PYTHON BASICO

Como vamos a trabajar y requisitos necesarios

Voy a utilizar Github para los códigos que vayamos realizando en el equipo además de este documento que lo tendremos también ahí.

1. Instalar Python

Texto

El contenido generado por IA puede ser incorrecto.

1. Visual Studio Code. Es un IDE para trabajar con cualquier lenguaje.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Python es un lenguaje de programación que se comunica con TODO.

Python se ha hecho popular por Hacking y Big Data

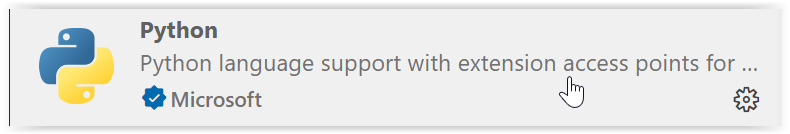
Dentro de Python tenemos multitud de características:

* Hacking: Como Python puede comunicar con cualquier dispositivo y sistema, se ha hecho popular para los programas de Hacking
* Big Data: Estos conceptos los utilizaban los matemáticos y decidieron este lenguaje porque es muy intuitivo y fácil de aprender.
* Django: Es un framework para generar páginas Web con código Python
* Fast Api/Flask: Es un Framework para generar apis con lenguaje Python
* IA: Crear modelos/agentes de inteligencia artificial
* Jupyter: Es un Framework para análisis de datos y generar gráficos

Visual Studio Code nos permite trabajar con cualquier entorno de los que hemos escrito.

Mediante extensiones de VS Code, podemos acceder a cualquier característica de Python o cualquier lenguaje.

Vamos a instalar las extensiones de Python/Jupyter dentro de VS Code.



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Todas las extensiones de ficheros de Python (mentira) terminan en **.py**

Para trabajar, vamos a crear una carpeta en nuestro ordenador y, a partir de ahí tendremos nuestros códigos.

Dependiendo del tipo de proyecto, se generan más carpetas de trabajo.

Abrimos esta carpeta desde Visual Studio Code

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.



**Importante en el lenguaje:**

* **Diferencia mayúsculas de minúsculas**
* **Las tabulaciones son básicas, son las que marcan el programa**
* **Las variables (mentira) no tienen tipado, simplemente se escriben sin declara su tipo. Se genera su tipo al almacenar un valor en su interior**

Vamos a trabajar de forma consecutiva al nombrar los programas.

Llamaremos a los programas **pythonXXdescripcion.py**

Vamos a comenzar viendo las variables y algún tipo de conversión.

Creamos un nuevo programa llamado **python01variables.py**

**print("Ejemplo de variables")**

**numero = 14**

**texto = "Primer Python"**

**#Comentarios en Python**

**#Comentar dentro de VS Code**

**#Comentar: CONTROL + K + C**

**#Descomentar: CONTROL + K + U**

**print(numero)**

**print(texto)**

**#El simbolo + sirve para concatenar y también para sumar**

**#Verifica el tipo de dato**

**print("Texto: " + texto)**

**#print("Numero: " + numero)**

**#También podemos utilizar la coma para concatenar**

**print("Numero: ", numero)**

**#print f nos permite concatenar múltiples variables en String sin**

**#importar el tipo de dato.**

**#Cada variable lógica debe ir entre llaves {}**

**#La letra f se escribe ANTES del String y fuera del string**

**print(*f*"El texto es {texto} y el número es {numero}")**

**#Dentro de las variables primitivas: str, int, boolean, float**

**#podemos convertir mediante funciones de Python**

**#str(variable): Convierte un tipo a String**

**#int(variable): Convierte un tipo a Entero**

**#float(variable): Convierte un tipo a Decimal**

**print("Numero: " + *str*(numero))**

CONDICIONALES

Un condicional permite estructurar un lenguaje según preguntas que vayamos realizando en el programa.

El código no será lineal.

Los condicionales están delimitados mediante Tabulaciones en sus instrucciones.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Operadores de comparación en Python:

> Mayor

< Menor

>= Mayor o igual

<= Menor o igual

== Igual

!= Distinto

Vamos a realizar un simple programa para visualizar la sintaxis y comparar si un número es positivo, negativo o cero.

Pediremos un número al usuario. Para pedir datos al usuario en este entorno se utiliza la función **input()**. Dicha función nos devuelve siempre un STRING, así que tendremos que convertir si deseamos tener un número.

Creamos un nuevo programa llamado **python02condicionales.py**

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Tenemos la posibilidad de incluir una condición ELSE

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Perfectamente se pueden **anidar** todas las instrucciones que deseemos, por ejemplo, si el número es cero.

Texto

El contenido generado por IA puede ser incorrecto.

Dentro del lenguaje, tenemos otra forma de preguntar dentro de un mismo IF si la pregunta tiene que ver entre sí.

Con la instrucción **ELIF** podemos seguir haciendo preguntas dentro de un mismo IF

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**DEPURACION DE CODIGO**

La depuración de código nos permite visualizar las variables dinámicamente en nuestro programa y nos permite averiguar que va sucediendo paso a paso.

En cualquier programa, tenemos tres tipos de errores:

1. **Errores de compilación:** Nos equivocamos al escribir y tenemos mal la sintaxis.

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

1. **Errores en ejecución:** El programa compila y ejecuta, pero se detiene en algún momento por alguna instrucción que no hemos controlado, por ejemplo, en el programa anterior, que escriba un texto en lugar de un número.

Texto

El contenido generado por IA puede ser incorrecto.

1. **Errores lógicos:** Mi programa funciona, todo es correcto, ningún error, pero NO hace lo que yo quiero, necesito saber qué está pasando dentro del código. Solución es utilizar Debug

Para poder depurar dentro de Python debemos crear un punto de interrupción a la izquierda de los números. En la línea que deseemos comenzar a depurar nuestro código.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Para ejecutar el programa en modo Debug, entramos en un icono a la izquierda de un bicho y pulsamos en **Run and Debug**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Cuando hablamos de condicionales, tenemos también los operadores relacionales para hacer más de una pregunta

Dos palabras clave:

* **and:** Todas las condiciones deben cumplirse
* **or:** Cualquier condición entrará dentro del IF.

Para probar esta funcionalidad vamos a realizar un ejemplo para averiguar el mayor, menor e intermedio de tres números que vamos a pedir al usuario.

Creamos un nuevo programa llamado **python03mayormenor.py**

Texto

El contenido generado por IA puede ser incorrecto.

**print("Operadores relacionales")**

**num1 = *int*(input("Numero 1: "))**

**num2 = *int*(input("Numero 2: "))**

**num3 = *int*(input("Numero 3: "))**

**mayor = 0**

**menor = 0**

**intermedio = 0**

**if (num1 >= num2 and num1 >= num3):**

**mayor = num1**

**elif (num2 >= num1 and num2 >= num3):**

**mayor = num2**

**else:**

**mayor = num3**

**if (num1 <= num2 and num1 <= num3):**

**menor = num1**

**elif (num2 <= num1 and num2 <= num3):**

**menor = num2**

**else:**

**menor = num3**

**suma = (num1 + num2 + num3)**

**intermedio = suma - mayor - menor**

**print(*f*"Mayor: {mayor}, Menor: {menor}, Intermedio: {intermedio}")**

**print("Fin de programa")**

**BUCLES**

Un bucle es una secuencia que se repite N veces en el código.

Tenemos dos tipos de bucles:

1. **while:** Bucle condicional que finaliza cuando se cumple una condición expresada en su declaración. Dentro del bucle, debemos realizar acciones para que la condición se cumpla y salga del bucle.

Sintaxis:

while (condicion == true):

#instrucciones

condicion = false

print(“Fin de programa”)

1. **for:** Bucles contador. Se repiten N veces en una secuencia y finalizan cuando el bucle ha llegado al final del contador. Se declara la propia variable contador dentro de la definición del bucle.

**Por defecto, un bucle for siempre comienza en CERO**

Sintaxis:

for **contador** in range(número final):

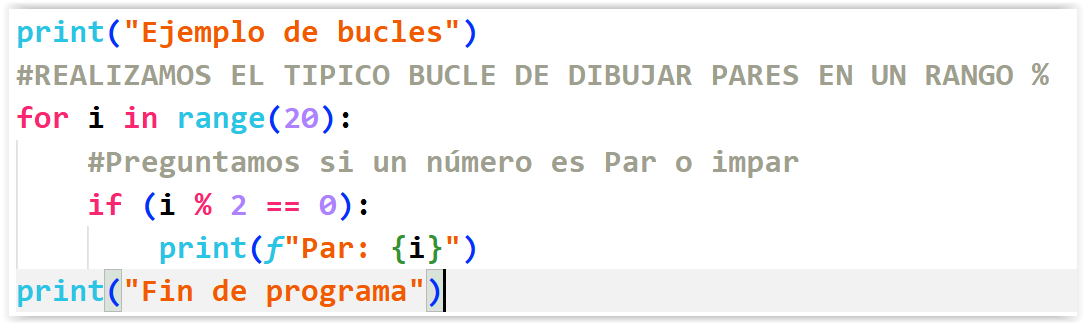
#instrucciones

Tenemos otro tipo de sintaxis en el que podemos indicar el inicio del bucle, además del final.

for **contador** in range(**inicio**, final):

#instrucciones

Creamos un nuevo programa llamado **python04bucles.py**



Vamos a crear un bucle para comprobar la Conjetura de Collatz

La conjetura de Collatz indica lo siguiente:

Todo número entero positivo será siempre 1 siguiendo estas instrucciones:

* Si el número es Par, dividimos entre 2
* Si el número es Impar, multiplicamos por 3 y sumamos 1

6, 3, 10, 5, 16, 8, 4, 2, 1

Interfaz de usuario gráfica, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.