```

También sirve para listas



# Cadenas en Python (strings)

Los elementos de una cadena también se pueden acceder por su índice, como si fueran una lista de caracteres.

```
In [6]: cadena = "Hola mundo"
```

```
In [7]: cadena[0]
```

```
Out[7]: 'H'
```

```
In [8]: cadena[1]
```

```
Out[8]: 'o'
```

```
In [9]: cadena[-1]
```

```
Out[9]: 'o'
```

```
In [10]: cadena[:]
```

```
Out[10]: 'Hola mundo'
```



# Diccionarios en Python

Estructura de datos más compleja (hash). Parejas key-value. Podemos verlo como una lista en la que los índices son cadenas → Objeto mutable.

```
diccionario = {'clave1': valor1, 'clave2': valor2, 'clave3': valor3}
```

```
diccionario = {'a': 1, 'b': 2, 'c': '3'}
```

| Nombre | Tipo | Tamaño | Valor   |
|--------|------|--------|---|
| dir1   | dict | 4      | {'Edad': 32, 'Nombre': 'Daniel', 'Profesion': 'Investigador', 'altura': 1.79} |
| dir2   | dict | 2      | {'a': [1, 2, 3], 'b': (1, 2)}   |

Inspector de objetos

Explorador de variables

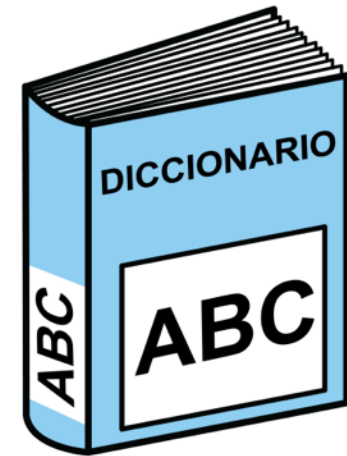
Explorador de archivos

Terminal de IPython

IP: Terminal 3/A

```
In [45]: dir1 = {'Nombre' : "Daniel", 'Edad' : 32, 'Profesion' : "Investigador", 'altura' : 1.79}
```

```
In [46]: dir2 = {'a' : [1, 2, 3], 'b' : (1,2)}
```



# Diccionarios en Python

Métodos muy útiles para el manejo de diccionarios, `.items()`, `.values()`, `.keys()`

- `diccionario.items()` # devuelve las key-value como una lista de tuplas
- `diccionario.values()` # devuelve una lista de valores
- `diccionario.keys()` # devuelve una lista de keys

```
In [47]: dir1.values()
Out[47]: [32, 'Daniel', 'Investigador', 1.79]

In [48]: dir1.keys()
Out[48]: ['Edad', 'Nombre', 'Profesion', 'altura']

In [49]: dir1.items()
Out[49]:
[('Edad', 32),
 ('Nombre', 'Daniel'),
 ('Profesion', 'Investigador'),
 ('altura', 1.79)]
```

Todos los métodos que podemos utilizar con diccionarios:

<https://docs.python.org/2/library/stdtypes.html#typesmapping>



# Diccionarios en Python

Es un tipo de datos modificable por lo que podemos añadir una key-value cuando queramos.

```
In [52]: dir1['Nacionalidad'] = "Española"
```

```
In [53]: dir1
```

```
Out[53]:
```

```
{'Edad': 32,  
 'Nacionalidad': 'Española',  
 'Nombre': 'Daniel',  
 'Profesion': 'Investigador',  
 'altura': 1.79}
```

```
In [54]: dir1['Nombre']  
Out[54]: 'Daniel'
```

Acceder a una key-value en concreto

Eliminar una key-value con “del” → se puede utilizar para borrar cualquier objeto.

```
In [56]: del(dir1['Nombre'])
```

```
In [57]: dir1
```

```
Out[57]:
```

```
{'Edad': 32,  
 'Nacionalidad': 'Española',  
 'Profesion': 'Investigador',  
 'altura': 1.79}
```

```
In [4]: var1= 2
```

```
In [5]: del(var1)
```

```
In [6]: var1
```

```
Traceback (most recent call last):
```

```
File "<ipython-input-6-43811df42071>", line 1, in <module>  
    var1
```

```
NameError: name 'var1' is not defined
```



# Constantes en Python

En Python las constantes se definen en mayúsculas. Es decir, cuando una variable se considera que no va a ser modificada se escribe en mayúsculas.

Esto sólo es un convenio entre programadores. Cuando una variable contiene un dato que no quiero modificarlo, lo ponemos en mayúsculas, pero realmente sí se puede modificar.

```
In [11]: CONSTANTE = 10
```

```
In [12]: CONSTANTE = 5
```

```
In [13]: CONSTANTE  
Out[13]: 5
```

**NO EXISTEN CONSTANTES EN PYTHON COMO EN OTROS LENGUAJES DE PROGRAMACIÓN**



# Objetos mutables y objetos inmutables

❑ En Python las variables de tipo *listas, sets y diccionarios son mutables!!*

Mutable significa que su contenido **SÍ** puede variar.

```
In [31]: l1 = [1, 2, 3]
```

```
In [32]: id(l1)  
Out[32]: 195727112L
```

```
In [33]: l1.append(10)
```

```
In [34]: id(l1)  
Out[34]: 195727112L
```

| Class     | Description                          | Immutable? |
|-----------|--------------------------------------|------------|
| bool      | Boolean value                        | ✓          |
| int       | integer (arbitrary magnitude)        | ✓          |
| float     | floating-point number                | ✓          |
| list      | mutable sequence of objects          |            |
| tuple     | immutable sequence of objects        | ✓          |
| str       | character string                     | ✓          |
| set       | unordered set of distinct objects    |            |
| frozenset | immutable form of set class          | ✓          |
| dict      | associative mapping (aka dictionary) |            |



# Antes de continuar ... Scripts en Python

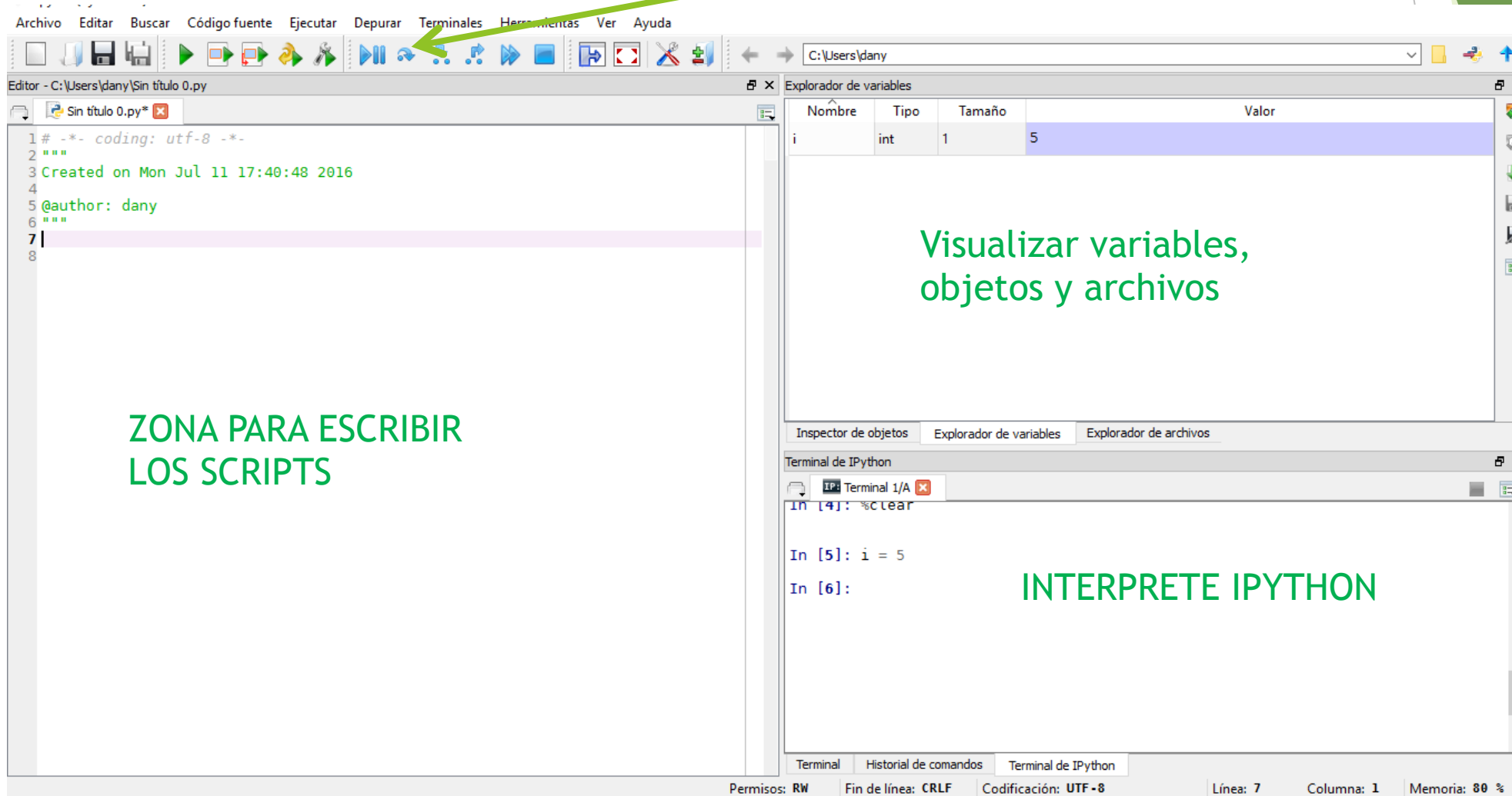
- ❑ Un script no es más que una ***secuencia de código en Python*** que se ejecuta de forma secuencial.
- ❑ Los scripts en Python tiene la **extensión .py**
- ❑ Se pueden ejecutar desde un terminal ipython con el comando *run* (ojo hay con el directorio donde nos encontramos).
- ❑ Los ***comentarios en un script comienza con el carácter #***. Líneas de código que no se ejecuta, se utilizar para dejar notas sobre lo que hace el código.
- ❑ Los scripts normalmente se empiezan importando los paquetes o módulos que se van a utilizar posteriormente (lo veremos más adelante).





# Scripts de Python en Spyder

Barra de herramientas



The screenshot shows the Spyder Python IDE interface. The main editor window on the left contains a Python script with a docstring. The top toolbar is labeled 'Barra de herramientas'. The right sidebar contains the 'Explorador de variables' (Variables Explorer) and the 'Terminal de IPython'. The 'Explorador de variables' shows a variable 'i' of type 'int' with a value of 5. The 'Terminal de IPython' shows the execution of the script, including the assignment 'i = 5'. The status bar at the bottom indicates the file encoding as UTF-8 and the current line and column.

ZONA PARA ESCRIBIR  
LOS SCRIPTS

Visualizar variables,  
objetos y archivos

INTERPRETE IPYTHON

| Nombre | Tipo | Tamaño | Valor |
|--------|------|--------|-------|
| i      | int  | 1      | 5     |

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Jul 11 17:40:48 2016
4
5 @author: dany
6 """
7
8
```

```
In [4]: %clear

In [5]: i = 5

In [6]:
```

Permisos: RW Fin de línea: CRLF Codificación: UTF-8 Línea: 7 Columna: 1 Memoria: 80 %

# Mi primer script en Python



-Ejecutar todo el script

Spyder (Python 2.7)

Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Herramientas Ver Ayuda

Editor - C:\Users\dany\script\_python\script1.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Jul 11 17:40:48 2016
4
5 @author: dany
6 """
7
8 # Aqui comienza el scrip que está vacío
9
10
11 print "Hola mundo"
```

Explorador de archivos

| Nombre     | Tamaño    | Tipo       | Última modificación |
|------------|-----------|------------|---------------------|
| script1.py | 155 bytes | py Fichero | 11/07/2016 18:13:56 |

Inspector de objetos Explorador de variables Explorador de archivos

Terminal de IPython

```
In [15]: runfile('C:/Users/dany/script_python/script1.py',
wdir='C:/Users/dany/script_python')
Hola mundo

In [16]:
```

Resultado



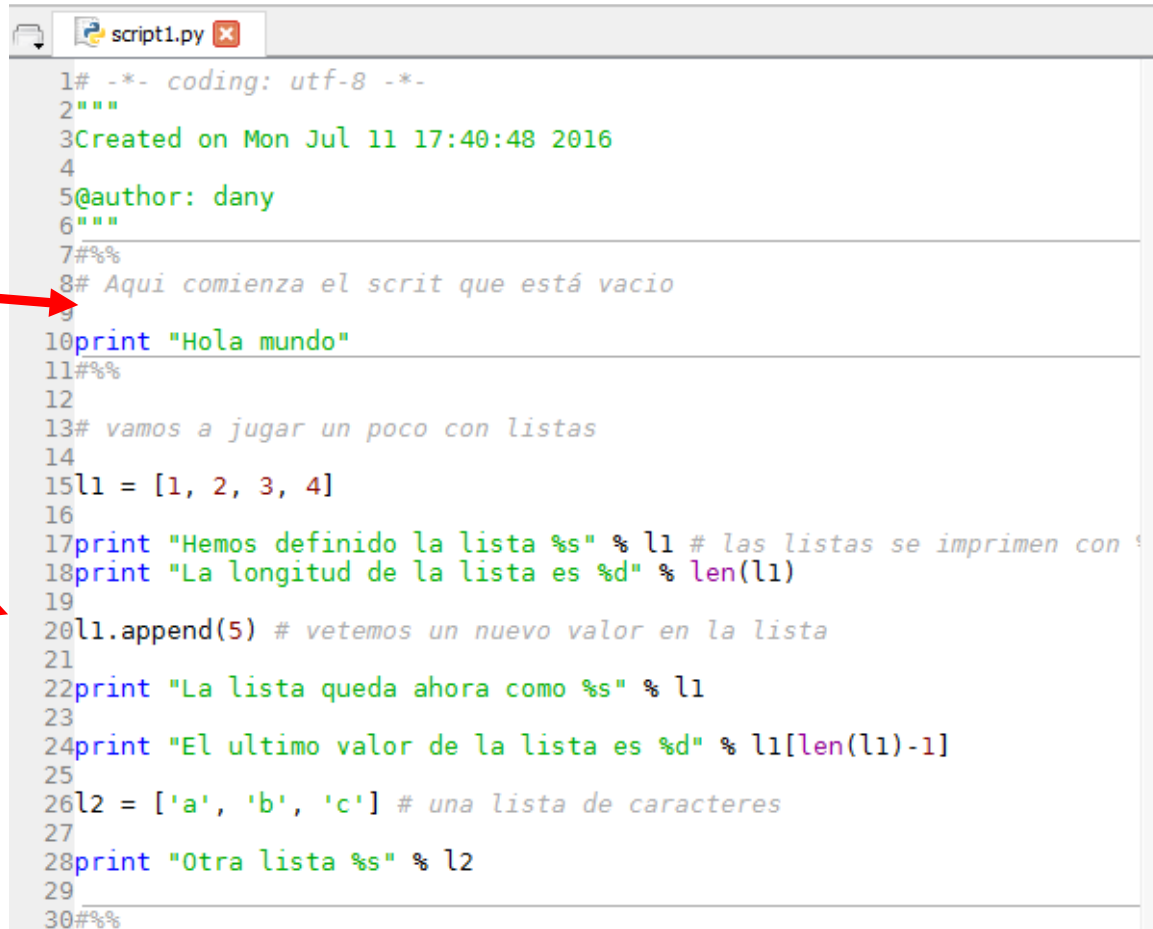
# Mi primer script en Python

❑ Abrir → **script1.py**

❑ `#%%` Para definir celdas → Muy interesante podemos ejecutar el código de forma independiente por celda.

Celda 1

Celda 2



```
1# -*- coding: utf-8 -*-
2"""
3Created on Mon Jul 11 17:40:48 2016
4
5@author: dany
6"""
7#%%
8# Aqui comienza el scrip que está vacío
9
10print "Hola mundo"
11#%%
12
13# vamos a jugar un poco con listas
14
15l1 = [1, 2, 3, 4]
16
17print "Hemos definido la lista %s" % l1 # las listas se imprimen con %s
18print "La longitud de la lista es %d" % len(l1)
19
20l1.append(5) # vetemos un nuevo valor en la lista
21
22print "La lista queda ahora como %s" % l1
23
24print "El ultimo valor de la lista es %d" % l1[len(l1)-1]
25
26l2 = ['a', 'b', 'c'] # una lista de caracteres
27
28print "Otra lista %s" % l2
29
30#%#
```



# Mi primer script en Python



Ejecutar la celda actual

Spyder (Python 2.7)

Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Herramientas Ver Ayuda

Editor - C:\Users\dany\script\_python\script1.py

```
1# -*- coding: utf-8 -*-
2"""
3Created on Mon Jul 11 17:40:48 2016
4
5@author: dany
6"""
7###
8# Aqui comienza el script que está vacío
9
10print "Hola mundo"
11###
12
13# vamos a jugar un poco con listas
14
15l1 = [1, 2, 3, 4]
16
17print "Hemos definido la lista %s" % l1 # las listas se imprimen con %s
18print "La longitud de la lista es %d" % len(l1)
19
20l1.append(5) # vamos un nuevo valor en la lista
21
22print "La lista queda ahora como %s" % l1
23
24print "El último valor de la lista es %d" % l1[len(l1)-1]
25
26l2 = ['a', 'b', 'c'] # una lista de caracteres
27
28print "Esta lista es %s" % l2
```

Explorador de variables

| Nombre | Tipo | Tamaño | Valor   |
|--------|------|--------|---|
| cad1   | str  | 1      | Hay una leyenda que recorre el mundo entero ... |
| dir1   | dict | 2      | {'Edad': 32, 'Nombre': 'Antonio'}               |
| l1     | list | 5      | [1, 2, 3, 4, 5]                                 |
| l2     | list | 3      | ['a', 'b', 'c']                                 |

Inspector de objetos Explorador de variables Explorador de archivos

Terminal de IPython

```
...: print "Hemos definido la lista %s" % l1 # las listas se imprimen con %s
...: print "La longitud de la lista es %d" % len(l1)
...: l1.append(5) # vamos un nuevo valor en la lista
...: print "La lista queda ahora como %s" % l1
...: print "El último valor de la lista es %d" % l1[len(l1)-1]
...: l2 = ['a', 'b', 'c'] # una lista de caracteres
...: print "Esta lista es %s" % l2
```

Comentarios

nero

# Mi primer script en Python

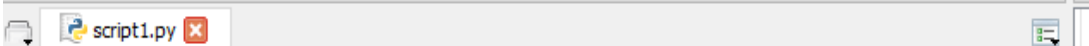


Spyder (Python 2.7)

Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Herramientas Ver Ayuda



Editor - C:\Users\dany\script\_python\script1.py



```
1# -*- coding: utf-8 -*-
2"""
3Created on Mon Jul 11 17:40:48 2016
4
5@author: dany
6"""
7#%%
8# Aqui comienza el script que está vacío
9
10print "Hola mundo"
11#%%
12
13# vamos a jugar un poco con listas
14
15l1 = [1, 2, 3, 4]
16
17print "Hemos definido la lista %s" % l1 # las listas se imprimen con %s
18print "La longitud de la lista es %d" % len(l1)
19
20l1.append(5) # vamos un nuevo valor en la lista
21
22print "La lista queda ahora como %s" % l1
23
24print "El último valor de la lista es %d" % l1[len(l1)-1]
25
26l2 = ['a', 'b', 'c'] # una lista de caracteres
27
28print "Esta lista es %s" % l2
```

Ruta actual

C:\Users\dany\script\_python

| Nombre | Tipo | Tamaño | Valor   |
|--------|------|--------|---|
| cad1   | str  | 1      | Hay una leyenda que recorre el mundo entero ... |
| dir1   | dict | 2      | {'Edad': 32, 'Nombre': 'Antonio'}               |
| l1     | list | 5      | [1, 2, 3, 4, 5]                                 |
| l2     | list | 3      | ['a', 'b', 'c']                                 |

Inspector de objetos

Explorador de variables

Explorador de archivos

Terminal de IPython

IP: Terminal 1/A

```
.... print "Hemos definido la lista %s" % l1 # las listas se imprimen con %s
.... print "La longitud de la lista es %d" % len(l1)
....
.... l1.append(5) # vamos un nuevo valor en la lista
....
.... print "La lista queda ahora como %s" % l1
.... print "El último valor de la lista es %d" % l1[len(l1)-1]
.... l2 = ['a', 'b', 'c'] # una lista de caracteres
....
```



# Mi primer script en Python

- Desde el terminal de ipython puedo ejecutar un script con el comando:
  - `run archivo.py`



# Continuamos con Python ... Operaciones básicas

## Suma y resta:

| Explorador de variables |      |        |       |
|-------------------------|------|--------|-------|
| Nombre                  | Tipo | Tamaño | Valor |
| num1                    | int  | 1      | 5     |
| num2                    | int  | 1      | 10    |
| num3                    | int  | 1      | 15    |
| num4                    | int  | 1      | -5    |

|                      |                         |                        |
|----------------------|-------------------------|------------------------|
| Inspector de objetos | Explorador de variables | Explorador de archivos |
|----------------------|-------------------------|------------------------|

| Terminal de IPython |   |
|---------------------|---|
| IP: Terminal 1/A    | X |

```

In [2]: num1 = 5
In [3]: num2 = 10
In [4]: num3 = num1 + num2
In [5]: num3
Out[5]: 15
In [6]: num4 = num1 - num2
In [7]: num4
Out[7]: -5

```

## Multiplicación y división:

```

Terminal de IPython
IP: Terminal 1/A X

In [9]: num5 = num1 * num2
In [10]: num5
Out[10]: 50

In [11]: num6 = num1 / num2
In [12]: num6
Out[12]: 0

In [13]: num1 = 5.0
In [14]: num6 = num1 / num2
In [15]: num6
Out[15]: 0.5

```

*Ojo con las operaciones cuyo resultado es un número flotante.*

## Resto y potencia:

```

In [17]: num7 = num1 % num2
In [18]: num7
Out[18]: 5.0

In [19]: num8 = 5**2
In [20]: num8
Out[20]: 25

```



# Continuamos con Python ... Operaciones básicas

## Ojo con las operaciones matemáticas con listas:

| Explorador de variables |      |        |                              |
|-------------------------|------|--------|------------------------------|
| Nombre                  | Tipo | Tamaño | Valor                        |
| l1                      | list | 4      | [1, 2, 3, 4]                 |
| l2                      | list | 4      | [10, 11, 12, 13]             |
| l3                      | list | 8      | [1, 2, 3, 4, 10, 11, 12, 13] |
| l4                      | list | 8      | [1, 2, 3, 4, 1, 2, 3, 4]     |

Sumas y multiplicaciones son en realidad concatenaciones!!

Más adelante veremos vectores (Numpy)

```

In [50]: l1 = [1, 2, 3, 4]
In [51]: l2 = [10, 11, 12, 13]
In [52]: l3 = l1 + l2
In [53]: l4 = l1 * l2
Traceback (most recent call last):
  File "<ipython-input-53-fbd0eeed4946>", line 1, in <module>
    l4 = l1 * l2
TypeError: can't multiply sequence by non-int of type 'list'

In [54]: l4 = l1 * 2

```





# Conversión de tipos

```
In [26]: a = 5
```

```
In [27]: a = float(a)
```

```
In [28]: a  
Out[28]: 5.0
```

```
In [29]: a = str(a)
```

```
In [30]: a  
Out[30]: '5.0'
```

```
In [31]: a = list(a)
```

```
In [32]: a  
Out[32]: ['5', '.', '0']
```

```
In [38]: cad = "Hola"
```

```
In [39]: cad = list(cad)
```

```
In [40]: cad  
Out[40]: ['H', 'o', 'l', 'a']
```

```
In [41]: l1 = list((1,2))
```

```
In [42]: l1  
Out[42]: [1, 2]
```



# Control de flujo en Python

Lo vamos a ver con un script → **Abrir script 2**

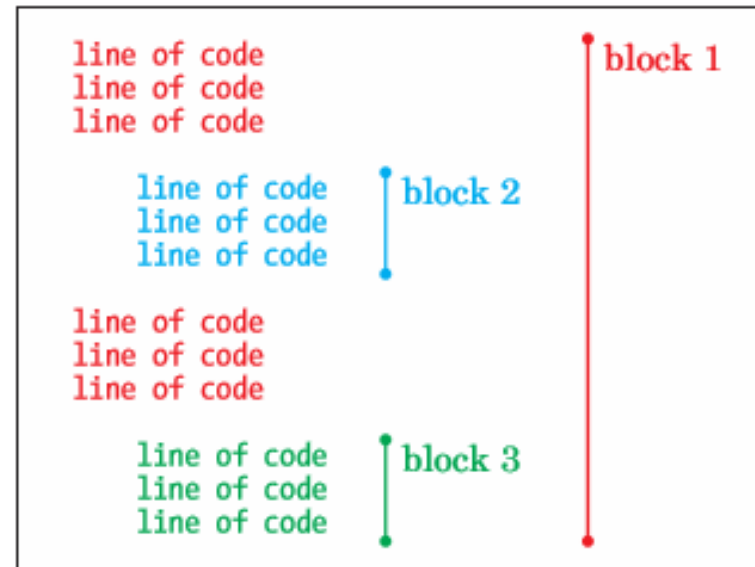


¿raw\_input()  
vs input?

Detalles sobre las sentencias if - else:

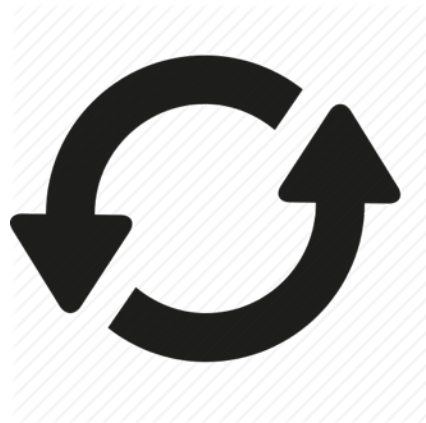
- Python es un lenguaje “indentado” - sangrado → “Ojo con los tabuladores” → Puede ser lioso al principio.
- No olvidar los “:” después de la sentencias if y esle.

Función raw\_input() → Sirve para obtener información del terminal, cualquier tipo de información puede ser tomada → No olvidar que el tipo de una variable se define sobre la marcha.



# Bucles “for” en Python

- Son un poco diferentes a los bucles for en otros lenguajes de programación (C, Matlab).
- Los bucles for recorren iterables tales como listas o cadenas.
- **Abrir script3**
- La sentencia `in range(inicio, stop)` es muy útil → Ojo `stop - 1`! Si no ponemos inicio, considera el cero como el valor de inicio.
- También podemos dar un paso `in range(inicio, stop, paso)`, si el paso es negativo iríamos hacia atrás. → **Ojo no podemos dar como paso un float (numpy). Más adelante “arrays”.**

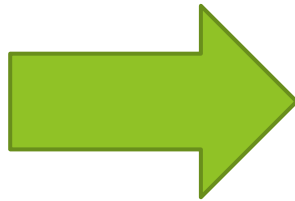


# List Comprehensions

- Python permite crear listas utilizando bucles for de una forma muy potente.
- La idea es emular como los matemáticos definen listas de valores.

## Definición matemática

```
S = {x2 : x in {0 ... 9}}
V = (1, 2, 4, 8, ..., 212)
M = {x | x in S and x even}
```



## Python

```
>>> S = [x**2 for x in range(10)]
>>> V = [2**i for i in range(13)]
>>> M = [x for x in S if x % 2 == 0]
>>>
>>> print S; print V; print M
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]
[0, 4, 16, 36, 64]
```

→ En script3.py

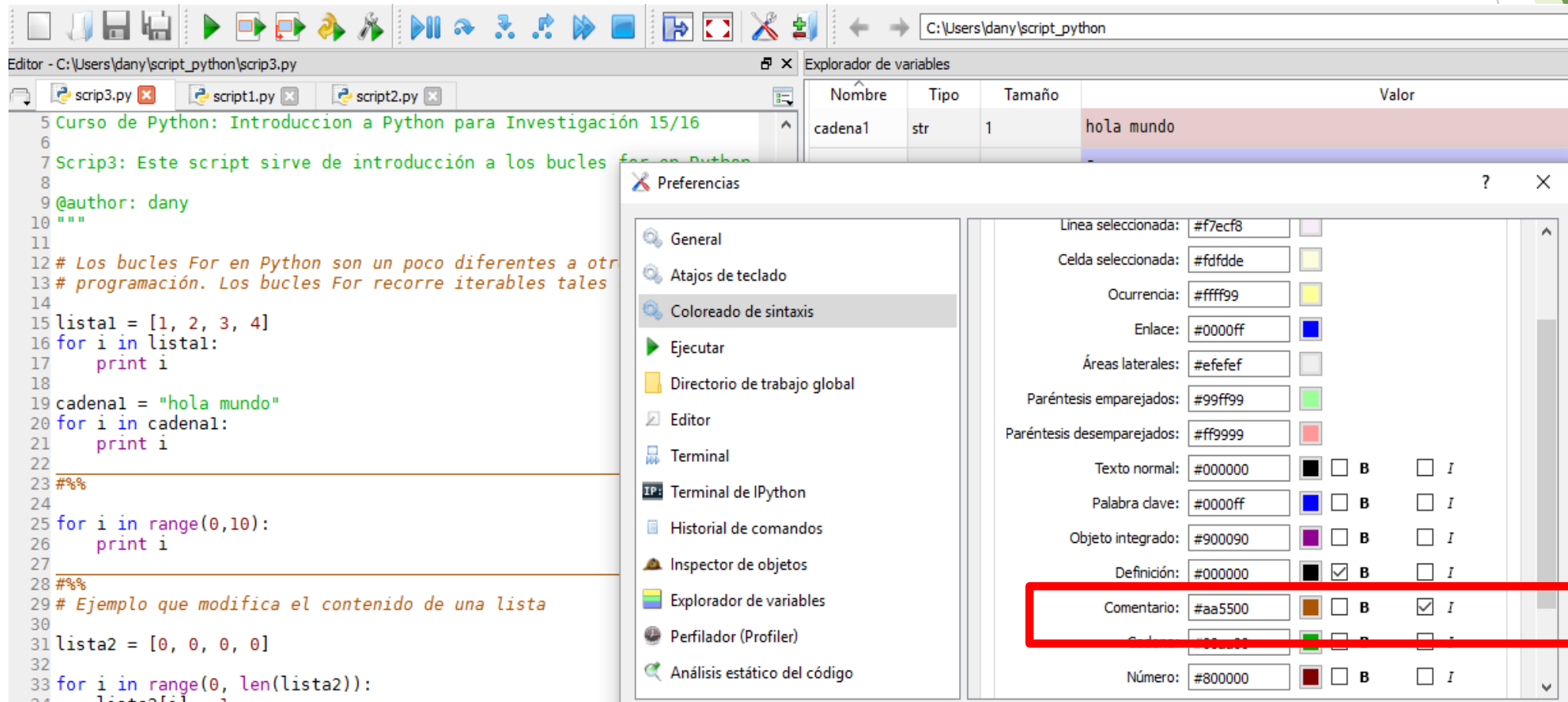
*### podemos hacerlo todo el bucle en una sola linea!!!!!! --> List comprehensions*

`[x * x for x in range(10)]` # ojo! solo es válido si el resultado es una lista



# Otro paréntesis ... Colores de sintaxis

- Podemos definir los colores de la sintaxis que utiliza Python en la siguiente ruta:  
Herramientas → Preferencias → Coloreado de Sintaxis : Pestaña Spyder.



The screenshot shows the Spyder IDE interface with the Preferences dialog box open. The 'Coloreado de sintaxis' (Syntax Coloring) tab is selected. The dialog displays various color settings for different code elements, with the 'Comentario' (Comment) color set to #aa5500, which is highlighted by a red rectangle.

The background shows a Python script in the editor with the following code:

```
5 Curso de Python: Introduccion a Python para Investigación 15/16
6
7 Scrip3: Este script sirve de introducción a los bucles for en Python
8
9 @author: dany
10 """
11
12 # Los bucles For en Python son un poco diferentes a otros
13 # programación. Los bucles For recorre iterables tales
14
15 lista1 = [1, 2, 3, 4]
16 for i in lista1:
17     print i
18
19 cadena1 = "hola mundo"
20 for i in cadena1:
21     print i
22
23 #%%
24
25 for i in range(0,10):
26     print i
27
28 #%%
29 # Ejemplo que modifica el contenido de una lista
30
31 lista2 = [0, 0, 0, 0]
32
33 for i in range(0, len(lista2)):
34     lista2[i] = 1
```

The 'Explorador de variables' (Variable Explorer) panel on the right shows a variable named 'cadena1' of type 'str' with a value of 'hola mundo'.



# Bucles “while” in Python

→ [Abrir script4.py](#)

Otras sentencias útiles para utilizarlas con bucles:

- ❖ `break` → para salir del bucle en cualquier momento.
- ❖ `continue` → para seguir con la siguiente iteración, no sigue con lo que hay debajo.
- ❖ `pass` → no hace nada. Sirve para utilizarse en partes de código que no han sido todavía escritas.



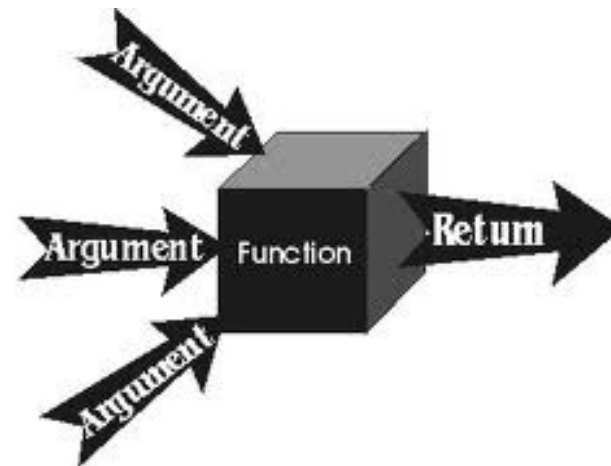
# Funciones en Python

Una función es un conjunto de sentencias agrupadas en una pieza de código que posee cierta independencia → El objetivo principal es no repetir código!!

❖ Sintaxis:

```
def funcion_suma(var1, var2):  
    resultado = var1 + var2  
    print resultado  
    return resultado
```

à Abrir script5.py



# Funciones en Python

Seguimos viendo la utilidad de las funciones:

à Abrir script6.py

à Abrir script7.py



Podemos definir funciones con ciertas variables de entrada predefinidas.





# Funciones en Python

Podemos asignar una función a una variable en Python

```
def funcion_suma(a,b):  
    return a + b  
  
y = funcion_suma  
  
print y(2,4)
```

**NO OLVIDAR:** Las funciones en Python son también objetos!!!!



# Funciones “Built-In” en Python

Funciones que siempre tenemos accesibles, vienen con el interprete de Python, el interprete las identifica y las pone en color verde → No tenemos que “importar” ningún módulo ni paquete

→ Abrir script8.py

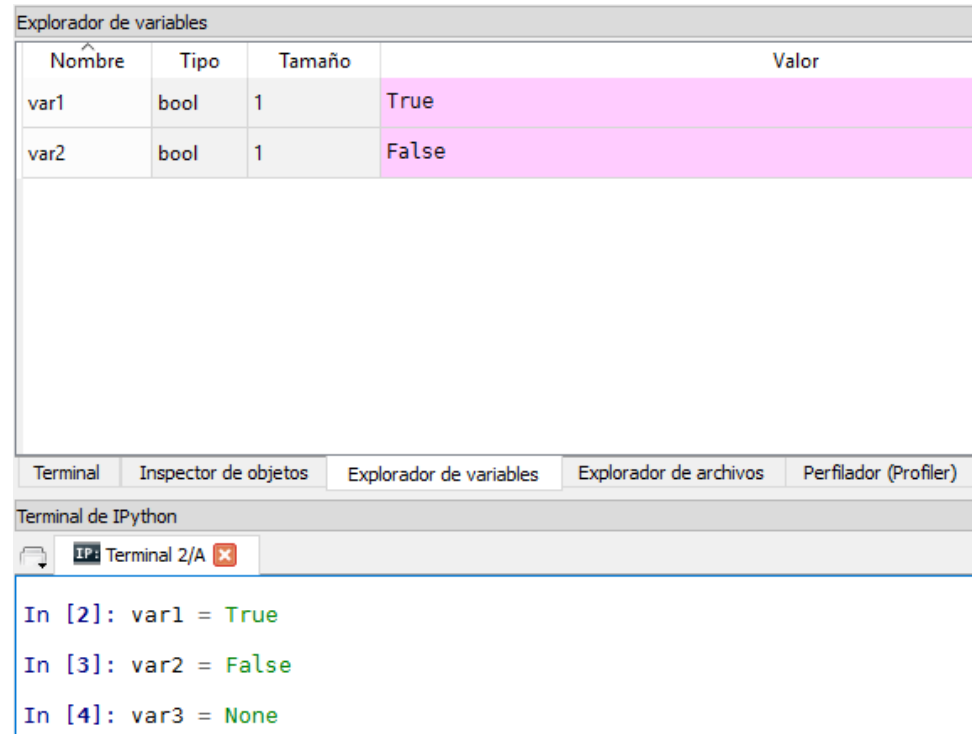
| Built-in Functions         |                          |                           |                         |                             |
|----------------------------|--------------------------|---------------------------|-------------------------|-----------------------------|
| <code>abs()</code>         | <code>dict()</code>      | <code>help()</code>       | <code>min()</code>      | <code>setattr()</code>      |
| <code>all()</code>         | <code>dir()</code>       | <code>hex()</code>        | <code>next()</code>     | <code>slice()</code>        |
| <code>any()</code>         | <code>divmod()</code>    | <code>id()</code>         | <code>object()</code>   | <code>sorted()</code>       |
| <code>ascii()</code>       | <code>enumerate()</code> | <code>input()</code>      | <code>oct()</code>      | <code>staticmethod()</code> |
| <code>bin()</code>         | <code>eval()</code>      | <code>int()</code>        | <code>open()</code>     | <code>str()</code>          |
| <code>bool()</code>        | <code>exec()</code>      | <code>isinstance()</code> | <code>ord()</code>      | <code>sum()</code>          |
| <code>bytearray()</code>   | <code>filter()</code>    | <code>issubclass()</code> | <code>pow()</code>      | <code>super()</code>        |
| <code>bytes()</code>       | <code>float()</code>     | <code>iter()</code>       | <code>print()</code>    | <code>tuple()</code>        |
| <code>callable()</code>    | <code>format()</code>    | <code>len()</code>        | <code>property()</code> | <code>type()</code>         |
| <code>chr()</code>         | <code>frozenset()</code> | <code>list()</code>       | <code>range()</code>    | <code>vars()</code>         |
| <code>classmethod()</code> | <code>getattr()</code>   | <code>locals()</code>     | <code>repr()</code>     | <code>zip()</code>          |
| <code>compile()</code>     | <code>globals()</code>   | <code>map()</code>        | <code>reversed()</code> | <code>__import__()</code>   |
| <code>complex()</code>     | <code>hasattr()</code>   | <code>max()</code>        | <code>round()</code>    |                             |
| <code>delattr()</code>     | <code>hash()</code>      | <code>memoryview()</code> | <code>set()</code>      |                             |



# Constantes “Built-In” en Python

Algunas constantes que podemos utilizar en Python (por defecto las tenemos)

- **False** → Valor falso para variables booleanas.
- **True** → Valor verdadero para variables booleanas.
- **None** → Se utiliza para representar la ausencia de un valor.

A screenshot of a Python IDE interface. The top panel, titled 'Explorador de variables', shows a table with two rows: 'var1' of type 'bool' with value 'True', and 'var2' of type 'bool' with value 'False'. The bottom panel, titled 'Terminal de IPython', shows three lines of code: 'In [2]: var1 = True', 'In [3]: var2 = False', and 'In [4]: var3 = None'. The terminal also shows a tab for 'IP: Terminal 2/A'.

# Funciones lambda en Python

Funciones en línea → Se define la función en una línea de la siguiente forma:

Lambda <parámetro>: expresión

à **ABRIR script9.py**

Módulo de funciones matemáticas:

<https://docs.python.org/2/library/math.html>

También se conocen como funciones anónimas

λ



# Variables globales vs Variables locales

*Las variables que creamos en las funciones son variables locales.*

Las variables que definimos fuera de las funciones son variables globales. Si queremos modificar su contenido debemos utilizar “global”. (Recordar que las listas van aparte).

→ Abrir `script10.py`

More info:

[http://www.python-course.eu/global\\_vs\\_local\\_variables.php](http://www.python-course.eu/global_vs_local_variables.php)



Aunque soy un gran programador ... A veces me equivoco 😊

Depurar código siempre es algo esencial.



#### Six Stages of Debugging

1. That can't happen.
2. That doesn't happen on my machine.
3. That shouldn't happen.
4. Why does that happen?
5. Oh, I see.
6. How did that ever work?

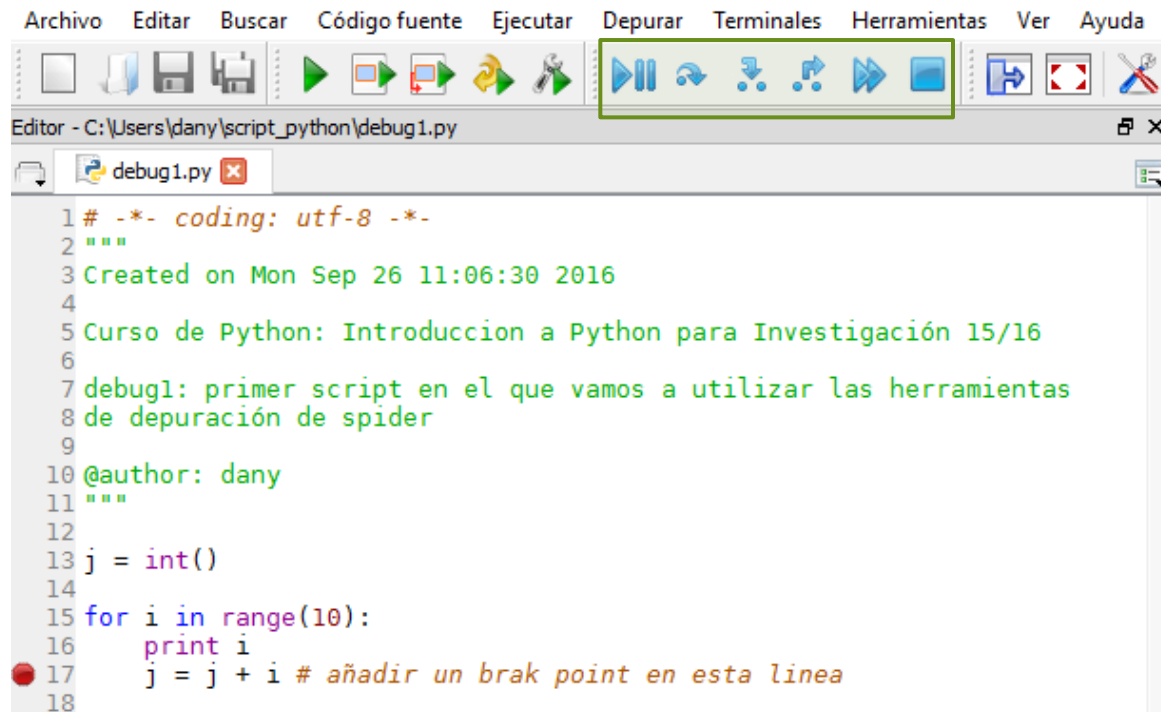


# Depurar código en Python -- Spider

Spider utiliza el depurador de Python (ipdb)

<https://github.com/gotcha/ipdb>

En spider tenemos una serie de botones que nos permiten de forma sencilla depurar scripts.



- 1) Pasar modo depuración
- 2) Ejecutar línea de código
- 3) Ingresar en la función o método
- 4) Ejecutar hasta el final de la función o método
- 5) Ejecutar hasta el siguiente breakpoint
- 6) Salir del modo depuración



# Depurar código en Python

→ Abrir debug1.py

à Abrir debug2.py

## Traceback error

En Python cuando se produce un error el terminal nos indica la traza (camino) que ha producido el error. Esto es interesante para saber dónde se ha producido el error.

→ Abrir debug3.py

```
File "C:/Users/dany/script_python/debug3.py", line 19, in <module>
    f1()

File "C:/Users/dany/script_python/debug3.py", line 16, in f1
    f2()

File "C:/Users/dany/script_python/debug3.py", line 13, in f2
    1/0

ZeroDivisionError: integer division or modulo by zero
```





# Módulos en Python → Librerías

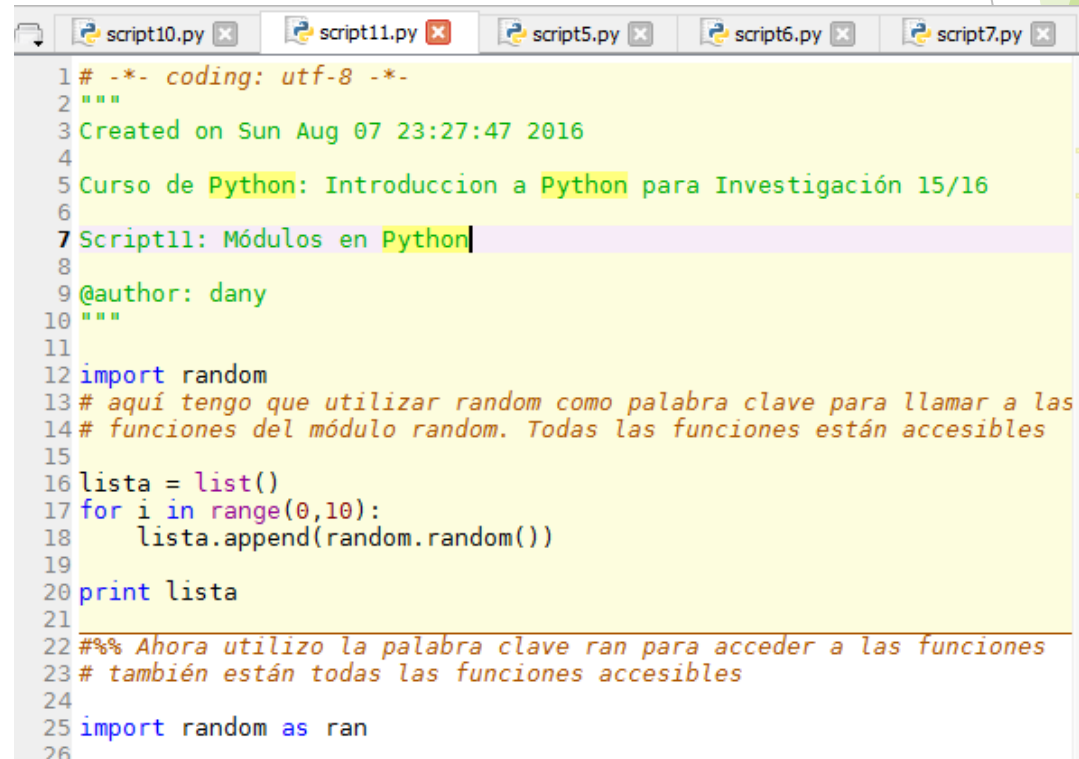
Conjunto de funciones y variables que han sido definidos en un script y que pueden ser usadas en otros scripts.

→ Abrir script11.py

Vamos a utilizar de ejemplo un módulo muy usado en Python →>random.

Información sobre el módulo random:

<https://docs.python.org/2/library/random.html>



```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun Aug 07 23:27:47 2016
4
5 Curso de Python: Introduccion a Python para Investigación 15/16
6
7 Script11: Módulos en Python
8
9 @author: dany
10 """
11
12 import random
13 # aquí tengo que utilizar random como palabra clave para llamar a las
14 # funciones del módulo random. Todas las funciones están accesibles
15
16 lista = list()
17 for i in range(0,10):
18     lista.append(random.random())
19
20 print lista
21
22 %% Ahora utilizo la palabra clave ran para acceder a las funciones
23 # también están todas las funciones accesibles
24
25 import random as ran
26
```



# Módulos en Python

Formas de importar un módulo en Python:

1) `import random` # nos importa todas las funciones definidas en random.

à Para usarlas tenemos que poner `random.<nombre_función>`

2) `from random import uniform` # solo importamos esta función.

3) `from random import *` # importamos todas las funciones.

4) `import random as ran` # podemos cambiar el nombre.

à ahora tendríamos que utilizar las funciones como `ran.<nombre_función>`

**EL OBJETIVO DE TODO ESTO ES NO LIARSE CON LOS NOMBRES Y PODER REUTILIZAR LOS NOMBRES.**

**TAMBIÉN AHORRAR TIEMPO, SI IMPORTAMOS EL MÓDULO COMPLETO LA EJECUCIÓN DEL CÓDIGO TARDARA MÁS.**

Lista de módulos útiles:

<https://wiki.python.org/moin/UsefulModules>

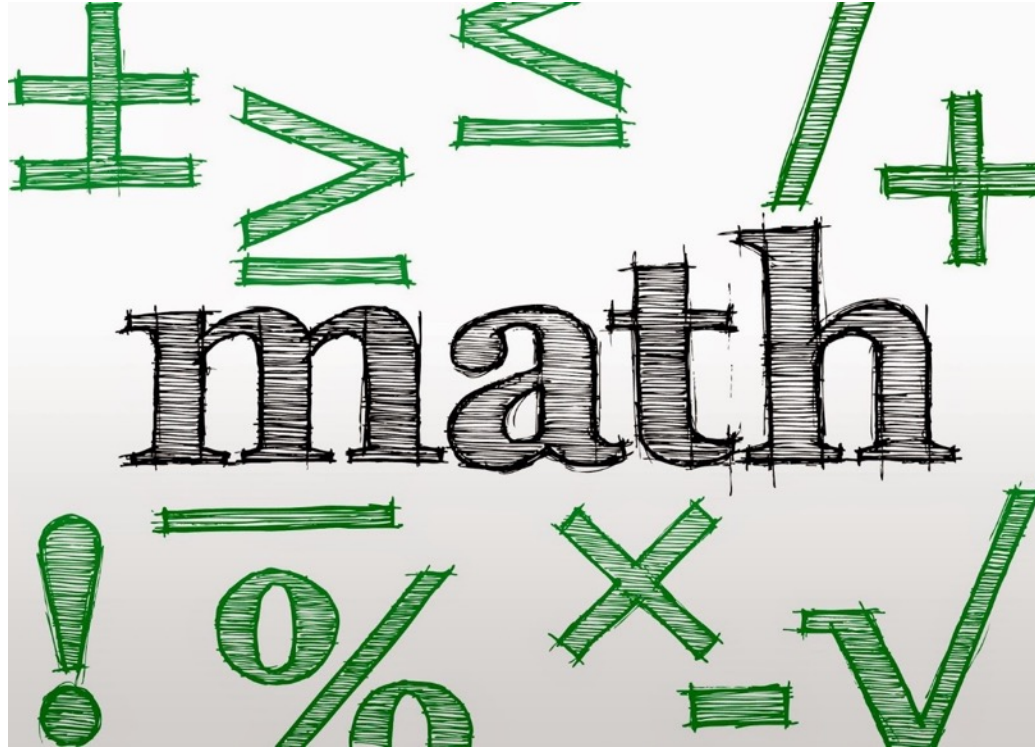


# Módulos en Python

Creamos un módulo de funciones matemáticas:

à [Abrir funciones\\_matematicas.py](#)

à [Abrir script13.py](#)



# Cómo instalar otros módulos en Spyder?

Utilizamos la aplicación pip, con el comando `pip install <nombre-modulo>`

Como lo vamos a hacer en el terminal de Ipython, tenemos que poner `!pip ...`

```
In [2]: !pip install deap
Collecting deap
  Downloading deap-1.0.2.post2.tar.gz (852kB)
Building wheels for collected packages: deap
  Running setup.py bdist_wheel for deap: started
  Running setup.py bdist_wheel for deap: finished with status 'done'
  Stored in directory: C:\Users\DaniG\AppData\Local\pip\Cache\wheels\c9\9c\cd\d52106f0148e675df35718c0efff2ecf03cc8d5bdcfb91db5
Successfully built deap
Installing collected packages: deap
Successfully installed deap-1.0.2

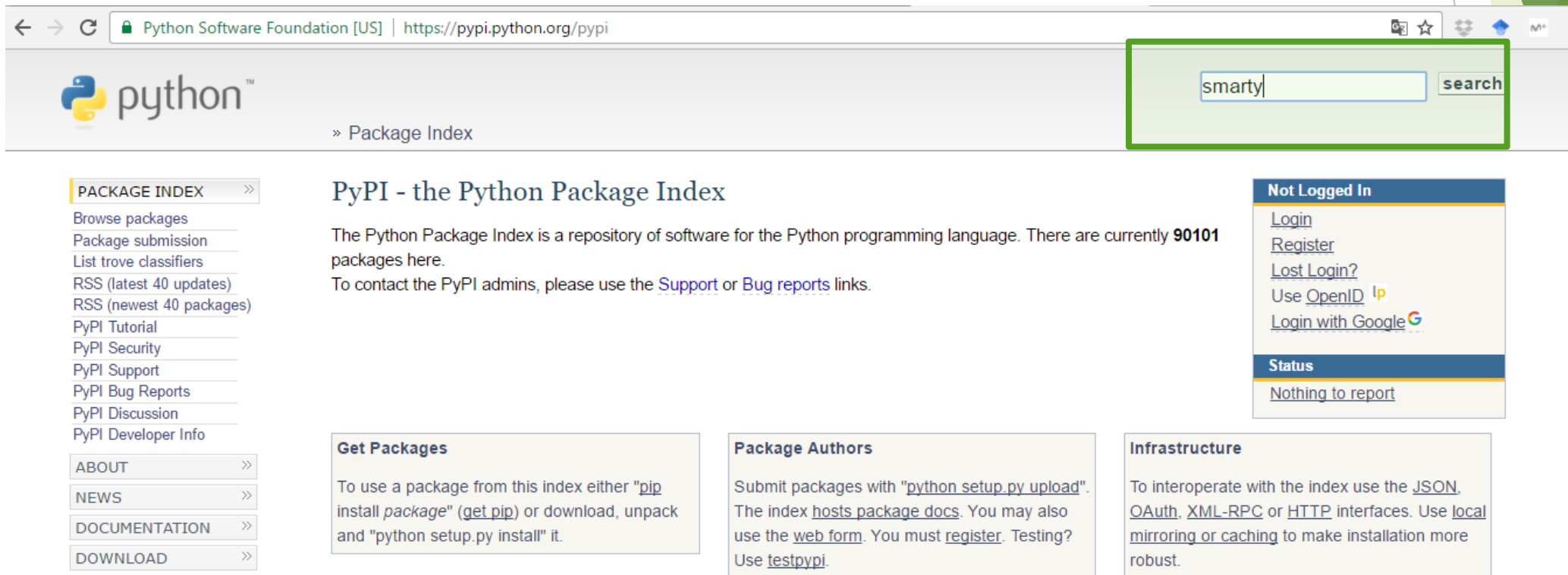
In [3]: import deap
```



# Repositorios de paquetes disponible con Pip

<https://pypi.python.org/pypi>

Utilizar el buscador para identificar paquetes del campo en el que trabajamos



The screenshot shows the PyPI website interface. At the top, the browser address bar displays "Python Software Foundation [US] | https://pypi.python.org/pypi". Below the address bar, the Python logo and "python™" are visible, followed by "» Package Index". A search bar is highlighted with a green box, containing the text "smarty" and a "search" button. On the left side, there is a sidebar with links: "PACKAGE INDEX" (highlighted), "Browse packages", "Package submission", "List trove classifiers", "RSS (latest 40 updates)", "RSS (newest 40 packages)", "PyPI Tutorial", "PyPI Security", "PyPI Support", "PyPI Bug Reports", "PyPI Discussion", and "PyPI Developer Info". Below these are "ABOUT", "NEWS", "DOCUMENTATION", and "DOWNLOAD" links. The main content area features the heading "PyPI - the Python Package Index" and a description: "The Python Package Index is a repository of software for the Python programming language. There are currently 90101 packages here. To contact the PyPI admins, please use the [Support](#) or [Bug reports](#) links." Below this, there are three boxes: "Get Packages" (explaining how to use packages with pip), "Package Authors" (explaining how to submit packages), and "Infrastructure" (explaining interoperability options). On the right side, there is a "Not Logged In" section with links for "Login", "Register", "Lost Login?", "Use OpenID", and "Login with Google", followed by a "Status" section with a link for "Nothing to report".



# Números aleatorios → Módulo random

Módulo para la generación de números aleatorios:

```
import random
```

Más información este módulo:

<https://docs.python.org/2/library/random.html>

à Abrir script14.py



# Manejo de archivos en Python

En Python podemos escribir y leer de archivos de forma muy sencilla

```
fichero = open("nombre_fichero.txt", r) # abrimos el fichero en modo lectura
```

```
fichero = open("nombre_fichero.txt", w) # abrimos el fichero en modo escritura
```

→ Abrir lectura1.py

à Abrir escritura1.py





# Otro detalle de Spyder ...



Spyder (Python 2.7)

Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Herramientas Ver Ayuda

Editor - C:\Users\dany\Dropbox\Curso Python Malaga\Bloque2\_Ejercicios\grafica6.py

```
12 @authors:DanyGR MarioDM IgnacioGP
13 """
14
15 import matplotlib.pyplot as plt
16 import numpy as np
17
18 numeros=np.random.randn(1000)
19 plt.hist(numeros)
20 #plt.hist(numeros,bins=15) #Número de barras que queremos añadir
21 plt.figure()
22 ##Representar muestras de un distribución normal
23
24 y=np.random.normal(10,5,1000)
25 plt.hist(y)
26 plt.xlabel("x")
27 plt.ylabel("y")
28 plt.title("Distribucion normal media=10")
29 plt.figure()
30
31 ##Representar muestras de una distribución exponencial
32
33 y=np.random.exponential(3,1000)
34 plt.hist(y)
35 plt.xlabel("x")
36 plt.ylabel("y")
37 plt.title("Distribucion exponencial")
38 plt.figure()
```

Explorador de archivos

| Name                | Size      | Type      | Date Modified       |
|---------------------|-----------|-----------|---------------------|
| main.py             | 301 bytes | py File   | 09/11/2016 17:45:20 |
| medidas.csv         | 257 bytes | csv File  | 15/09/2016 11:59:50 |
| medidas.xlsx        | 8 KB      | xlsx File | 15/09/2016 11:45:20 |
| multi_procesos1.py  | 548 bytes | py File   | 24/02/2017 11:12:07 |
| multi_procesos1.pyc | 530 bytes | pyc File  | 24/02/2017 0:52:54  |
| multiobjetivo1.py   | 5 KB      | py File   | 06/11/2016 14:29:19 |
| multiobjetivo2.py   | 4 KB      | py File   | 23/11/2016 16:21:27 |
| mus.mp3             | 10,7 MB   | mp3 File  | 02/01/2016 13:06:35 |
| nada.py             | 159 bytes | py File   | 25/10/2016 17:12:33 |
| numba_example.py    | 608 bytes | py File   | 25/01/2017 18:10:01 |
| numeros.txt         | 66 bytes  | txt File  | 09/09/2016 10:55:04 |

Terminal

Inspector de objetos Explorador de variables Explorador de archivos Perfilador (Profiler)

Terminal de IPython

IP: Terminal 1/A



# Clases en Python

## Definición:

Las clases y los objetos son poderosas herramientas de programación. Facilitan la tarea de programar. Una clase es la “clasificación” de un objeto. Tal como “persona” o “imagen.” Un objeto es una instancia particular de una clase. Tal como “María” es una instancia de “Persona.”

Los objetos poseen atributos, tales como el nombre, altura y edad de una persona. Los objetos también poseen métodos. Los métodos definen qué acciones pueden realizar los objetos, tales como correr, saltar o sentarse. Un método es una función que existe dentro de una clase.

Algunas reglas para la construcción de una clase:

- Los atributos deben ir primero, los métodos después.
- El primer parámetro, en cualquier método, debe ser self.
- Las definiciones de métodos deben ir indentadas exactamente una tabulación.



# Clases en Python

## Más detalles sobre las clases en Python:

En Python, las clases tienen una función especial que es llamada en el momento en que una instancia de esa clase es creada. Añadiendo esa función, llamada constructor, el programador puede añadir el código necesario, el cual, automáticamente, será ejecutado cada vez que una instancia de la clase sea creada.



# Clases en Python

*Python es un lenguaje orientado a objetos. En realidad todo en Python es un objeto.*

à **ABRIR** `clase1.py`

Se puede heredar propiedades de otras clases.

**class** `Mi_clase` (`clase_origen1`, `clase_origen2`):

**Docstring:** Sirve para documentar el código.



# Bibliografía



- ▶ Automate the Boring Stuff with Python Practical programming for total beginners. Written by Al Sweigart. Online: <https://automatetheboringstuff.com/>
- ▶ Matplotlib for Python Developers. Build remarkable publication quality plots the easy way. Sandro Tosi. 2009.
- ▶ Python for kids. A Playful Introduction to Programming. A Playful Introduction to Programming. Jason R. Briggs. Jason R. Briggs.
- ▶ <http://www.python-course.eu/course.php><http://www.python-course.eu/course.php>
- ▶ Documentación oficial: <https://docs.python.org/2/contents.html>
- ▶ <http://effbot.org/>

