Tema 1:
Python 1

Índice

1. Introducción

Python es un lenguaje de programación de alto nivel creado por Guido van Rossum.

Posee las siguientes características:

- Es interpretado: el código que escribamos se va a ejecutar línea a línea donde el intérprete de Python (el programa que ejecuta el código) convierte el código fuente en bytecode (un conjunto de bytes) el cual se ejecuta por la máquina virtual de Python (Python Virtual Machine o PVM).
- Multiplataforma: existen intérpretes de Python para Windows, Linux, macOS y muchas otras plataformas. Esto permite que un código escrito en Python se ejecute en cualquier plataforma que cuente con un intérprete.
- Es de código abierto (Open source[1]) lo que permite el libre uso y modificación del mismo.

Python se ha hecho muy popular en los últimos años, entre los más reseñables:

- Inteligencia Artificial (IA) y aprendizaje automático (machine learning). TensorFlow, PyTorch o scikit-learn.
- Análisis de datos (Big data). Pandas o Matplotlib.
- Desarrollo web. Django.
- Aplicaciones de escritorio o móvil: Kivy o beeware.
- Automatización de tareas. Selenium o pyautogui.

2. Instalación de Python

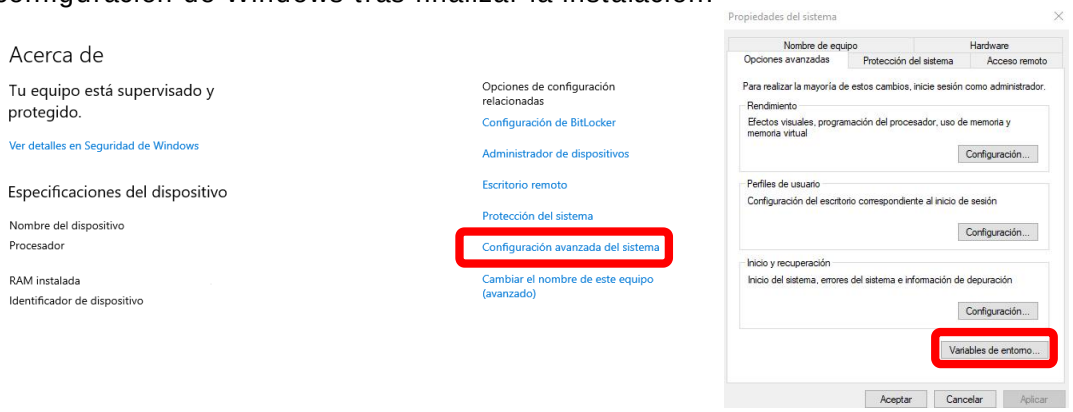
Python viene instalado en sistemas Linux y Mac OS X por defecto, en sistemas operativos Windows es necesario la instalación del intérprete, en este caso utilizaremos la versión 3 de Python.

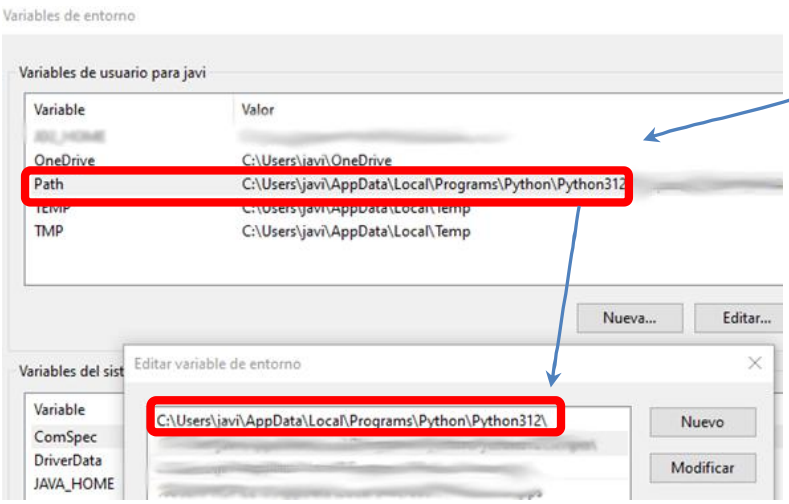
Para descargar la versión de Windows iremos a la página web www.python.org y desde la sección Downloads —> Windows descargaremos la última versión:

[1] https://es.wikipedia.org/wiki/Código_abierto

Una vez descargado el fichero se ejecuta para comenzar con la instalación:

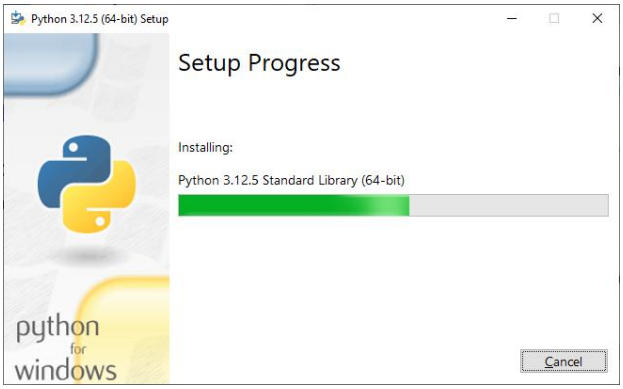
1. Añade la ruta del intérprete al path. De esta forma se puede ejecutar el intérprete directamente una consola. Esta acción se puede también desde las herramientas de configuración de Windows tras finalizar la instalación.





2. Necesario para instalar Python para todos los usuarios, no lo usaremos

3. Personalizar la instalación de Python escogiendo que elementos queremos instalar. Dejaremos sus valores por defecto.



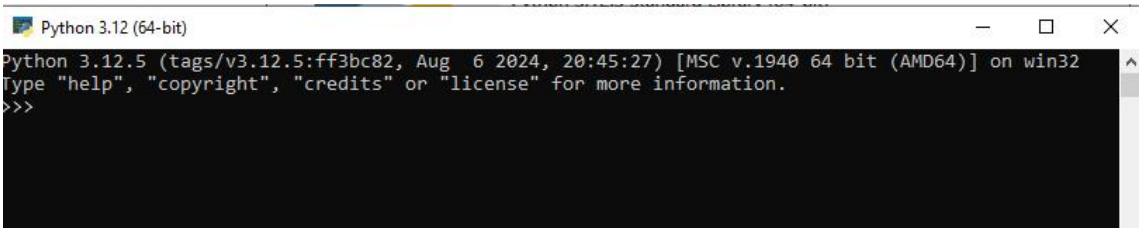
4. Comenzamos con la instalación

Una vez finalizada la instalación vamos a ejecutar el intérprete de Python pare crear nuestro primer programa en Python: el tradicional “Hola Mundo”.

Para ello desde el símbolo de sistema ejecutamos: Python o directamente ejecutamos el interprete de Python desde el menú de inicio:

:

Se mostrará la siguiente ventana:



Donde los >>> indica que el interprete esta a la espera de un comando.

Ahora escribimos la siguiente instrucción: print(“Hola mundo”) y presionamos enter.

Si la instalación ha sido correcta se debe visualizar el mensaje: Hola mundo.

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hola mundo")
Hola mundo
>>>
```

2.1. Entornos de desarrollo IDE

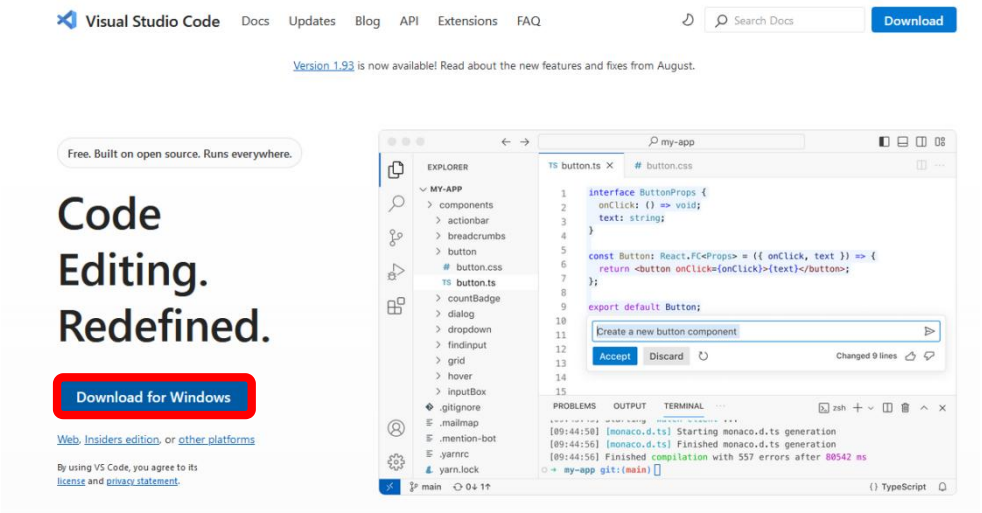
Aunque para crear un programa se puede usar estos se hacen poco usables cuando el programa crece.

Para escribir un programa se utiliza un IDE (entorno de desarrollo integrado). Los IDEs más usados para desarrollar en Python son:

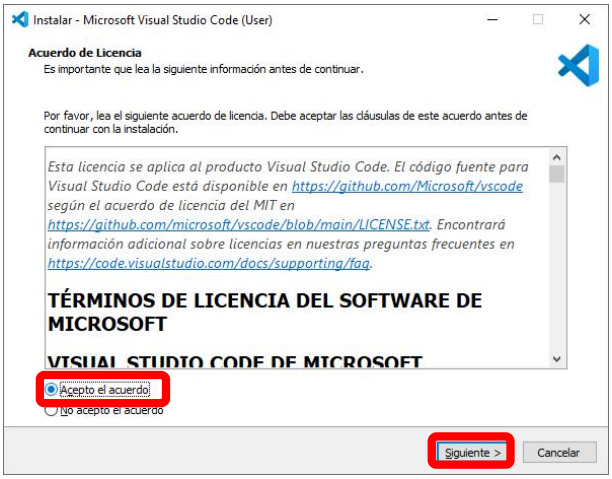
- Visual Studio Code
- PyCharm

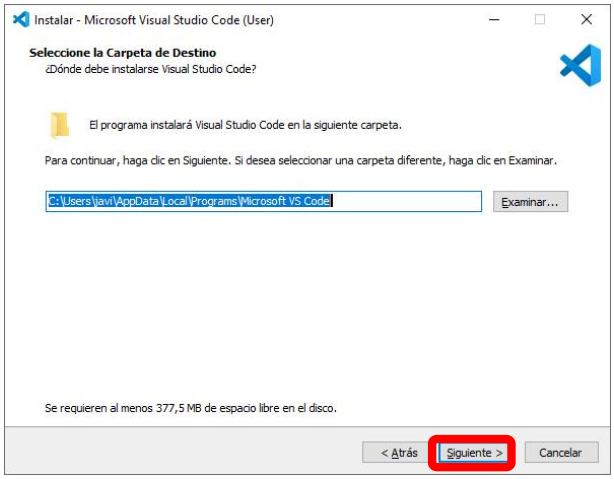
Mientras PyCharm es específico para Python Visual Studio Code tiene soporte para muchos lenguajes de programación. Este es el IDE que usaremos para codificar nuestros programas.

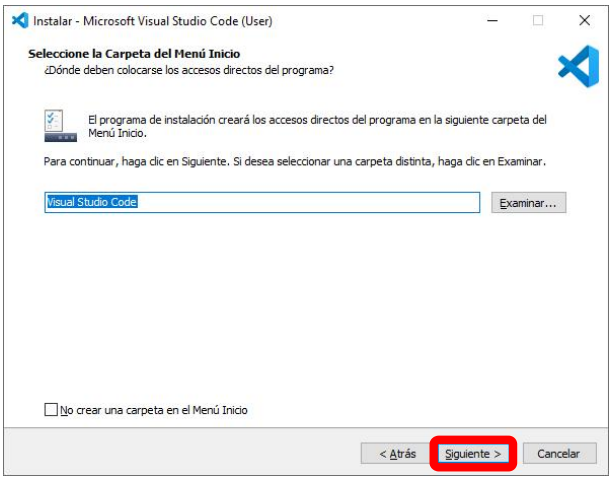
Para ello descargamos el instalador de su página web: https://code.visualstudio.com/

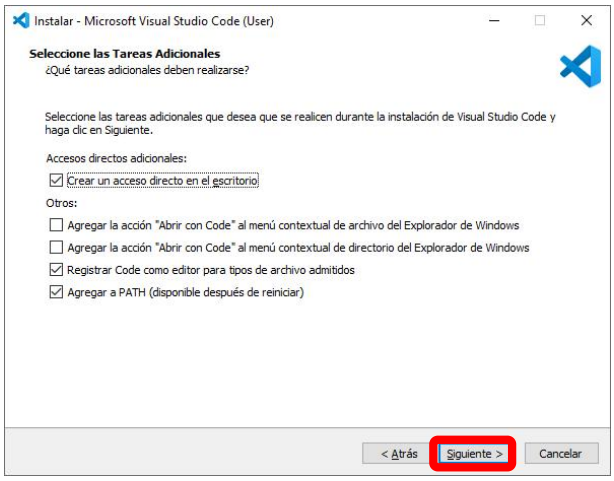


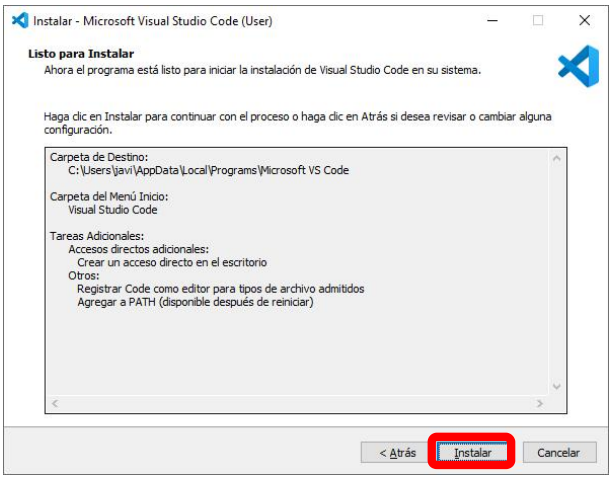
Una vez descargado ejecutamos el instalador:

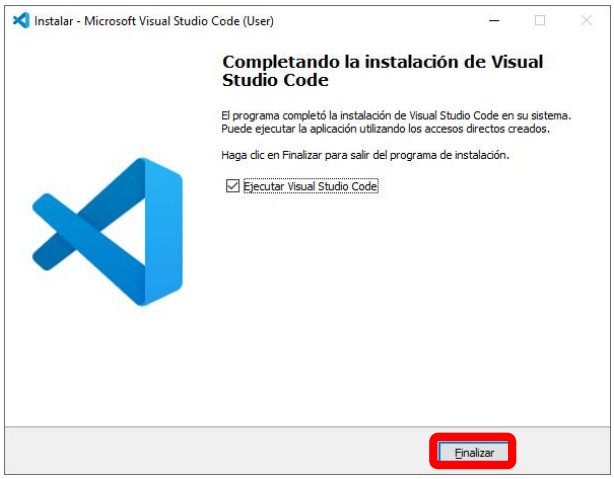




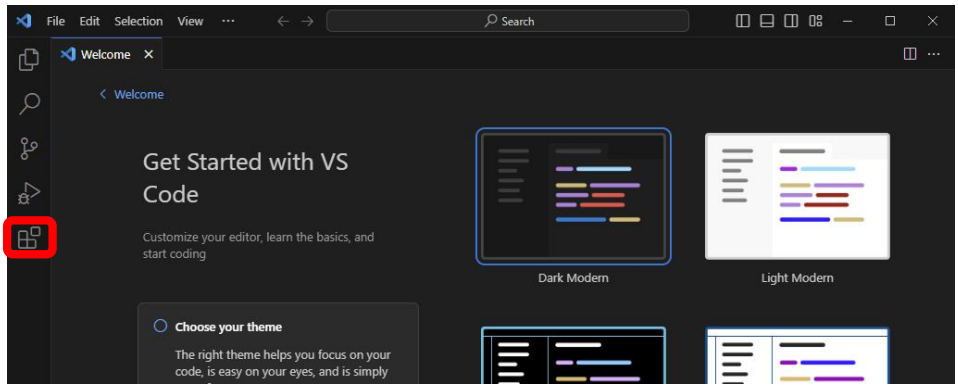




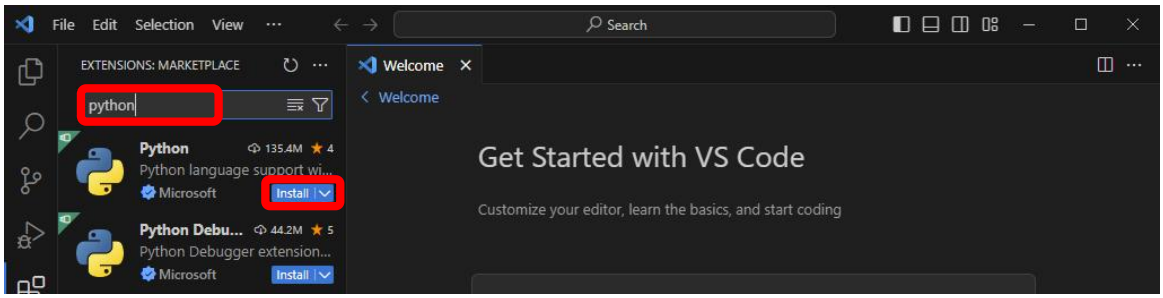




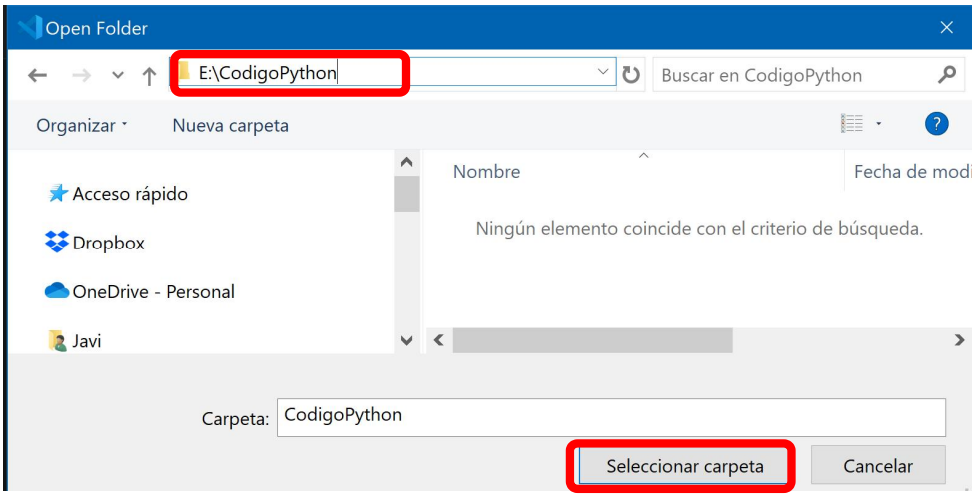
Una vez terminada la instalación ejecutamos Visual Studio Code y procedemos a instalar las extensiones que dan soporte a Python.



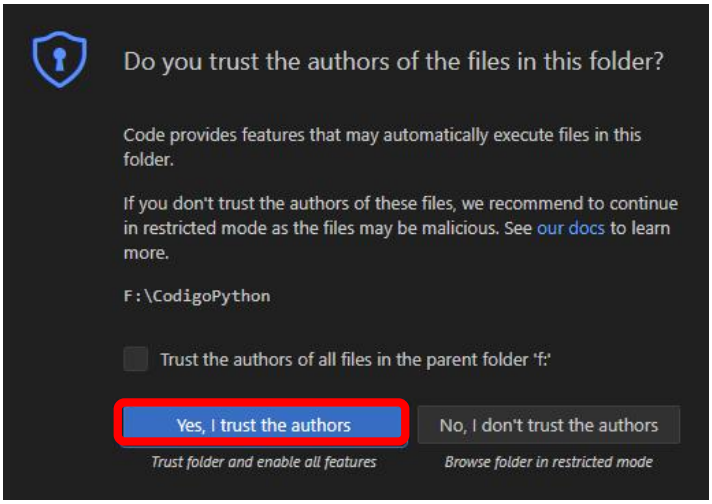
Tenemos que buscar e instalar la extensión de Python:



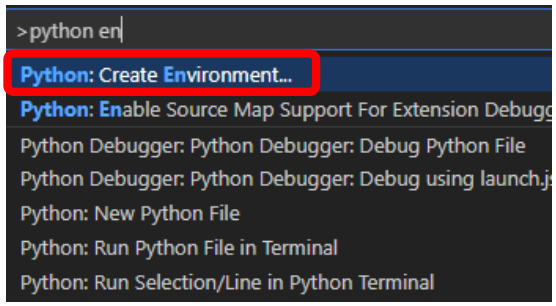
Procedemos ahora a crear un proyecto Python. Para ello creamos una carpeta con File —> Open Folder o Ctrl+k Ctrl+o. Donde deberemos escoger, o crear, la carpeta en la que estarán los ficheros de nuestro proyecto Python.



Y pulsamos en el botón que pone que confiamos en los autores.

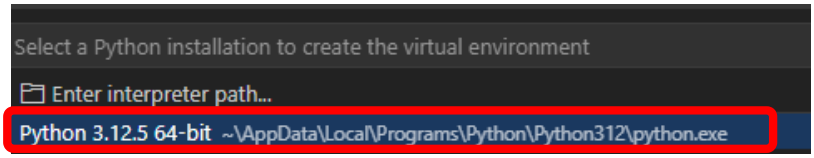


Ahora realizaremos la última configuración . En la paleta de comandos View —> Command Palette o Ctrl+Shift+p buscamos el siguiente comando:



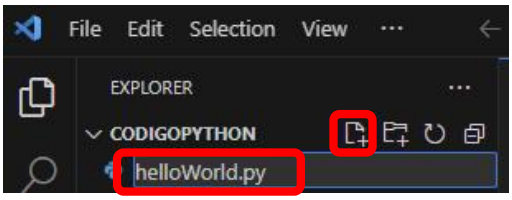
Escogemos Venv:

Y escogemos un intérprete de Python. En el ejemplo escogemos el intérprete de Python que instalamos en el punto anterior.



Ahora podemos crear el fichero de código que contendrá nuestro primer programa Python.

Para ello pulsamos en File —> New File o en el siguiente icono del explorer estableciendo su nombre a helloWorld.py.



Y en el escribimos el siguiente programa:

```
print("Hola mundo")
```

Y lo ejecutamos con el botón de ejecutar fichero o Ctrl+F5

Donde debería verse la siguiente salida en la terminal:

```
Hola mundo
```

3. Conceptos básicos de programación en Python

En este punto vamos a ver una serie de conceptos que utilizaremos a lo largo del curso:

- En case sensitive. Esto es diferencia mayúsculas de minúsculas. Por ejemplo print no es igual a Print.
- Algoritmo: es un conjunto de instrucciones ordenadas, no ambiguas, que permite resolver un programa.
- Dato: información, de cualquier tipo, que maneja que maneja un algoritmo.
- Tipo de dato: es el tipo de que puede tener un dato. Entre otros tenemos enteros, números con decimales, cadenas, … se verá en un punto posterior.
- Literal: es un valor fijo, de un tipo, que se utiliza en el código. Su valor no varía.
  - Entero: 1, 5, 10, ...
  - Cadena de texto: "Rojo", 'Verde', ...
- Variable: es un espacio en menoría que permite almacenar un dato para uso posterior. Pueden cambiar su valor. Una variable tiene un nombre y es de un tipo.
- Expresiones: son una combinación de literales, operadores, variables, paréntesis, funciones, … por ejemplo: (num*4)+3
- Cada sentencia ocupa una línea y se ejecutan en orden: de arriba hacia abajo. Si queremos insertar más de una sentencia en una línea estas se han de separar un ;

3.1. Comentarios

Los comentarios se utilizan para documentar nuestros programas y son ignorados durante la ejecución. Comienzan por # y afectan desde su posición hasta el final de la línea.

Ejemplos de comentarios:

```
5 + 2 # Esto es una suma, el interprete de Python puede ejecutar cálculos
# Este comentario afecta a toda la línea
```

3.2. Tipos de datos

Los tipos de datos básicos que posee Python son los siguientes:

- Números Enteros (int): son número sin parte decimal. Por ejemplo 45.

  Los números enteros se pueden representar en notación decimal, octal (anteponiendo un “0o” al valor) o hexadecimal (anteponiendo “0x” al valor).

- Números Reales o de coma flotante (float): son número que pueden tener parte decimal. De tipo. Por ejemplo 12.5
- Cadenas de texto o string (str): son un conjunto de caracteres. Se representan tanto entre comillas simples como dobles ("hola" y 'hola' son equivalentes). La utilización de las comillas ha de ser consistente si se empiezan con comillas dobles se tiene que terminar también con comillas dobles y lo mismo para las comillas simples … es decir no se pueden mezclar para especificar una misma cadena.
- Booleanos (bool): tienen dos posibles valores: True o False. Es una subclase de int, de forma que True tiene un valor de 1 y False de 0.
- None: se utiliza para representar la ausencia de valor.

3.3. Funciones

Python dispone de una serie de funciones predefinidas que se pueden usar en un programa y que se irán viendo a lo largo del curso. Las funciones permiten ejecutar una acción y pueden producir un resultado. Más adelante se verá cómo escribir nuestras propias funciones.

Una función está compuesta de un nombre y una lista de parámetros entre paréntesis (en caso no de tener parámetros los paréntesis irán vacíos).

Cuando se ejecuta una función:

- Python abandona el flujo normal del programa y salta dentro de la función.
- Ejecuta el código que hay dentro de la función
- Termina la función volviendo al flujo previo a la ejecución de la función.

Inicialmente veremos las tres funciones siguientes:

help

Muestra la ayuda de la función, tipo de dato, … pasado como parámetro.

```
help(input)
```

quit

Permite salir del intérprete de Python

```
quit()
```

type

Permite obtener el tipo de dato del parámetro que se le pasa

```
type(1)
```

```
<class 'int'>
```

```
type(1.5)
```

```
<class 'float'>
```

```
type("Hola")
```

```
<class 'str'>
```

print

La función print toma los parámetros que se la pasan y los visualiza por pantalla.

```
print("Hola")                # Escribe la cadena Hola
print(7+6)                   # Realiza la suma y escribe 13
print()                      # Imprime una línea en blanco
print("uno", "dos", "tres")  # Imprime los tres parámetros que se le pasan
```

Cuando se usan con números decimales se puede omitir el cero cuando es el único dígito antes o después del punto decimal.

```
print(5.,.5)
```

```
5.0 0.5
```

En Python, como en la mayoría de lenguajes de programación, usa el carácter \ para definir caracteres de escape[2]. Esos son caracteres con un significado especial:

Secuencia	Significado
\n	Salto de línea
\t	Tabulación
\r	Vuelve al principio de la línea
\b	Retrocede un carácter

Secuencia	Significado
\'	Comilla simple
\"	Comilla doble
\\	Barra inversa

Por ejemplo si queremos separar un print en varias líneas podemos utilizar \n.

```
print("Hola\nplaneta\n\tazul")
```

```
Hola
planeta
        azul
```

Si queremos que print imprima conservando el formato con el que se escribe se puede usar print con triple coma, ya sea doble o sencilla.

```
print('''Soy una línea
         y otra
      y una tercera''')
```

```
Soy una línea
         y otra
      y una tercera
```

Si queremos sobrescribir parte de una cadena:

```
print("12345\b\ba")
```

```
123a5
```

Print tiene dos parámetros especiales. Estos son opcionales y son independientes entre sí y siempre después de las cadenas a escribir:

- sep: indica la cadena que se escribe como separador entre cada parámetro.
- end: indica la cadena que se escribe después de escribir todos los parámetros.

```
print("uno","dos","tres","cuatro","cinco", sep=" + ", end=' = ?' )
```

```
uno + dos + tres + cuatro + cinco = ?
```

[2] https://es.wikipedia.org/wiki/Caracteres_de_escape

Ejercicios

1. Utiliza la función print para imprimir la cadena "Hola" por la pantalla.
2. Elimina las comillas de la instrucción anterior y ejecútala. ¿Qué de error se produce? ¿Qué significa ese error?.
3. Utiliza la función print para imprimir la cadena "Hola mundo" por la pantalla.
4. Cambia las comillas dobles por comillas simples. ¿Qué sucede?.
5. Elimina las comillas de la instrucción anterior y ejecútala. ¿Qué de error se produce? ¿Qué significa ese error?.
6. Ejecuta la siguiente instrucción: print “hola mundo” ¿Qué sucede y por qué?
7. En una misma línea ejecuta dos instrucciones print con el texto que desees.
8. Ejecuta estas instrucciones print:

```
print("2+3")
print(2+3)
print("2"+"3")
Print("2+3")
print(2"+"3)
```

   ¿Qué muestran?¿Por qué?
9. Muestra la ayuda del comando print
10. Muestra en un único print varias cadenas diferentes separadas por un tabulador. Al terminar se ha de mostrar la cadena fin.
11. Muestra una única cadena que en su interior ha de tener una comilla simple, una comilla doble y una barra invertida. Define la cadena tanto con comillas simples como con comillas dobles.

3.4. Variables

Como se comentó la principio de este punto una variable es un espacio de memoria permite almacenar un valor. Se identifican por un nombre y son de un tipo.

Sus características son:

- Su nombre solo pueden contener letras (mayúsculas o minúsculas), dígitos o el guion bajo _.
- Debe comenzar por una letra o un guión bajo.
- No puede usar una palabra reservada de Python.
- Su valor se asigna con un igual = (también se utiliza para modificarlo). Puede ser tanto un literal, una expresión, el resultado de una función, …
- Por convenio los nombres de variables solo usan minúsculas.
- Python tiene un tipado dinámico, esto es el tipo de una variable puede cambiar tras su creación.

Por ejemplo:

```
edad=20
print(edad)
```
```
20
```
```
edad=edad+10
print(edad)
```
```
30
```
```
print(type(edad))   # devuelve <class 'int'>
```
```
<class 'int'>
```
```
edad="Ahora soy una cadena"
print(edad)
```
```
Ahora soy una cadena
```
```
print(type(edad))   # devuelve <class 'str'>
```
```
<class 'str'>
```
```
edad=0xa
print(edad)         # ¿Por qué imprime 10?
```
```
10
```

3.5. Entrada de datos

La instrucción input permite que el usuario pueda introducir datos desde el teclado. Los datos introducidos son devueltos como una cadena de texto, la cual se puede asignar a una variable. Permite opcionalmente mostrar un mensaje de texto:

```
print('¿Cómo te llamas? ')
nombre =input()
ciudad =input('¿Dónde vives? ')
print(nombre, 'vive en', ciudad)
```

En caso de introducir los datos David y Vigo mostraría el siguiente mensaje:

```
David vive en Vigo
```

3.6. Conversiones

Como input devuelve una cadena de texto no se debe utilizar como un argumento para operaciones aritméticas, antes debe convertirse a un valor numérico (**casting**). Para ello utilizamos las funciones:

- int() toma un argumento (por ejemplo, una cadena) e intenta convertirlo a un valor entero; si no se puede realizar la conversión el programa fallará.
- float() toma un argumento (por ejemplo, una cadena) e intenta convertirlo a un numero en coma flotante; si la conversión falla el programa entero fallará.
- str() toma un argumento y lo convierte a cadena.

```
num_ent=12
num_float=1.3
cad='12'

n_float=float(num_ent)
print(n_float,type(n_float))
```
```
12.0 <class 'float'>
```

```
n_str=str(num_float)
print(n_str,type(n_str))
```
```
1.3 <class 'str'>
```
```
n_ent=int(cad)
print(n_ent,type(n_ent))
```
```
12 <class 'int'>
```

3.7. Operadores aritméticos

Un operador es un símbolo del lenguaje de programación que es capaz de realizar operaciones con los valores.

Cuando los datos y operadores se unen, forman juntos expresiones.

Disponemos de los siguientes operadores aritméticos:

Operador	Operación	Ejemplo	Resultado
+	Suma	a=1+2	a=3
-	Resta	a=2-5	a=-3
*	Multiplicación	a=2*3.0	a=6.0
**	Potencia	a=2**3	a=8
/	División	a=5/2	a=2.5
//	División entera	a=5//2	a=2
%	Módulo	a=5%2	a=1

El orden de prioridad de los operadores es el siguiente. Este orden se puede variar mediante paréntesis.

1. +, - unario
2. **
3. *, /, //, %
4. +, - binario

```
print(4+6*2)
```
```
16
```
```
print((4+6)*2)
```
```
20
```

Donde:

- Cuando ambos operandos son enteros, el resultado es entero también.
- Cuando al menos un operando es de coma flotante, el resultado también es flotante.
- El orden de evaluación de operadores con la misma prioridad de de izquierda a derecha.
- La potencia permite elevar un número a otro.
- El módulo es el resto de la división entera. Muy útil para determinar si un número es par/impar, múltiplo de otro o para generar series numéricas circulares (`01230123012301230123...`)

Operadores abreviados

Las operaciones aritméticas con el siguiente formato:

```
variable= variable operador cantidad
```

Se puede abreviar de la siguiente manera:

```
variable operador = valor;
```

Por ejemplo:

```
contador = 0
contador = contador + 1
```

Es lo mismo que:

```
contador = 0
contador += 1
```

Ejercicios

12. Realiza un programa que pida dos números al usuario y los sume
13. Crea un programa que pida al usuario una base y un exponente y muestre el valor de elevar la base al exponente.
14. Crea un programa que dada una cantidad en euros, devuelve su valor en dólares.
15. Crea un programa que dada una cantidad en dólares, devuelve su valor en euros.
16. Crea un programa que a partir de la hora de comienzo de un evento y su duración (3 valores: hora, minutos, duración en minutos) calcula la hora de finalización del evento. No te preocupes de los errores que pueda introducir el usuario (hora inválida). PISTA: Emplea el operador módulo (`%`)
17. Crea un programa que calcule y muestre el IMC (índice de masa corporal) de un usuario. Para ello debe:
    - Pedir el nombre de un usuario
    - Pedir su peso
    - Pedir su altura (en metros)

    Calcular el IMC según la siguiente fórmula: (peso / altura^2)
18. A partir de tres datos a, b, c calcula el resultado de la siguiente operación: $(a+b+c)^3$