

Python. Las herramientas de trabajo

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, el procesamiento de lenguaje natural, la matemática simbólica, la 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/>)

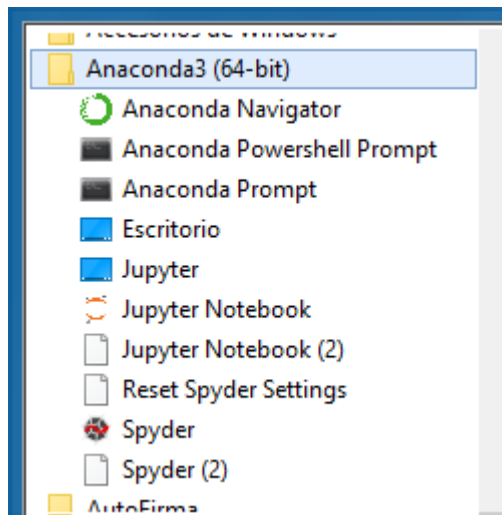
En ella encontrarás unos pequeños ejemplos con los que te formarás una idea rápida de algunas de las características de Python.

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:

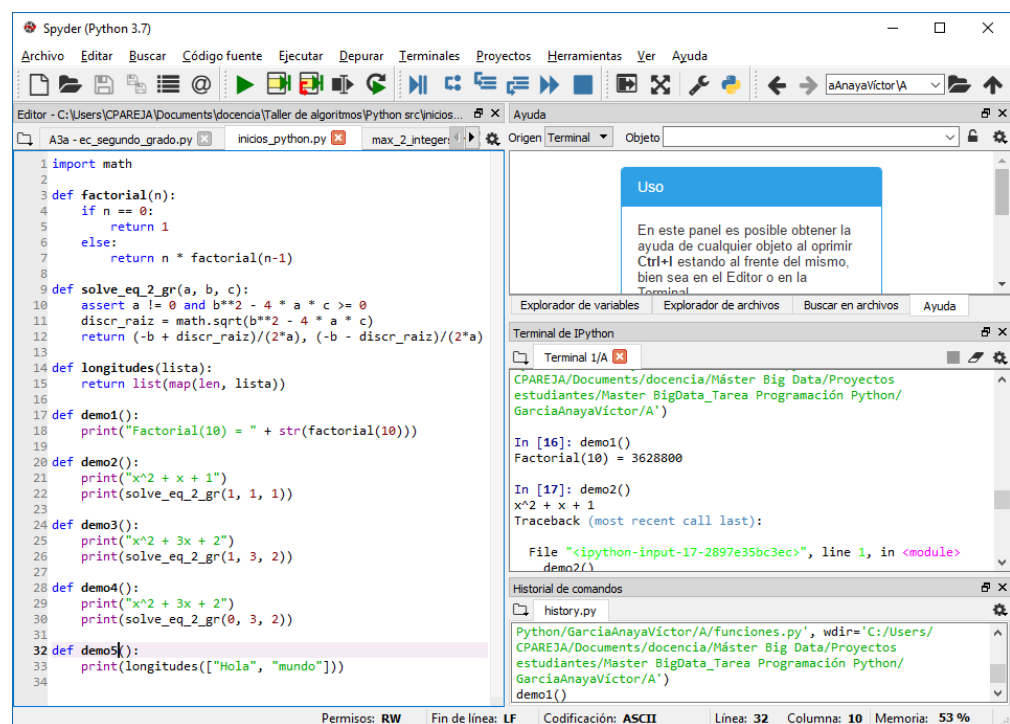
<https://www.anaconda.com/download> (<https://www.anaconda.com/download>)



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:

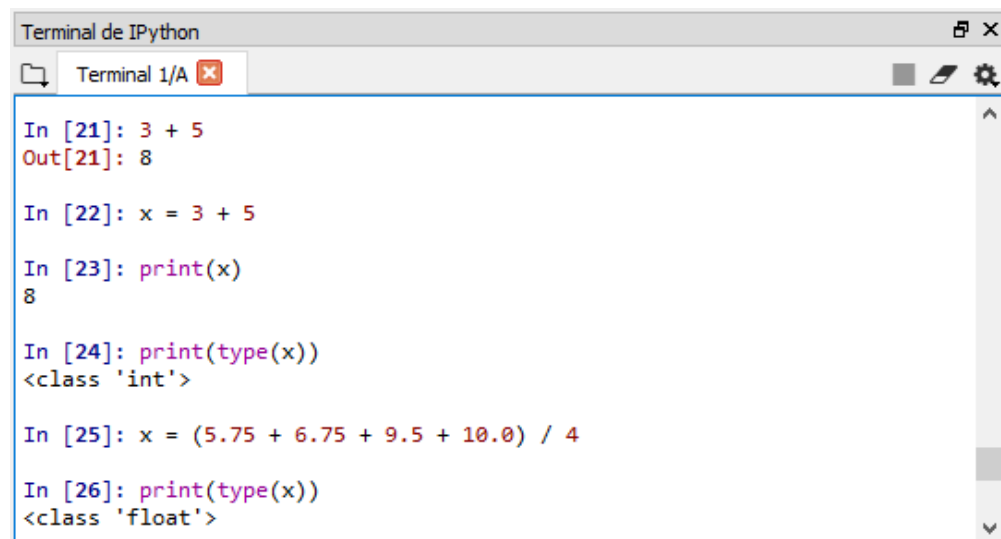


El nombre *SPYDER* proviene de *Scientific PYthon Development EnviRonment*.

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 (*IPython*), una terminal donde se pueden ir haciendo pruebas interactivas. Veámoslo con más detalle en el siguiente apartado.

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

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 ti 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 *L^AT_EX* como ésta:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \text{ tablas, etc.}$$

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. Es corriente, además, desarrollar proyectos profesionales con Jupyter.

Por ejemplo, este documento es un archivo de Notebook, y tú puedes usarlo para leerlo (tanto su código fuente como la imagen legible generada), para probar el funcionamiento de

...

Modo interactivo

Tanto en IPython como en el Notebook de Jupyter, se puede trabajar de manera interactiva. La manera más sencilla de iniciarse es similar al uso de una calculadora:

```
In [1]: ► x = 30 + 50
        print(x + 23)
```

103

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

Como hemos visto, podemos ir ejecutando *instrucciones*, y se van ejecutando una a una, el orden.

Por ejemplo, una *instrucción de asignación*, $x = 30 + 50$, que sirve para almacenar el resultado de un cálculo en una variable o la *instrucción de escritura*, `print(x)`, que muestra el valor de la variable `x` en la pantalla.

```
In [2]: x = 3 + 5
        print(x)

        y = 2**x
        print(x, " -> ", y)

8
8 -> 256
```

Es decir, la suma $3 + 5$ se almacena en la variable `x`, y luego se escribe el valor de esta variable. Luego, esa variable mantiene su valor y se puede utilizar de nuevo...

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))

<class 'int'>
```

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", 2, 72.0, "91123985"]
        print(x)
        print(type(x))

8.0
<class 'float'>
Python, mi serpiente favorita
<class 'str'>
['Cristóbal', 2, 72.0, '91123985']
<class 'list'>
```

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 (`nota_media` mejor que `x`, en este caso) 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, no el de la serpiente"
print(frase)
print(type(frase))

# Mis datos: nombre, núm. de hermanos, peso, teléfono:
datos_cris = ["Cristóbal", 2, 72.0, '91123456']
print(datos_cris)
print(type(datos_cris))

8.0
<class 'float'>
Python es el nombre de un grupo humorístico, no el de la serpiente.
<class 'str'>
['Cristóbal', 2, 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* su definición e ir de nuevo a la ventana de la derecha, donde ya es posible ejecutar la función definida:

```
In [7]: ▶ print(factorial(5))

120
```

In [8]: `print(factorial(500))`

```
12201368259911100687012387854230469262535743428031928421924135883858453
73153881997605496447502203281863013616477148203584163378722078177200480
78520515932928547790757193933060377296085908627042917454788242491272634
43056701732707694610628023104526442188787894657547771498634943677810376
44274033827365397471386477878495438489595537537990423241061271326984327
74571554630997720278101456108118837370953101635632443298702956389662891
16589747695720879269288712817800702651745077684107196243903943225364226
05234945850129918571501248706961568141625359056693423813008856249246891
56412677565448188650659384795177536089400574523894033579847636394490531
30623237490664450488246650759467358620746379251842004593696929810222639
71952597190945217823331756934581508552332820762820023402626907898342451
71200620771464097945611612762914595123722991334016955236385094288559201
87274337951730145863575708283557801587354327688886801203998823847021514
67605445407663535984174430480128938313896881639487469658817504506926365
338175055478128640000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
```

Es frecuente desear que se realicen varias pruebas a la vez:

In [9]: `# 1!, 2!, ..., 5!:`

```
for k in range(1, 5+1):
    print(k, " -> ", factorial(k))

print(".....")

for k in [1, 10, 20, 30]:
    print(k, " -> ", factorial(k))
```

```
1  ->  1
2  ->  2
3  ->  6
4  ->  24
5  ->  120
.....
1  ->  1
10 ->  3628800
20 ->  2432902008176640000
30 ->  265252859812191058636308480000000
```

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.

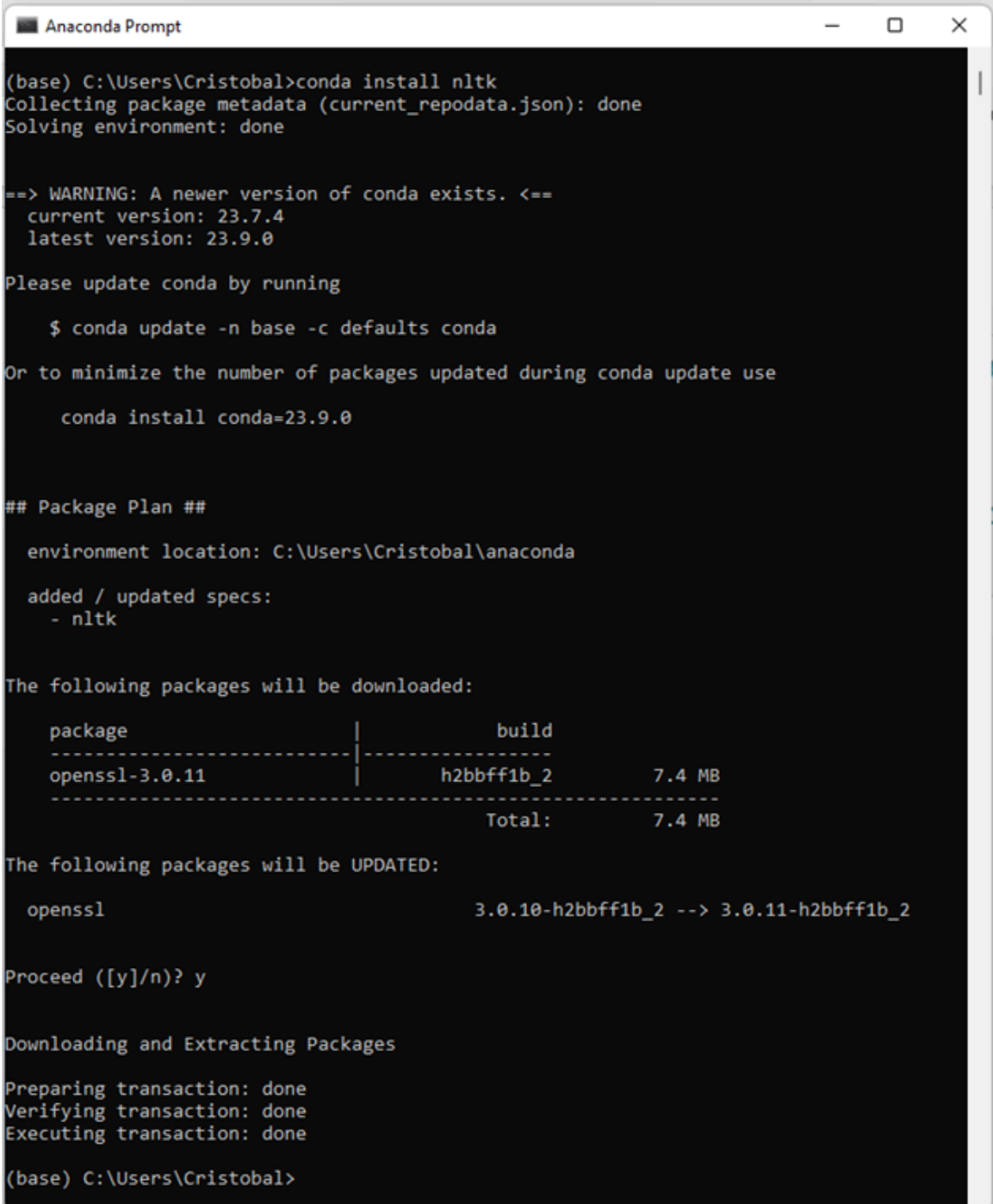
```
In [10]: ▶ # Importación de La librería math:
import math

# 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\Cristobal>conda install nltk
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 23.9.0

Please update conda by running

  $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

  conda install conda=23.9.0

## Package Plan ##

  environment location: C:\Users\Cristobal\anaconda

  added / updated specs:
    - nltk

The following packages will be downloaded:



| package        | build      | size   |
|----------------|------------|--------|
| openssl-3.0.11 | h2bbff1b_2 | 7.4 MB |
| Total:         |            | 7.4 MB |



The following packages will be UPDATED:

  openssl 3.0.10-h2bbff1b_2 --> 3.0.11-h2bbff1b_2

Proceed ([y]/n)? y

Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\Users\Cristobal>
```

Por dónde continuar ahora

Una vez introducidas las herramientas básicas, ha llegado el momento de instalarlas y de familiarizarse con ellas.

También ha llegado la hora de 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/>)