

# Python: las herramientas básicas

---

Si queremos empezar por el principio, debemos hablar de Python antes que de cualquier otra cosa.

Python se ha convertido en el principal lenguaje de programación usado en áreas tan importantes en la actualidad como son la ciencia de los datos, el aprendizaje de máquina, el análisis de datos, el procesamiento de datos a gran escala, el análisis predictivo, procesamiento de lenguaje natural, matemática simbólica, estadística, y un largo etcétera.

En esta sesión introducimos las herramientas básicas, desde el punto de vista práctico, con las que vamos a trabajar en este curso: Python e IPython, Anaconda, Spyder y el Jupyter Notebook de Python.

## ¿Qué es Python?

Python es un lenguaje de Programación creado por Guido van Rossum en 1991. El nombre de este lenguaje no guarda ninguna relación con la serpiente pitón --aunque luego resulta divertido hacer analogías con este reptil-- sino con el grupo de humoristas británicos, los Monty Python.

Te aconsejo que eches un vistazo a la página oficial de Python:

<https://www.python.org/> (<https://www.python.org/>)

En ella encontrarás unos pequeños ejemplos con los que te formarás una idea rápida de algunas de las características de Python.

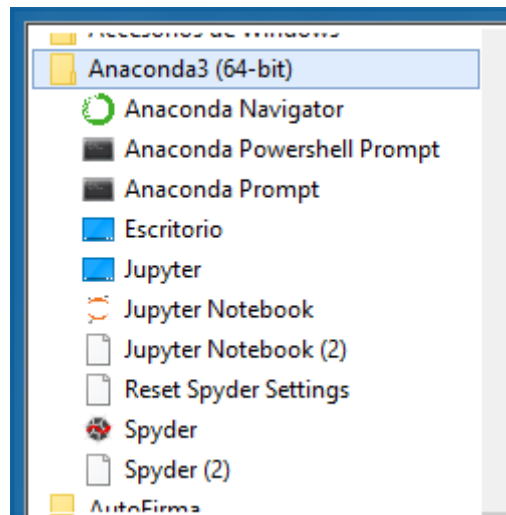
Este lenguaje posee una licencia de código abierto, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1. Nosotros usaremos la versión 3, que no es completamente compatible con las anteriores; esto es un detalle importante porque si se usa código Python de la versión 2, es posible que se produzcan algunos errores.

## Anaconda

Anaconda es una distribución científica de Python gratuita, que incluye varias herramientas que nos facilitarán el trabajo y es perfecta para empezar. Para instalarla, sigue las instrucciones incluidas en la siguiente página:

<https://store.continuum.io/cshop/anaconda/> (<https://store.continuum.io/cshop/anaconda/>)

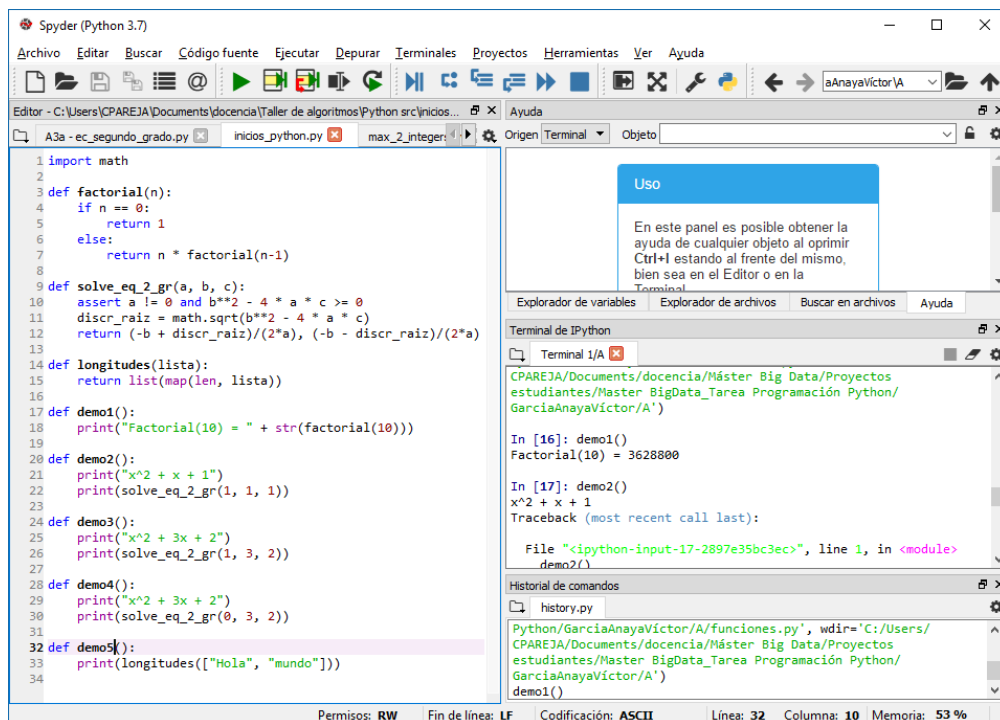
Al instalarla, podemos comprobar que aparecen las siguientes herramientas en nuestro sistema:



## Spyder

Una de las herramientas que proporciona Anaconda es Spyder, un entorno de desarrollo integrado (en inglés, IDE = *Interactive Development Environment*) para el lenguaje Python. Spyder puede instalarse y usarse independientemente de Anaconda. Spyder ofrece facilidades para edición, pruebas interactivas, depuración, etc., e incluye una serie de paquetes típicamente usados para computación científica y ciencia de los datos.

Al arrancar Spyder, se abre una ventana como la siguiente:

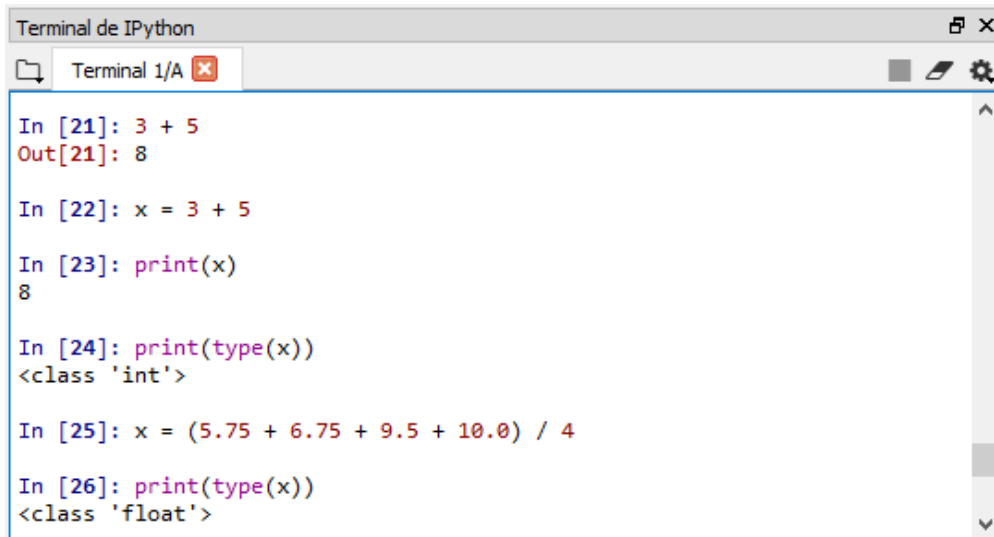


El nombre *SPYDER* proviene de *Scientific PYthon Development EnviRonment*.

Las dos partes más importantes ahora en la ventana de Spyder son el cuadro de la izquierda, en el que se puede ir escribiendo código de Python, y la parte central de la derecha, una terminal donde se pueden ir haciendo pruebas interactivas.

## IPython

IPython es una consola interactiva de Python, que amplía sus funcionalidades con otras, como son el resaltado de líneas y errores mediante colores, autocompletado de variables mediante el tabulador, etc.



```

Terminal de IPython
Terminal 1/A x

In [21]: 3 + 5
Out[21]: 8

In [22]: x = 3 + 5

In [23]: print(x)
8

In [24]: print(type(x))
<class 'int'>

In [25]: x = (5.75 + 6.75 + 9.5 + 10.0) / 4

In [26]: print(type(x))
<class 'float'>

```

Al arrancar Spyder, mi primera recomendación es que te sitúes en la ventana de IPython y pruebes por tí mismo su funcionamiento con las instrucciones que ves en el ejemplo o con otras que tú mismo elijas.

## El Jupyter Notebook de Python

El Jupyter Notebook de Python es una interfaz de Python que añade algunas mejoras, permitiendo combinar código de Python con fragmentos de texto y con imágenes, y donde el texto está enriquecido con tablas, tipos de letra, fórmulas de  $LaTeX$ , como ésta:  $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ , tablas, etc. Por lo tanto, es una herramienta ideal para crear documentos interactivos, y también por eso existen actualmente muchos cursos en que el material se organiza como una colección de archivos de Notebook.

Por ejemplo, este documento es un archivo de Notebook, y tú puedes usarlo para leerlo, para probar el funcionamiento de las instrucciones que contiene y experimentar con ellas o con otras nuevas, puedes cambiar en él lo que quieras y añadir o modificar texto, código o imágenes.

Los archivos manejados por el Notebook de Python se almacenan con la extensión *ipynb*. Este archivo es un ejemplo de ello, aunque también se puede almacenar en forma de *pdf*.

## Modo interactivo

En IPython y en el Notebook de Jupyter se puede trabajar de manera interactiva. La manera más básica de iniciarse es similar al uso de una calculadora:

In [1]:

```
x = 30 + 50
print(x + 23, type(x))
```

103 <class 'int'>

En el modo interactivo, podemos ir escribiendo *expresiones*, como la anterior, y se calculan directamente.

También podemos ir ejecutando *instrucciones*, y se van ejecutando una a una:

Por ejemplo, una *instrucción de asignación*, que sirve para almacenar el resultado de un cálculo en una *variable*, o la *instrucción de escritura*, que escribe en la pantalla el resultado de un cálculo:

In [2]:



```
x = 3 + 5
print(x)
```

8

Es decir, la suma de  $3 + 5$  se almacena en la variable `x`, y luego se escribe el valor de esta variable.

La variable `x` es como un contenedor, una cajita en la que se guarda un valor, y que se usa diciendo su nombre o *identificador*, `x`. En este caso, `x` es una variable entera, es decir, el *tipo de datos* de dicha variable es un entero:

In [3]:



```
print(type(x))
```

&lt;class 'int'&gt;

Pero podemos manejar reales, cadenas de caracteres, listas...

In [4]:



```
x = (5.75 + 6.75 + 9.5 + 10.0) / 4
print(x)
print(type(x))

x = "Python, mi serpiente favorita"
print(x)
print(type(x))

x = ["Cristóbal", 59, 72.0, "91123985"]
print(x)
print(type(x))
```

8.0

&lt;class 'float'&gt;

Python, mi serpiente favorita

&lt;class 'str'&gt;

['Cristóbal', 59, 72.0, '91123985']

&lt;class 'list'&gt;

Observamos que la variable `x` es un entero, luego una cadena de caracteres (llamada un *string*), luego una lista, etc.

A medida que vamos escribiendo secuencias de código más largas, comprendemos la necesidad de usar identificadores adecuados para las variables, y también la necesidad de añadir *comentarios* que aclaren el cometido de las instrucciones:

In [5]:



```
# Cálculo de mi nota media:
nota_media = (5.75 + 6.75 + 9.5 + 10.0) / 4
print(nota_media)
print(type(nota_media))

# Origen del nombre del Lenguaje Python:
frase = "Python es el nombre de un grupo humorístico."
print(frase)
print(type(frase))

# Mis datos: nombre, edad, peso, teléfono:
datos_cris = ["Cristóbal", 59, 72.0, '91123456']
print(datos_cris)
print(type(datos_cris))
```

```
8.0
<class 'float'>
Python es el nombre de un grupo humorístico.
<class 'str'>
['Cristóbal', 59, 72.0, '91123456']
<class 'list'>
```

## Scripts de Python

La ventana de la izquierda de Spyder permite escribir funciones y programas:

In [6]:



```
# Función factorial, versión iterativa:

def factorial(n):
    acum = 1
    for i in range(1, n+1):
        acum = acum * i
    return acum
```

Para usar una función, debemos ejecutar este archivo e ir de nuevo a la ventana de la derecha, donde ya es posible ejecutar la función definida:

In [7]:



```
print(factorial(5))
```

120

## Librerías

Existen miles de librerías de Python para resolver problemas en las áreas más diversas, tales como cálculos matemáticos, numéricos o simbólicos, procesamiento de lenguaje natural, estadística, conexión con bases de datos, redes neuronales, procesamiento de imágenes, gráficos, web scraping, estructuras de datos, automatic learning, y un largo etcétera.

In [8]:



```
# Importación de la librería math:
import math

# y uso de una constante definida en dicha librería:
print(math.cos(math.pi/3))
```

0.5000000000000001

## Conda: instalación de librerías

Una de las herramientas que se instalan con Anaconda es el *Anaconda prompt*, una consola desde la cual se pueden actualizar e instalar librerías para los fines más diversos. He aquí un ejemplo de uso, en el que he abreviado la pantalla de ejecución por claridad:

```
Anaconda Prompt
(base) C:\Users\CPAREJA>conda install nltk
WARNING: The conda.compat module is deprecated and will be removed in a future
release.
Collecting package metadata: done
Solving environment: done

## Package Plan ##

environment location: C:\ProgramData\Anaconda3

added / updated specs:
- nltk
```

## Por dónde continuar ahora

Una vez introducidas las herramientas básicas, toca en primer lugar instalarlas y familiarizarse con ellas.

Toca también prepararse para estudiar un curso de Python desde el principio, al ritmo que necesites, según sea tu experiencia en programación en algún otro lenguaje. Y si eres un principiante en el mundo de la programación, simplemente prepárate para disfrutar sin prisa de un lenguaje muy sencillo, muy bonito, y que es actualmente usado ampliamente en el mundo del análisis de datos, de la ciencia de los datos y de la ingeniería de datos.

## Referencias

El material elaborado sobre lo estudiado en esta pieza es enorme. Ahora y en general, procuro dar una selección de referencias muy reducida, que sea útil y ofrezca una cantidad de información controlada. La escueta selección que he recopilado esta vez se compone de las páginas oficiales de las herramientas introducidas:

- <https://www.python.org/> (<https://www.python.org/>)
- [https://en.wikipedia.org/wiki/Anaconda\\_\(Python\\_distribution\)](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)) ([https://en.wikipedia.org/wiki/Anaconda\\_\(Python\\_distribution\)](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)))
- <https://ipython.org/> (<https://ipython.org/>)
- <https://www.spyder-ide.org/> (<https://www.spyder-ide.org/>)

