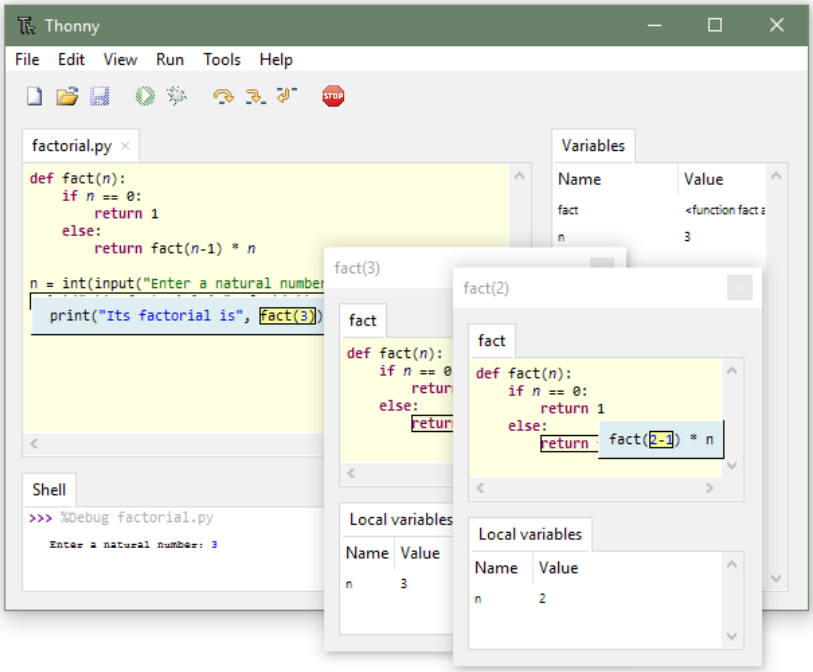
Capítulo 0

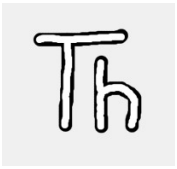
Instalación
Entorno Thonny
Generalidades



Python es un lenguaje de programación interpretado que por defecto ya se encuentra instalado en muchos sistemas operativos. De todas formas, se trata de un lenguaje desarrollado bajo la filosofía de software libre y está disponible de manera gratuita para todos los sistemas operativos. (http://www.python.org)

Para escribir programa en Python, sólo necesitamos un editor de textos (Bloc de Notas, gEdit, etc...) y el intérprete que será el que haga que las instrucciones escritas en el editor de textos se ejecuten.

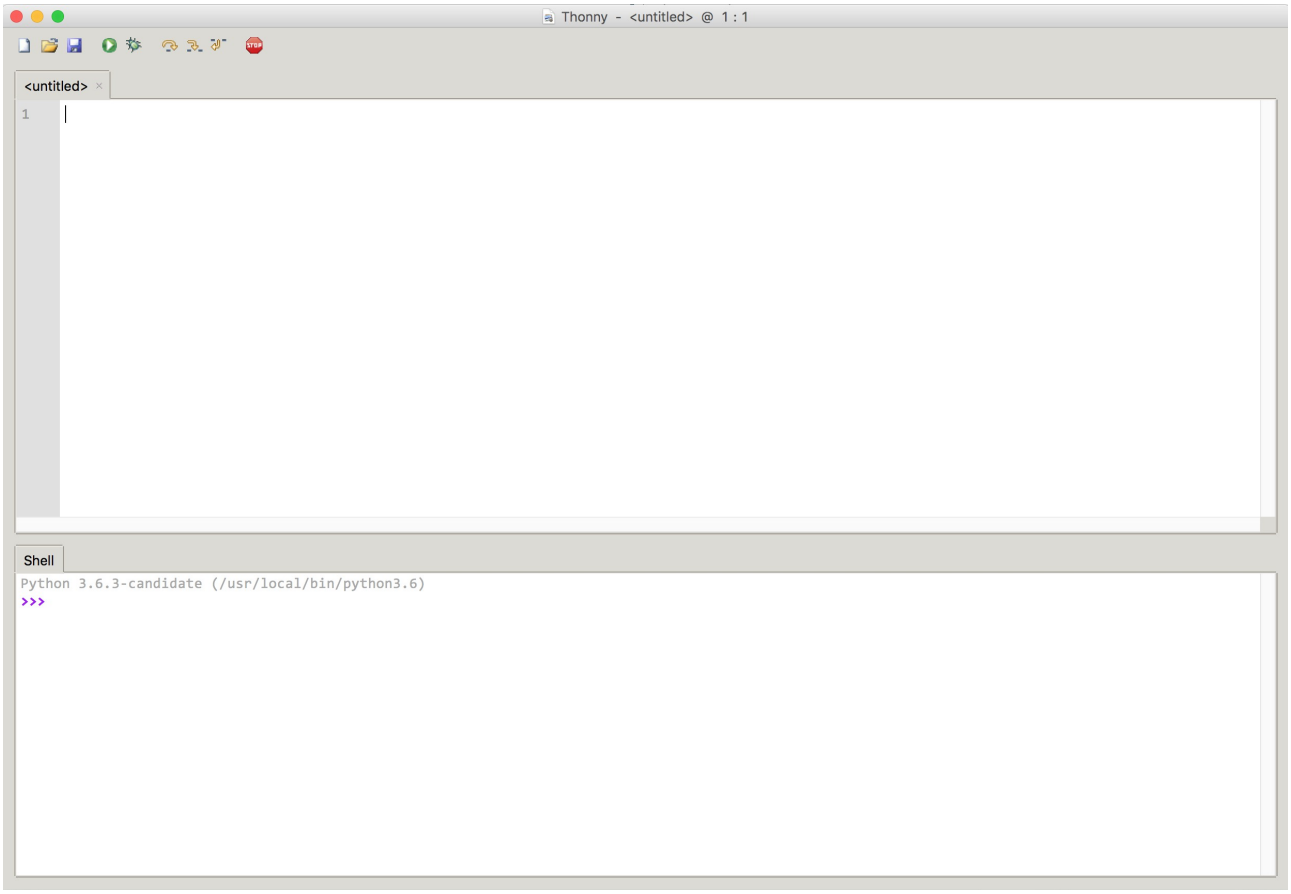
Hay muchos (programas) entornos de desarrollo integrados (IDE's) de gran calidad y con muchísimas posibilidades para desarrollar aplicaciones en Python de manera profesional. Sin embargo, estos entornos tienen tantas características y opciones, que suelen abrumar a las personas se están iniciando en el mundo de la programación y resultan más bien una barrera, si de lo que se trata es de dar los primeros pasos en programación.



Por lo expuesto en los párrafos anteriores, propongo usar el entorno de programación **Thonny**, el cual está diseñado y pensado precisamente para las personas que desean iniciarse en el mundo de la programación en Python. En mi opinión es un entorno ideal para el alumnado de Secundaria y Bachilleratos.

Sin embargo, todo lo recogido en este libro se puede llevar a cabo con cualquiera de los restantes IDE's existentes en la actualidad para Python.

El IDE Thonny, al igual que Python, está desarrollado bajo la filosofía de software libre y se encuentra disponible para los principales sistemas operativos. Podemos obtenerlo desde la web http://www.thonny.org, la instalación es muy fácil y se encuentra bien explicada en la página de descarga del programa. **Thonny ya trae Python incluido en su instalación.**



El programa inicialmente divide la ventana en 2 partes bien diferenciadas, aunque esta configuración se puede modificar mediante el menú View y acomodar al gusto de cada usuario. De todas formas, para el aprendizaje recomiendo que se respeten únicamente las 2 regiones que se muestran en la imagen anterior, puesto que facilita la comprensión de la programación por parte del alumno.

El editor de código

La región superior de la ventana, se trata de un editor de texto como el Bloc de Notas, gEdit, o cualquier otro al que estemos acostumbrados a utilizar, es decir, un programa que guarda el contenido como texto plano, sin ningún tipo de formato.

Este editor al estar pensado para escribir texto en Python, tiene además la particularidad de que nos resaltará la sintaxis y nos ayudará con la escritura de instrucciones facilitándonos las propiedades y métodos disponibles en cada situación y auto completando nombres de funciones y variables. Para ello será de gran ayuda el uso de la tecla del **Tabulador** o de la combinación de teclas **Ctrl + Espacio**.

El editor de código nos permitirá guardar el trabajo en archivos que deben tener la extensión **.py**, de esta forma en cualquier momento podemos recuperar dicho archivo y continuar con el trabajo en el punto donde lo hayamos dejado.

En el editor de texto, una vez que tengamos nuestro código elaborado y listo para su ejecución, sólo es necesario pulsar el botón Play (o pulsar F5) y nuestro programa se ejecutará sin más complicaciones.

Es importante destacar que también tenemos la posibilidad de ejecutar el programa paso a paso e ir comprobando en cada momento el resultado de cada una de las operaciones que se están llevando a cabo.

El IDE Thonny tiene por defecto un diseño simple que ayuda a centrar la atención en la escritura del código y en los resultados que se obtienen de su ejecución, descarga al alumno de infinidad de opciones que son muy útiles para programadores experimentados, pero que distraen y dificultan el aprendizaje de los alumnos que se están iniciando en el mundo de la programación.

El intérprete de Python

La región inferior de la ventana, muestra lo que en Python se denomina el intérprete. Se trata de una línea de comando donde podemos introducir instrucciones y comprobar el resultado que produce su ejecución.

Este intérprete es similar al que podemos obtener desde la línea de comandos de nuestro sistema operativo al ejecutar el comando python (c:\>python, usuario$ python, ...)

El intérprete es ideal para hacer pruebas de pequeñas porciones de código y para aprender el funcionamiento de determinadas instrucciones.

En el siguiente capítulo de este libro, comenzaremos a utilizar este intérprete para familiarizarnos con aspectos básicos de Python.

El intérprete de Python dispone de una ayuda que podemos consultar para conoce la sintaxis de cualquier instrucción. Se trata de un pequeño programa interactivo que podemos ejecutar con las función help()

```
>>> help()

help>
```

Observa que al ejecutar el función help(), se obtiene un nuevo prompt en el intérprete.

El aspecto de la ayuda es: help>

En el nuevo prompt podemos escribir cualquiera de las palabras reservadas de Python y obtener la ayuda que está disponible:

```
help> for
```

```
help> for
The "for" statement
*******************

The "for" statement is used to iterate over the elements of a sequence
(such as a string, tuple or list) or other iterable object:

   for_stmt ::= "for" target_list "in" expression_list ":" suite
                ["else" ":" suite]

The expression list is evaluated once; it should yield an iterable
object.  An iterator is created for the result of the
"expression_list".  The suite is then executed once for each item
provided by the iterator, in the order returned by the iterator.  Each
item in turn is assigned to the target list using the standard rules
for assignments (see Assignment statements), and then the suite is
executed.  When the items are exhausted (which is immediately when the
sequence is empty or an iterator raises a "StopIteration" exception),
the suite in the "else" clause, if present, is executed, and the loop
terminates.

A "break" statement executed in the first suite terminates the loop
without executing the "else" clause's suite.  A "continue" statement
executed in the first suite skips the rest of the suite and continues
with the next item, or with the "else" clause if there is no next
item.

The for-loop makes assignments to the variables(s) in the target list.
This overwrites all previous assignments to those variables including
those made in the suite of the for-loop:

   for i in range(10):
       print(i)
       i = 5             # this will not affect the for-loop
                         # because i will be overwritten with the next
                         # index in the range
```

Para salir de modo help>, podemos dar la instrucción **quit** y volver al intérprete:

```
help> quit

>>>
```

Desde el intérprete también podemos obtener ayuda directamente:

>>> help('for')

Palabras reservadas de Python (palabras claves):

and	as	assert	break	class
continue	def	del	elif	else
except	finally	for	from	global
if	import	in	is	lambda
nonlocal	not	or	pass	raise
return	try	while	with	yield
False	None	True		

Generalidades

Para finalizar este primer capítulo introductorio, vamos a exponer algunos aspectos generales sobre Python que resultan de vital importancia para poder escribir código en este lenguaje.

Sensible a mayúsculas y minúsculas, Python es un lenguaje que diferencia entre mayúsculas y minúsculas, esto significa: For es diferente de for, nombre es diferente de Nombre, etc.

Bloques de código. Hay determinadas instrucciones (for, while, id, elif, else, def, try, except) que tienen por objetivo ejecutar un conjunto de instrucciones que están relacionadas entre sí, a estas instrucciones se le llaman bloque de código.

Las instrucciones que definen el comienzo del bloque de código, deben terminar en el carácter de los 2 puntos (**:**).

Indentación, es la forma que tiene Python de realizar bloques de código. Se llama así una especie de sangría antes del texto que se establece para clarificar la escritura del código. Veámoslo mejor con un ejemplo:

```
1     for n in (1,2,3,4,5,6):
2         print(n, n*10, n*100)
3         print('------', n, '------')
4         print()
```

En la línea 1 se ha usado una instrucción que lleva implícita la ejecución de un bloque de código, por ese motivo esa instrucción finaliza en **:**

Las líneas 2, 3, y 4 forman el bloque de código asociado a la instrucción de la línea 1. Estas líneas están asociadas porque están desplazadas unos espacios hacia la derecha con respecto a la línea 1.

Aunque no es obligatorio, se ha acordado que la indentación estándar sea de 4 espacios.

La indentación es una práctica habitual en la mayoría de lenguajes de programación porque mejora la lectura del código, es una buena forma de visualizar las líneas que están relacionadas entre sí y las que están incluidas, unas dentro de otras. Sin embargo, en Python es una regla para escribir el código, es decir, no existen otros caracteres para agrupar código, necesariamente hay que hacerlo a través de la indentación.

Thonny, nos ayudará con la indentación en la escritura del código cada vez que sea coherente con el código que estemos escribiendo.

Comentarios

Un comentario se llaman a un texto incluido dentro del código pero que no tiene ningún efecto sobre la ejecución del código. Un comentario es una nota aclaratoria sobre la funcionalidad del programa, el uso de una variable, o cualquier otra aclaración que se considere necesaria para comprender mejor el código que se está escribiendo.

En Python tenemos 2 formas de escribir comentarios:

Con la almohadilla #, podemos escribir comentarios de una sola línea: # Comentario

Con las triples comillas podemos escribir código de varias líneas. Son válidas las comillas dobles y las comillas simples:

''' Este texto aunque sea de varias líneas contiene un comentario '''