Tipos de datos

En Python tenemos varios tipos o estructuras de datos, que son fundamentales en programación ya que almacenan información, y nos permiten manipularla, Ejemplo:

- **Texto(str)** ---- “Python” “750”
- **Números(int, floats)** ---- 250 12.50 -3.14.16
- **Booleanos(bool)** ---- True False

estructuras	mutable	ordenado	duplicados
listas [ ]	✓	✓	✓
tuplas ( )	✗	✓	✓
sets { }	✓	✗	✗
diccionarios { }	✓	✗*	✗:✓**

Listas (list)

Listas (list): Una lista es una colección ordenada de objetos, se escriben entre corchetes y pueden contener todo tipo de objetos. Las listas tienen un orden en sus elementos determinado por un índice que comienza de 0 a N.

Ejemplo:

- [“juan”,”pablo”,”ana”]
- [1,500,97,-6000,0]
- [52.3,1.255,-8.0001]
- [“sal”,1,-3,4.5,”marte”,0]

Diccionarios (dic)

Su contenido se escribe entre llaves { } y consisten en pares de palabras agrupados, cada par tiene 2 elementos, una clave : y un valor.
Estos pares no están ordenados con un índice dentro del diccionario.

Ejemplo:

- {‘color’:’rojo’, ‘arte’:’cine’}

- {‘Equipo’:’Barcelona’, ‘Jugador’:’Messi’}

Tuples (tuple)

Su contenido se escribe entre paréntesis ( ) y son muy parecidos a las listas, ya que son un conjunto ordenado de elementos y sus elementos pueden ser de cualquier tipo de datos, pero tienen una diferencia con las listas y es que su orden es inmutable, osea que no se puede cambiar.

Ejemplo:

- (‘lun’, ‘mar’, ‘mie’, ‘jue’, ‘vie’, ‘sab’, ‘dom’,)

- (‘ene’, ‘feb’, ‘mar’, ‘abr’, ‘may’, ‘jun’, ‘jul’, ‘ago’, ‘sep’, ‘oct’, ‘nov’, ‘dic’)

Sets (set)

Son un conjunto de elementos ordenados únicos, las listas y los tuples podrían contener elementos duplicados, los Sets NO. Cada uno de los elementos de un set son únicos e irrepetibles.

Ejemplo:

- (‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’)

- (‘a’, ‘e’, ‘i’, ‘o’, ‘u’)

Booleanos (bool)

Un dato booleano solo puede tener dos valores, True o False.

Estos tipos de datos son muy útiles cuando necesitamos que nuestro código tome decisiones de si se cumple una condición (true) haga algo, y sino se cumple la condición (false) haga esto otro.

Ejemplo:

- (100 > 50) True

- (100 % 50 != 0) False

Variables

Las variables son espacios de memoria que almacenan valores o datos de distintos tipos, y (como su nombre lo indica) pueden variar.
Se crean en el momento que se les asigna un valor, por lo cual en Python no requerimos declararlas previamente.

Integers y Floats

Existen dos tipos de datos numéricos básicos en Python: **int** y **float**.
Como toda variable en Python, su tipo queda definido al asignarle un valor a una variable.
La función **type()** nos permite obtener el tipo de dato almacenado en una variable.

Int, o integer, es un número entero, positivo o negativo, sin decimales, de un largo indeterminado.

```
1    num1 = 7
2    print(type(num1))
3
```

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

Integers y Floats

Float, o "número de punto flotante" es un número que puede ser positivo o negativo, que a su vez contiene una o más posiciones decimales.

```
1       num2 = 7.525587
2       print(type(num2))
3
```

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

Integers y Floats

No es posible realizar operaciones aritméticas con cadenas o caracteres.

```
1    edad = input("Dime tu edad: ")
2    print("Tu edad es " + edad + 1)
3
4    print(type(edad))
5
```

Escribe este código y comprueba su resultado.

Ejercicios con Integers y Floats

Declara una variable numérica llamada **num_entero** que contenga un valor de tipo integer. Imprime el tipo de dato de dicha variable.

Declara una variable numérica llamada **num_decimal** que contenga un valor de tipo float. Imprime el tipo de dato de dicha variable.

De que tipo es el resultado de la suma de 7.5 + 2.5? Genera el código para verificarlo. Crea dos variables llamadas num1=7.5 y num2=2.5.
Muestra en pantalla el tipo de dato que resulta de la suma de ambos números.

Conversiones entre tipos de datos

Python realiza conversiones implícitas de tipos de datos automáticamente para operar con valores numéricos.

En otros casos, necesitaremos generar una conversión de manera explícita.

- **int(var)** ---- Convierte el dato en integer

- **float(var)** ---- Convierte el dato en float

Conversiones implícitas

Observa los dos códigos a continuación y nota la diferencia en los tipos de datos impresos en pantalla por la función **type()**.

```
1   num1 = 20
2   num2 = 30.5
3
4   print(type(num1))
5   print(type(num2))
```

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

```
1   num1 = 20
2   num2 = 30.5
3
4   num1 = num1 + num2
5
6   print(type(num1))
7   print(type(num2))
```

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

En el segundo código, **num1** se convierte automáticamente en un tipo de dato **float** después de realizar una operación aritmética con **num2**.

Conversiones explicitas

Observa los dos códigos a continuación y nota la diferencia en los tipos de datos impresos en pantalla por la función **type()**.

```
1    num1 = 3.1416
2    num2 = int(num1)
3
4    print(num2)
5    print(type(num2))
```

```
3
<class 'int'>
```

```
1    edad = int(input("Dime tu edad: "))
2    nueva_edad = edad + 1
3
4    print("Tu nueva edad es: ")
5    print(nueva_edad)
```

```
Dime tu edad: 40
Tu nueva edad es:
41
```

Ejercicios de Casting

1.Convierte el valor de num1=7.5 en un int e imprime el tipo de dato que resulta.

2.Convierte el valor de num2=10 en un float e imprime el tipo de dato que resulta.

3.Suma los valores de **num1="7.5"** y **num2="10"**. No modifiques el valor de las variables ya declaradas, sino aplica las conversiones necesarias dentro de la función **print()**.

Formatear cadenas (format)

Para facilitar la concatenación de variables y texto en Python, contamos con dos herramientas que nos evitan manipular las variables, para incorporarlas directamente al texto:

Función format: se encierra las posiciones de las variables entre corchetes { }, y a continuación del string llamamos a las variables con la función format

```
1   color_auto = "Rojo"
2   matricula = "KQN-700"
3   print("Mi auto es {} y de placas {}".format( *args: color_auto, matricula))
```

```
C:\PycharmProjects\Analitica\.venv\Scrip
Mi auto es Rojo y de placas KQN-700

Process finished with exit code 0
```

Formatear cadenas (literales)

Para facilitar la concatenación de variables y texto en Python, contamos con dos herramientas que nos evitan manipular las variables, para incorporarlas directamente al texto:

Cadenas literales (f-strings): A partir de Python 3.8, podemos anticipar la concatenación de variables anteponiendo **f** al string.

```
1   color_auto = "Rojo"
2   matricula = "KQN-700"
3   print(f"Mi auto es {color_auto} y de placas {matricula}")
```

```
C:\PycharmProjects\Analitica\.venv\Scr
Mi auto es Rojo y de placas KQN-700

Process finished with exit code 0
```

Ejercicio Formatear cadenas 1

Necesitamos imprimir el nombre y numero de asociado dentro de la siguiente frase:

Estimado/a (nombre_asociado), su
número de asociado es:
(numero_asociado)

El texto debe quedar en 3 líneas tal como se muestra, y todo debe tener sus espacios y todo para que el resultado tenga coherencia.

Ejercicio Formatear cadenas 2

Muestra al usuario la cantidad de puntos acumulados dentro de la siguiente frase:

Has ganado (puntos_nuevos) puntos! En
Total, acumulas (puntos_totales)
puntos

El texto debe quedar en 3 líneas tal como se muestra, y todo debe tener sus espacios y todo para que el resultado tenga coherencia.

Ejercicio Formatear cadenas 3

Muestra al usuario la cantidad de puntos acumulados dentro de la siguiente frase:

Has ganado (puntos_nuevos) puntos! En
Total, acumulas (puntos_totales)
puntos

La cantidad de puntos acumulados (totales) será igual a los **puntos_anteriores** más los **puntos_nuevos**.

El texto debe quedar en 3 líneas tal como se muestra, y todo debe tener sus espacios y todo para que el resultado tenga coherencia.

Operadores matemáticos

Veamos cuales son los operadores matemáticos básicos de Python, que utilizaremos para realizar cálculos:

Suma:	`+`
Resta:	`-`
Multiplicación:	`*`
División:	`/`
Cociente (división "al piso"):	`//`
Resto (módulo):	`%`
Potencia:	`**`
Raíz cuadrada:	`**0.5`

útil para detectar valores pares ;)

¡es un caso especial de potencia!

Operadores matemáticos

Formatearemos la salida de datos por pantalla y realizamos la operación suma.

```
1   x = 6
2   y = 2
3
4   print(f"{x} mas {y} es igual a {x+y}")
```

```
C:\PycharmProjects\Analitica\.venv
6 mas 2 es igual a 8

Process finished with exit code 0
```

Operadores matemáticos

```python
x = 10
y = 5
z = 25

print(f"{x} + {y} = {x+y}")
print(f"{x} - {y} = {x-y}")
print(f"{x} * {y} = {x*y}")
print(f"{x} / {y} = {x/y}")
print(f"{x} // {y} = {x//y}")
print(f"{x} % {y} = {x%y}")
print(f"{x} ^ {y} = {x**y}")
print(f"Raíz cuadra de {z} = {z**0.5}")
```

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
10 // 5 = 2
10 % 5 = 0
10 ^ 5 = 100000
Raíz cuadra de 25 = 5.0
```

Ejercicios con operadores matemáticos

Muestra en pantalla el cociente (división al piso) de los siguientes dos números: 874 dividido entre 27.

Muestra en pantalla el módulo (es decir, el residuo) de la división entre 456 y 33. Debes mostrar solo el valor numérico que resulta de esta operación.

Calcula y muestra en pantalla la raíz cuadrada de 783. Debes mostrar solo el valor numérico que resulta de esta operación.

Redondeo

El redondeo facilita la interpretación de los valores calculados al limitar la cantidad de decimales que se muestran en pantalla.
También, nos permite aproximar valores decimales al entero más próximo.

round(number,ndigits)

Si se omite **ndigits** el resultado será un numero entero.

```
1   print(round(100/3))
2
3   print(round(12/7, 2))
```

```
C:\PycharmProjects\Analitica\.venv\Sc
33
1.71

Process finished with exit code 0
```

Redondeo

El redondeo facilita la interpretación de los valores calculados al limitar la

```
1    resul1 = 90/7
2    resul2 = round(90/7)
3    resul3 = round(90/7, 2)
4
5    print(f"Resultado 1 = {resul1}\nResultado 2 = {resul2}\nResultado 3 = {resul3}")
```

```
C:\PycharmProjects\Analitica\.venv\S
Resultado 1 = 12.857142857142858
Resultado 2 = 13
Resultado 3 = 12.86

Process finished with exit code 0
```

Ejercicios Redondeo

Redondea el resultado de la división **10/3** a un número con **2 decimales**, y muestra en pantalla el valor redondeado.

Redondea el número **10.676767** al entero mas próximo, y muestra en pantalla el resultado.

Calcula la **raíz cuadrada de 5**, y muestra en pantalla el resultado redondeado con **4 posiciones decimales**.

Actividad Unidad 2

Realice un programa que pida al usuario su nombre y el monto de ventas totales que ha obtenido en un mes. El programa debe calcular cual es el monto de comisión que ganara un trabajador de acuerdo al monto vendido por mes.

- Este programa debería comenzar preguntando cosas al usuario, por lo tanto, vas a necesitar input para poder recibir los ingresos del usuario y deberías usar variables para almacenar esos ingresos. Recuerda que los ingresos de usuarios se almacenan como strings. Por lo tanto, deberías convertir uno de esos ingresos en un float para poder hacer operaciones con él.

- ¿Y qué operaciones necesitas hacer? Bueno, calcular el 10% del monto que haya ingresado el usuario. Es decir, que debes multiplicar ese monto por 10 y luego dividirlo por 100, o simplemente puedes usar (0,1). Recuerda almacenar ese resultado en una variable.

- Sería bueno que para imprimir en pantalla el resultado te asegures de que esa información no tenga más de dos decimales, para que sea fácil de leer, y luego organiza todo eso en un string al que debes dar formato para mostrar todo en pantalla.