Python

**¿Qué es Python?**

Python es un lenguaje de programación interpretado de tipado dinámico cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma y disponible en varias plataformas.

Dicho de otro modo, Python es:

* **Interpretado**: Se ejecuta sin necesidad de ser procesado por el compilador y se detectan los errores en tiempo de ejecución.
* **Multiparadigma**: Soporta programación funcional, programación imperativa y programación orientada a objetos.
* **Tipado dinámico**: Las variables se comprueban en tiempo de ejecución.
* **Multiplataforma**: disponible para plataformas de Windows, Linux o MAC.
* **Gratuito**: No dispone de licencia para programar.

**¿Por qué Python?**

* Usado por grandes compañías
* Fácil de aprender y mantener
* Libre open source
* Una gran cantidad de librerías
* Una gran comunidad

Al hacer uso de una sintaxis legible, la curva de aprendizaje es muy rápida, siendo de este modo, uno de los mejores lenguajes para iniciarse en la programación en modo texto.

Python contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero.

El intérprete de Python y la extensa librería estándar se encuentran disponibles libremente en código fuente y de forma binaria para la mayoría de las plataformas desde la Web de Python, <https://www.python.org/>, y se pueden distribuir libremente. El mismo sitio también contiene distribuciones y referencias a muchos módulos libres de Python de terceros, programas, herramientas y documentación adicional.

Para una descripción de los objetos estándar y de los módulos, ver La biblioteca estándar de Python. Referencia del Lenguaje Python dónde se ofrece una definición más formal del lenguaje.

**¿Cómo programar en Python?**

Si queremos empezar a programar en Python, tenemos varias alternativas de cómo empezar a hacerlo:

* podemos invocar al **intérprete**, que se instala cuando instalamos Python. El intérprete funciona de manera similar al shell de Unix: cuando se le llama con una entrada estándar conectada a un terminal, lee y ejecuta comandos de manera interactiva; cuando se le llama con un argumento de nombre de archivo o con un archivo como entrada estándar, lee y ejecuta un script desde ese archivo.
* podemos usar Python sin ningún tipo de instalación. Sin duda la más sencilla y rápida. Para ello recurrimos a los **entornos con versión online**, por ejemplo Google Colab, JupyterLab, etc.
* podemos programar en Python con un **IDE** como por ejemplo Spyder, para lo que tendremos que instalar el propio lenguaje Python y el entorno de desarrollo Spyder.

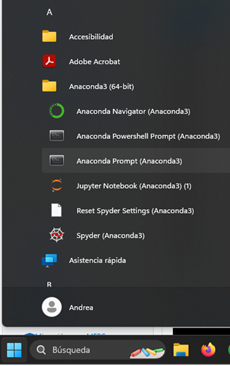
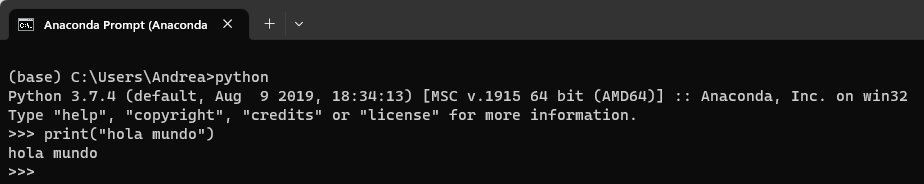
**Primer programa**

En programación, el primer programa que se crea es el que imprime el texto "Hola Mundo" en un dispositivo de visualización, terminal o shell.

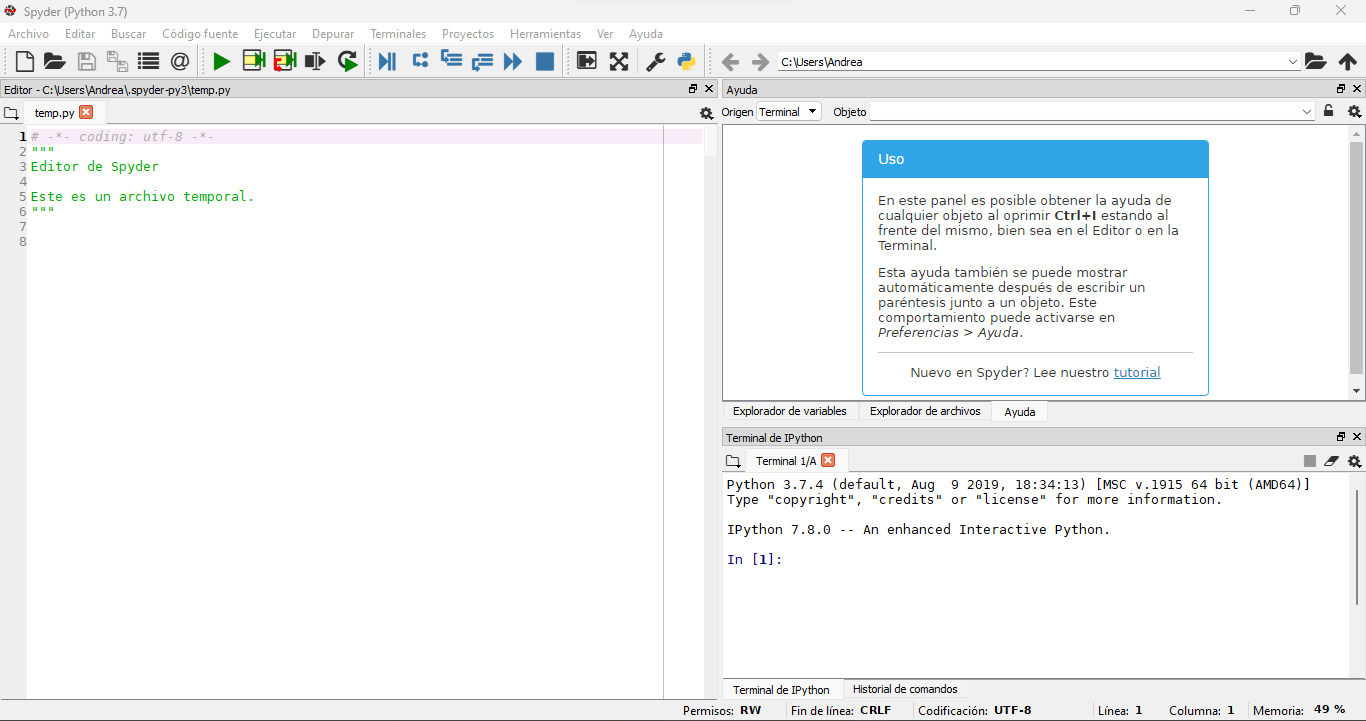
Si queremos escribir **usando el intérprete** debemos convocarlo primero y luego escribir las sentencias usando la sintaxis del lenguaje.

En máquinas con Linux en las que haya instalado Python, desde la terminal se escribe: ***python3***

En máquinas con Windows en las que haya instalado Python, se convoca al prompt de Anaconda y se escribe ***python***.



si queremos escribir el programa **usando el IDE** Spyder por ejemplo



En el curso vamos a trabajar usando el **IDE Spyder y los cuadernos de Google colab** para probar código y realizar ejercicios.

Repaso de contenidos

Operadores

**Operadores aritméticos**

Python es un lenguaje de programación que soporta el paradigma de la programación funcional. Esto quiere decir que está basada en una programación declarativa de funciones matemáticas.

La mayoría de los lenguajes de programación utilizan los mismos operadores aritméticos.

Al hacer operaciones con números enteros y decimales el resultado es siempre decimal, por ejemplo, en el caso de dividir 6 entre 2. Por otro lado, en el caso de que el resultado no tenga parte decimal, Python escribe 0 como parte decimal para indicar que el resultado es un número decimal.

Los operadores matemáticos son:

Suma +

Resta -

División /

Multiplicación \*

Potencia \*\*

Cociente //

**Operadores relacionales**

Los operadores relacionales son utilizados en la mayoría de las ocasiones en las estructuras condicionales.

Los resultados que devuelven las operaciones son de tipo booleano, es decir, verdadero (True) o falso (False).

Igualdad ==

Distinto !=

Mayor que >

Mayor o igual que >=

Menor que <

Menor o igual que <=

**Operadores Lógicos** En ocasiones nos vemos en la necesidad de comprobar varias igualdades, por ejemplo, cuando queremos calcular si un número está comprendido entre un rango de valores.

Las expresiones compuestas con operadores lógicos son utilizadas en las condiciones, bucles, etc.

Se evalúa la condición y a continuación la expresión lógica.

Negación not

Y and

O or

Tipos de datos

En programación un tipo de dato es la forma de representación del dato e indica la clase de dato que se va a manejar.

Esto incluye imponer restricciones en los datos, como qué valores pueden tomar y qué operaciones se pueden realizar.

Python es un lenguaje de programación de propósito general y por ello contiene una gran cantidad de tipos de datos con los que se puede programar.

* Numéricos: En Python los números pueden ser enteros, flotantes o complejos.
* Booleanos: Los tipos de datos booleanos se utilizan para representar verdadero y falso, mediante las palabras reservadas True o False respectivamente.
* Cadenas: Una cadena debe delimitarse con comillas simples o dobles y admite caracteres de escape.
* Listas: Una lista es una colección de objetos: números enteros, flotantes, complejos, cadenas, etc. Una lista se delimita utilizando [] y sus elementos han de separarse con comas. Es posible acceder a sus elementos indicando el índice del elemento deseado.
* Tuplas: En muchos sentidos una tupla es como una lista, contienen una colección de objetos de distinto tipo. Es decir, son lista de elementos de diferente tipo de dato.
* Diccionario: Un diccionario son un tipo de estructuras de datos que permite guardar un conjunto no ordenado de pares clave-valor, siendo las claves únicas dentro de un mismo diccionario, se compone de dos partes: una llave (palabra) y un valor (definición). Las llaves siempre deben ser un tipo de dato primitivo. La llave y el valor se separan con: y sus elementos con comas.

[ ]

nro = 23  
real = 4.56  
bol = True  
cad = "hola mundo"  
lista = [2, 5.8 , "pepe", False]  
dic = {1:'uno', 2:'dos'}

Variables

**¿Qué es una variable?**

Se define como variable al espacio reservado de la memoria que almacena un dato, que como su propio nombre indica, puede cambiar de valor en tiempo de ejecución.

En Python tenemos tipos de datos simples (números enteros, de coma flotante) cadenas de texto, y valores booleanos. Todos estos pueden almacenarse en la misma variable sin especificar de antemano el tipo de datos que almacenará, de ahí a que Python sea un lenguaje de programación de tipado dinámico.

Las variables se comprueban en tiempo de ejecución.

[ ]

var1 = 3  
var2 = "azul"  
print (var1\*2)  
print (var2\*var1)

6

azulazulazul

Instrucciones

**Ingreso de datos por teclado**

En la mayoría de los programas necesitamos interactuar con el usuario para solicitarle diferentes datos.

Debemos saber qué es lo que se le solicita al usuario para saber qué es lo que nos tiene que proporcionar, por lo tanto, al dato introducido deberemos realizar un casting acorde con el dato esperado.

Al procedimiento de transformar una variable primitiva a otro tipo se le conoce como casting

[ ]

nombre = input ("ingrese su nombre: ")  
edad = int(input ("ingrese su edad: "))  
temp =float(input("ingrese la temperatura: "))

ingrese su nombre: juan

ingrese su edad32

ingrese la temperatura37.5

**Salida de datos por pantalla**

En ocasiones no solamente queremos imprimir una frase, sinoque además queremos concatenar diferentes tipos de datos para mostrar una única línea de texto. Para concatenar el valor de una frase con el valor de una variable, utilizaremos el símbolo reservado "+". Para concatenar diferentes tipos de datos tenemos que decirle al intérprete el tipo de dato que queremos mostrar, y para ello, se debe convertir a tipo texto, haciendo un casting sobre él.

[ ]

nombre = input ("ingrese su nombre: ")  
edad = int(input ("ingrese su edad: "))  
print ("Su nombre es", nombre, "y tiene ",edad," años.-")  
print ("Su nombre es {} y tiene {} años".format(nombre,edad))  
print ("Ud. tiene {1} años y se llama {0}".format(nombre,edad))

Estructuras de programación

**Estructura de Decisión**

En programación, una condición es toda sentencia de la cual se puede determinar su veracidad (True) o falsedad (False).

En la mayoría de las ocasiones son comparaciones que podemos encontrar tanto en las decisiones como en bucles.

En Python es obligatorio intentar las instrucciones que forman parte de una decisión, bucle, función, etc. De esta forma el código queda muy legible.

En el caso de las decisiones, bucles o funciones debemos añadir **":"** para indicarle al interprete que empieza una estructura.

Por *ejemplo*, si queremos determinar si un número introducido por el usuario es positivo o negativo.

La comparación en este caso consiste en comprobar si el número es mayor o igual que 0 para el caso del positivo, y en caso contrario será negativo.

[ ]

nro = int(input ("ingrese un número: "))  
if nro >= 0:  
  print("el número es positivo")  
else:  
  print("el número es negativo")

ingrese un nro: 56

el nro es positivo

Sin embargo, en ciertas ocasiones queremos comprobar varios casos, como en este ejemplo, donde queremos saber si un número es positivo, negativo o cero. A este tipo de condición se le conoce como **condición anidada** ya que aparece una condición en el interior de otra indexada correctamente.

[ ]

nro = int(input ("ingrese un número: "))  
if nro > 0:  
  print("el número es positivo")  
elif nro == 0:  
  print("el número es cero")  
else:  
  print("el número es negativo")

ingrese un número: 0

el número es cero

**Estructuras repetitivas**

En programación, un \*bucle o estructura repetitiva \* es una sentencia que ejecuta repetidas veces un conjunto de código, hasta que la condición asignada a dicho bucle deja de cumplirse.

Los bucles más utilizados en programación son:

* el bucle for
* el bucle while

**Bucle (for)**

Un bucle for es un bucle que repite el bloque de instrucciones un número predeterminado de veces.

El bloque de instrucciones que se repite se suele llamar cuerpo del bucle y cada repetición se suele llamar iteración.

En los siguientes ejemplos se aprecia como podemos mostrar una determinada cantidad de iteraciones mediante la función range.

[ ]

for i in range(10):  
    print(i)

[ ]

for i in range(2,10):  
    print(i)

[ ]

for i in range(2,10,2):  
    print(i)

[ ]

for i in "Python":  
    print(i)

**Bucle (while)**

Un bucle while permite repetir la ejecución de un grupo de instrucciones mientras se cumpla una condición (es decir, mientras la condición tenga el valor True).

En este tipo de bucle hay que llevar cuidado porque es muy común caer en bucles infinitos cuando se empieza a programar.

En el siguiente ejemplo, decrementamos el valor de número en cada iteración.

[ ]

nro = 5  
while nro > 0:  
  print ("el número es: ",nro)  
  nro = nro - 1

Otro ejemplo, podemos escribir una parte inicial de la serie de Fibonacci así:

[ ]

a, b = 0, 1  
while a < 10:  
    print(a)  
    a, b = b, a+b

Ejercicios

Nueva sección

Variables

1. Crea una variable que contenga el año actual y muéstralo por pantalla.
2. Crea una variable con el valor de PI (3.14….) e imprímela.
3. Crea una variable con el valor Verdadero (True) e imprímela.

Operaciones

1. Pedir dos números por teclado e imprimir la suma de ambos.
2. Pedir dos números por teclado e imprimir la media aritmética.
3. Pedir peso y altura para calcular la masa corporal: mc = peso / altura^2.
4. Pedir radio para calcular la circunferencia de un círculo: C = 2*PI*r.
5. Pedir un número en Celsius y convertir a Fahrenheit: F = 1.8\*C + 32.

Condiciones

1. Pedir tres números por teclado e imprimir el mayor de ellos solamente.
2. Pedir un número por pantalla y decir si está entre 10 y 15 o no.
3. Pedir lado y alto de un cuadrilátero y decir si es cuadrado o rectángulo.
4. Definir como constantes MAX y MIN con valores de 50 y 25 respectivamente. Luego, ingresar una temperatura temp por teclado y decir:

* hace calor ( cuando es igual o superior a 50 temp>=50)
* está templado cuando es mayor o igual a 25 y menor que 50.
* hace frio si es menor a 25.

1. Solicitar el ingreso de un número, el programa deberá indicar si el número ingresado es múltiplo de 7.
2. Es sabido que un triángulo se forma con tres segmentos o lados de una determinada longitud pero, dados 3 segmentos de longitud cualesquiera, no siempre es posible construir un triángulo. Implemente un código que permita validar si a partir de 3 segmentos A, B y C los mismos representan los lados de un triángulo válido. Luego, debe informar qué tipo de triángulo es.

Bucles

1. Imprimir los 25 primeros números naturales.
2. Imprimir los números impares desde el 1 al 25, ambos inclusive.
3. Calcula e imprime la suma desde el 14 hasta el 38, ambos inclusive.
4. Calcula e imprime el producto de la serie 2x4x6x8x … x20.
5. Calcula e imprime la suma de la serie 50+48+46+ … +20.
6. Implementar un código que, ingresando 20 números, encuentre y devuelva el mayor de ellos.
7. Escribir un programa que permita

* ingresar un número N (entero positivo, si se ingresa otra cosa, se deberá mostrar un mensaje indicando un error)
* ingresar un número X (real positivo que debe estar entre 0 y π , si se ingresa otra cosa, se deberá mostrar un mensaje indicando un error)
* luego calcular y mostrar la aproximación al Seno del ángulo X como Sen (X) = (𝑥 - 𝑥3 / 3 ! + x5 / 5 ! - x7 / 7 ! + …) NOTA: N expresa la cantidad de términos a considerar en la serie de Taylor, utilizada para aproximar el Seno de X. Verificar que... a mayor N se obtiene una mejor aproximación.

1. Realizar un programa que permita adivinar un número. Para ello se solicitara el ingreso por teclado del numero a adivinar (entero). Luego se irán solicitando mas números y se deberá ir averiguando si el número a adivinar es mayor o menor que el introducido (ir mostrando carteles indicativos). El programa termina cuando se acierta el número.