ej_1

September 29, 2025

0.1 Variables

1) Crear una variable que contenga un elemento del conjunto de números enteros y luego imprimir por pantalla

[1]:
```
numero_entero = 10
print(numero_entero)
```

```
10
```

2) Imprimir el tipo de dato de la constante 8.5

[2]:
```
print(type(8.5))
```

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

3) Imprimir el tipo de dato de la variable creada en el punto 1

[3]:
```
print(type(numero_entero))
```

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

4) Crear una variable que contenga tu nombre

[4]:
```
nombre = "Lennin Temoche" # Reemplaza "Tu Nombre" con tu nombre real
print(nombre)
```

```
Lennin Temoche
```

5) Crear una variable que contenga un número complejo

[5]:
```
numero_complejo = 3 + 4j
print(numero_complejo)
```

```
(3+4j)
```

6) Mostrar el tipo de dato de la variable crada en el punto 5

[6]:
```
print(type(numero_complejo))
```

```
<class 'complex'>
```

7) Crear una variable que contenga el valor del número Pi redondeado a 4 decimales

[7]:
```
import math
pi_redondeado = round(math.pi, 4)
print(pi_redondeado)
```

```
3.1416
```

8) Crear una variable que contenga el valor 'True' y otra que contenga el valor True. ¿Se trata de lo mismo?

[8]:
```
variable_string = 'True'
variable_boolean = True
# No se trata de lo mismo. 'True' es una cadena de texto (string), mientras que␣
↪True es un valor booleano.
```

9) Imprimir el tipo de dato correspondientes a las variables creadas en el punto 8

[9]:
```
print(type(variable_string))
print(type(variable_boolean))
```

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

10) Asignar a una variable, la suma de un número entero y otro decimal

[10]:
```
suma_entero_decimal = 5 + 3.14
print(suma_entero_decimal)
```

```
8.14
```

11) Realizar una operación de suma de números complejos

[11]:
```
complejo1 = 2 + 3j
complejo2 = 1 - 2j
suma_complejos = complejo1 + complejo2
print(suma_complejos)
```

```
(3+1j)
```

12) Realizar una operación de suma de un número real y otro complejo

[12]:
```
real = 10
complejo = 5 + 2j
suma_real_complejo = real + complejo
print(suma_real_complejo)
```

```
(15+2j)
```

13) Realizar una operación de multiplicación

[13]:
```
num1 = 7
num2 = 3
multiplicacion = num1 * num2
print(multiplicacion)
```

```
21
```

14) Mostrar el resultado de elevar 2 a la octava potencia

[14]:
```
potencia = 2 ** 8
print(potencia)
```

```
256
```

15) Obtener el cociente de la división de 27 entre 4 en una variable y luego mostrarla

[15]:
```
cociente = 27 / 4
print(cociente)
```

```
6.75
```

16) De la división anterior solamente mostrar la parte entera

[16]:
```
parte_entera = 27 // 4
print(parte_entera)
```

```
6
```

17) De la división de 27 entre 4 mostrar solamente el resto

[17]:
```
resto = 27 % 4
print(resto)
```

```
3
```

18) Utilizando como operandos el número 4 y los resultados obtenidos en los puntos 16 y 17. Obtener 27 como resultado

[18]:
```
resultado_16 = 27 // 4
resultado_17 = 27 % 4
resultado = (resultado_16 * 4) + resultado_17
print(resultado)
```

```
27
```

19) Utilizar el operador "+" en una operación donde intervengan solo variables alfanuméricas

[19]:
```
cadena1 = "Hola"
cadena2 = " Mundo"
concatenacion = cadena1 + cadena2
print(concatenacion)
```

```
Hola Mundo
```

20) Evaluar si "2" es igual a 2. ¿Por qué ocurre eso?

[20]:
```
print("2" == 2)
# Esto ocurre porque "2" es una cadena de texto (string) y 2 es un número␣
 ↪entero (int).
```

```
# Python considera que son tipos de datos diferentes y por lo tanto no son␣
 ↪iguales en este caso.
```

```
False
```

21) Utilizar las funciones de cambio de tipo de dato, para que la validación del punto 20 resulte verdadera

[21]:
```
print(int("2") == 2)
# O también:
print("2" == str(2))
```

```
True
True
```

22) ¿Por qué arroja error el siguiente cambio de tipo de datos? a = float('3,8')

[22]:
```
a = float('3,8')
# Arroja error porque en Python, el separador decimal es el punto (.), no la␣
 ↪coma (,).
# Para que funcione, debería ser: a = float('3.8')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[22], line 1
----> 1 a = float(     )
      2 # Arroja error porque en Python, el separador decimal es el punto (.),␣
 ↪no la coma (,).
      3 # Para que funcione, debería ser: a = float('3.8')

ValueError: could not convert string to float: '3,8'
```

23) Crear una variable con el valor 3, y utilizar el operador '-=' para modificar su contenido y que de como resultado 2.

[23]:
```
a = 3
a -= 1 # Esto es equivalente a: a = a - 1
print(a)
```

```
2
```

24) Realizar la operacion 1 « 2 ¿Por qué da ese resultado? ¿Qué es el sistema de numeración binario?

[24]:
```
print(1 << 2)
# El resultado es 4.
# Esto es una operación de desplazamiento de bits a la izquierda. El número 1␣
 ↪en binario es 0001.
```

```
# El operador << 2 desplaza los bits dos posiciones a la izquierda, agregando␣
 ↪ceros al final: 0100.
# En el sistema decimal, 0100 en binario es 4.
```

4

25) Realizar la operación 2 + '2' ¿Por qué no está permitido? ¿Si los dos operandos serían del mismo tipo, siempre arrojaría el mismo resultado?

[25]:
```
print(2 + '2') # Esto arrojaría un TypeError
# No está permitido porque estás intentando sumar un número entero (int) y una␣
 ↪cadena de texto (string).
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 print(2 +      ) # Esto arrojaría un TypeError
      2 # No está permitido porque estás intentando sumar un número entero (int␣
 ↪y una cadena de texto (string).

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

26) Realizar una operación válida entre valores de tipo entero y string

[26]:
```
# Esto se puede hacer multiplicando una cadena por un entero para repetirla.
print("Hola " * 3)
# También se puede convertir el entero a string para concatenar.
print("El número es: " + str(10))
```

```
Hola Hola Hola
El número es: 10
```