Ejecutar código desde Visual Studio Code (VS Code)

Para ejecutar código desde Visual Studio Code (VS Code), puedes seguir estos pasos:

- Instala la extensión de Python: Si aún no tienes instalada la extensión de Python en VS Code, deberás instalarla desde la pestaña de Extensiones. Busca "Python" en la barra de búsqueda e instala la extensión oficial de Microsoft.
- Abre el archivo de código Python en VS Code.
- Configura el entorno virtual (opcional): para utilizar el entorno virtual para tu proyecto, asegúrate de activarlo. Puedes hacerlo abriendo una terminal integrada en VS Code (View > Terminal) y ejecutando los comandos necesarios para activar tu entorno virtual.
- Selecciona el intérprete de Python (opcional): Si estás utilizando múltiples versiones de Python o un entorno virtual, selecciona el intérprete de Python que deseas usar. Puedes hacerlo haciendo clic en el nombre del intérprete en la esquina inferior izquierda de la ventana de VS Code y seleccionando el intérprete adecuado.
- Ejecuta el código: Hay varias formas de ejecutar código en VS Code. Puedes hacer clic en el botón de "Run" (Ejecutar) en la esquina superior derecha del editor de texto. Puedes usar el atajo de teclado Ctrl + Alt + N. También puedes abrir una terminal integrada en VS Code (View > Terminal) y ejecutar el script Python directamente utilizando el comando python nombre_del_script.py.
- Observa la salida: La salida de tu script Python se mostrará en la terminal integrada de VS Code.

Siguiendo estos pasos, podrás ejecutar fácilmente tu código Python desde Visual Studio Code. Además, VS Code ofrece muchas otras características útiles para el desarrollo de Python, como la depuración integrada, la administración de entornos virtuales y más, que pueden facilitar tu flujo de trabajo de desarrollo. Te dejo el código.

Variables, tipos de datos, operadores

Datos Numéricos

Tipado Dinámico

```
x = 15
print ("Tipo de dato de X:", type (x))
y = 2
print ("Tipo de dato de Y:", type(y))
z = x/y
print (z)
print ("Tipo de dato de Z:", type(z))
#x= "hola"
#print (x)
#print (type(x))
```

Podemos observar que a pesar de ser las variables x, y variables enteras, la variable z se convierte en variable real ya que la división de los enteros almacenados en x, y genera como resultado un número real. A nivel de programación, esto nos elimina el conflicto entre tipos de datos y automatiza el proceso de conversión de datos. **vamos entendiendo por qué se dice que Python es un lenguaje sencillo de aprender, ¿no?**

Operadores Aritméticos

Symbol	Task Performed
+	Suma

Symbol	Task Performed
-	Resta
/	División
%	Residuo
*	Multiplicación
//	División entera
**	Exponenciación
~	Negación

Podrías decirnos antes de ejecutar el siguiente código cuál es el resultado de cada operación:

```
#print (5/2)
print (5%2)
print (5**2)
print (15//4)
```

Desde python 3.6 los números enteros se pueden expresar con un guion bajo _ que les brinde mayor claridad de lectura. Por ejemplo si queremos sumar 1250000 + 3240000 podemos expresarlo de la siguiente manera:

```
x=1_250_000 + 3_240_000
y=1250000 + 3240000
print (x, y)
```

Operadores Relacionales

Símbolo	Tarea
==	Verdadero, si valores son iguales
is	Verdadero, si son idénticos, por ejemplo el **mismo** objeto
!=	Verdadero, si no son iguales
<	Menor que
>	Mayor que
<=	Menor o igual a
>=	Mayor o igual a
in	Comprueba si un elemento está dentro de una colección (list, set, dictionary)

```
#print (3 > 5)
#print (3 <= 5)
#print (3 != 5)
print (type(3) is type(5))
print (type(3) is type(5.0))
print (3 in [4,5,6])
print (3 in [4,3,5])
print ("1"==1)
```

Operadores Lógicos

Operador	Nombre	Ejemplo
and	Devuelve True si ambos elementos son True	True and True = True
or	Devuelve True si al menos un elemento es True	True or False = True
not	Devuelve el contrario, True si es Falso y viceversa	not True = False

Puedes identificar qué pasa en el siguiente código con una persona

a) soltera, linda de 20 años y no buena persona?

b) buena persona, no linda, soltera y de 45 años?

c) soltera de 31 años, no buena persona y linda?

d) casada, de 25 años, linda y buena persona?

Los números flotantes

Acompañados de parte decimal. Ejemplo:

0.2

.25

4.5

E xponente significa diez a la n potencia ¿Cuál será el resultado de las siguientes líneas?

```
print (2**3)
#print (2E3)
#print (2.5E2)
#print (.5E4)
```

Tipos de datos Cadena

Las cadenas en python son objetos inmutables a los que les podemos aplicar diferentes métodos. Otra característica es que cada cadena está conformada por un conjunto de caracteres a los cuales podemos acceder mediante sus índices

```
cadena = "Python"
print(type(cadena))
print (cadena)
print (cadena [0])
print (cadena [3])
print (cadena.count("o"))
```

```
<class 'str'>
Python
P
h
1
```

Cadenas con múltiples líneas

```
#Cadenas multilíneas
cad ="""Escucha hermano
        la canción de
        la alegría
        una linea mas"""
print (cad)

cad2 = "\nEl canto alegre\ndel que espera\nun nuevo\ndia"
print (cad2)
```

```
#Concatenación y Formateo de cadenas
print ("1) hola "+"Mundo "+"python"+ str(3))
print ("2) hola","Mundo","python",3)


lenguaje,version = "python",3   #asignación múltiple
print(lenguaje)
print(version)
```

```
lenguaje,version = "python",3   #asignación múltiple

print ("3) hola Mundo %s %s" %(lenguaje, version))

print ("4) hola Mundo {} {}".format(lenguaje, version))

print (f"5) hola Mundo {lenguaje} {version}")

print ("6) hola",lenguaje,"mundo")
```

Un dato curioso es que en python podemos multiplicar las cadenas.

```
cadena ="si,"
print (cadena*10,"este amor es tan profundo" )
print ((("*"*8)+"\n")*4)
```

```
si,si,si,si,si,si,si,si,si,si, este amor es tan profundo
********
********
********
********
```

Podrías decir el resultado del siguiente mensaje:

```
print("+" + 10 * "-" + "+")
print(("|" + " " * 10 + "|\n") * 5, end="")
print("+" + 10 * "-" + "+")
```

```
+----------+
|          |
|          |
|          |
|          |
|          |
+----------+
```

Slicing de Cadenas

Podemos tomar parte de una cadena a partir de los índices de sus posiciones internas.

En este caso se reciben dos índices separados por dos puntos **:** El primer índice se refiere a la posición inicial y el segundo a la posición final -1. En caso de faltar alguno de sus índices, se toma el valor por defecto para dicho índice.

Ejemplos:

```
cadena="lenguaje de programación python"
print (cadena[0:8])
print (cadena[9:11])
```

```
#imprima solo la palabra programacion
cadena="lenguaje de programación python"
print (cadena[12:24])
```

```
#que imprime???
cadena="lenguaje de programación python"
print (cadena[-6:], cadena[:8])
```

```
python lenguaje
```

Comprendiendo los valores por defecto y valores negativos en el slicing

```
cadena="lenguaje de programación python"
print (cadena[:3])
print (cadena[25:])
print (cadena[-19:])
print (cadena[-6:-2])
```

Conversiones entre tipos

Python cuenta con la conversión de tipos de datos (*cast* o *casting*) entre variables

```
int(variable)
float(variable)
str(variable)
```

ejemplo:

```
a, b = "3","5"

print (a+b)

#print (int(a)+int(b))   #pasando strings a enteros
```

```
print ("python", 3)
print ("python " + str(3))
```

Métodos de cadenas

A las cadenas les podemos aplicar una serie de métodos útiles predefinidos en python. Acá un par de ejemplos:

```
cadena = "mi pObre angelito".lower()#convierte a minuscula
cadena_mayuscula = cadena.upper() #convierte a mayúsculas
print (cadena_mayuscula)

numero_de_o = cadena.count("o")   #devuelve el número de veces que se encuentra una subcadena
print (numero_de_o)


print (f"la letra 'o' está {numero_de_o} veces en la palabra/frase '{cadena}'")
```

Lectura de datos

La función **input** permite realizar la lectura de datos de entrada por parte del usuario. La función input devuelve lo ingresado por el usuario en forma de cadena, si se desea capturar un dato de otro tipo se debe hacer la respectiva conversión

```
marca = input ("Ingrese la marca del auto:")

print ("La marca ingresada por el usuario es", marca)

precio = float(input("Ingrese el precio:"))
print (f"El auto marca {marca} tiene un precio de ${precio} y con el descuento del 10% le queda en ${precio*.9}")


#btw
print ("El auto marca {} tiene un precio de ${:,} y con el descuento del 10% le queda en ${:,}".format(marca,precio,precio*0.9))
```

Apropiación 1

1. **Tipos de datos y métodos básicos de formateo:**

Solicita al usuario que ingrese su nombre y su edad, luego imprime un mensaje personalizado utilizando f-strings para saludarlo y decirle cuántos años tiene.

2. Variables:

Solicite al usuario que ingrese dos numeros enteros, intercambie los valores de las variable sin usar una variable temporal e imprima los valores intercambiados.

3. Operadores aritméticos:

Pide al usuario que ingrese tres números y realiza las siguientes operaciones aritméticas con ellos:

- Suma de los tres números.
- Resta del tercer número al resultado de la suma de los dos primeros.
- Multiplicación de los tres números.
- División del primer número entre la suma del segundo y tercer número. Imprime los resultados.

4. Operadores Relacionales y lógicos:

Pide al usuario que ingrese tres números. Utiliza solo operadores lógicos y relacionales para verificar si al menos dos de los números ingresados son iguales entre sí. Imprime un mensaje indicando el resultado de la verificación, por ejemplo: "Al menos dos de los números son iguales:" False

5. Operadores Aritmeticos

Realice un programa que calcule el índice de cosecha de un cultivo en función de la cantidad de frutos recolectados y la cantidad de frutos producidos en total.

Fórmula:

```
Índice de cosecha = (cantidad de frutos recolectados / cantidad de frutos producidos) * 100%
```

---

6. Manipulación de Cadenas:

Pide al usuario que ingrese una frase y un carácter. Luego, cuenta cuántas veces aparece ese carácter en la frase e imprime el resultado.

7. Manipulación de cadena

Dibujar la P de Python que abarque 7 filas y 5 columnas. Use solo una línea de código

8. Operadores Aritméticos

Un alumno desea saber cuál será su calificación final en la materia de Matemáticas. Dicha calificación se compone de los siguientes porcentajes:

- 55% del promedio de sus tres calificaciones parciales (se debe leer cada calificación parcial)
- 30% de la calificación del examen final.
- 15% de la calificación de un trabajo final.

9. Ejercicio de búsqueda de subcadena:

Imagina que estás diseñando un programa para un sistema de facturación donde necesitan redondear los montos de las transacciones financieras al valor entero mayor más próximo al valor entregado.

Debera solicitarle al usuario el monto de 5 transacciones todas con valores reales, además calcular el valor total de lo que se perdio en el redondeo en todas las transacciones y presentar este valor con un redondeo de hasta 3 decimales. Debes mostrar cada monto original y el valor redondeado, la suma de todas las transacciones redondeadas y el valor que se perdio del redondeo.

10. Ejercicio de conversión de mayúsculas/minúsculas:

Pide al usuario que ingrese una frase. Luego, utiliza los métodos upper() y lower() para imprimir la frase en mayúsculas y minúsculas, respectivamente.