1.- ¿Cuál es la diferencia entre una lista y una tupla en Python?

- Listas: Son mutables, se pueden modificar sus elementos después de crearlas.
- Tuplas: Son inmutables, no se pueden modificar sus elementos después de crearlas.

```
# Lista - Mutable
my_list = ["Mallorca", "Menorca", "Ibiza", "Tabarca"]
my_list[3] = "Formentera" # cambia Tabarca por Formentera
my_list.append("Cabrera") # añade el elemento "Cabrera"
print(my_list)
```

✓ 0.0s

```
['Mallorca', 'Menorca', 'Ibiza', 'Formentera', 'Cabrera']
```

```
# Tupla - Inmutable
my_tuple = ("Cantabrico", "Atlantico", "Mediterraneo")
my_tuple[1] = "Adriatico" # Nos devuelve un error porque las tuplas son inmutables
```

⊗ 0.2s

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 3
      1 # Tupla - Inmutable
      2 my_tuple = ("Cantabrico", "Atlantico", "Mediterraneo")
----> 3 my_tuple[1] = "Adriatico"

TypeError: 'tuple' object does not support item assignment
```

Buenas prácticas:

Se recomienda usar listas en el caso de que las colecciones puedan cambiar con el paso del tiempo. Sin embargo, es más recomendable el uso de tuplas para los datos que no deban modificarse, así evitamos errores o posibles fallos.

2.- ¿Cuál es el orden de las operaciones?

Python sigue el orden de operaciones conocido como PEMDAS:

P Parentheses ()

E Exponents **

M Multiplication *

D Division \

A Addition +

S Substraction -

```python
rdo = 7 + 2 * 3 ** 2 - (8 / 2)
print(rdo)

"""
Orden de las operaciones (PEMDAS):
Paréntesis: (8 / 2) = 4 -> 7 + 2 * 3 ** 2 - 4
Exponentes: 3 ** 2 = 9 -> 7 + 2 * 9 - 4
Multiplicación: 2 * 9 = 18 -> 7 + 18 - 4
Adición: 7 + 18 = 25 -> 25 - 4
Sustracción: 25 - 4 = 21
"""
```

✓ 0.0s

```
21.0
```

Buenas prácticas:

Se recomienda usar paréntesis para que haya más claridad incluso cuando no sean necesarios. Ejemplo: 15 + (4 * 3).

3.- ¿Qué es un diccionario Python?

Es una colección de elementos que contiene una llave (key) y un valor (value). Los diccionarios son mutables.

```python
my_dict = {
    1: "Python",
    2: "JavaScript",
    3: "SQL"
}

my_dict[4] = "R" # Son mutables. Nos permite añadir más elementos

print(my_dict)
```

✓ 0.0s

```
{1: 'Python', 2: 'JavaScript', 3: 'SQL', 4: 'R'}
```

Buenas prácticas:

Se recomienda representarlo de esta manera en lugar de todo en una línea seguida para que tenga una mayor legibilidad y se puedan identificar mejor las claves y los valores.

4.- ¿Cuál es la diferencia entre el método ordenado y la función de ordenación?

- Método ordenado sort: Modifica la lista original. Sólo se puede usar con las listas.
- Función de ordenación sorted: No modifica la lista original. Devuelve una nueva lista ordenada. Se puede usar con las listas, las tuplas, etc.

```
# Método sort()
my_list = ["Grease", "It", "Titanic", "Armageddon", "Troya", "Wonka", "Alien"]
my_list.sort()
print(my_list)
```
✓ 0.0s
```
['Alien', 'Armageddon', 'Grease', 'It', 'Titanic', 'Troya', 'Wonka']
```

```
# Función sorted()
my_list = ["Grease", "It", "Titanic", "Armageddon", "Troya", "Wonka", "Alien"]
new_list = sorted(my_list)
print(new_list)
print(my_list)
```
✓ 0.0s
```
['Alien', 'Armageddon', 'Grease', 'It', 'Titanic', 'Troya', 'Wonka']
['Grease', 'It', 'Titanic', 'Armageddon', 'Troya', 'Wonka', 'Alien']
```

Buenas prácticas:

Se recomienda usar el método sort cuando no se necesite conservar la lista original. Por el contrario, se recomienda usar la función sorted cuando queramos mantener la lista original y trabajar con una copia ordenada.

5.- ¿Qué es un operador de reasignación?

Es un método que se utiliza para modificar el valor de una variable. Tipos de operadores: +=, -=, *=, /=.

```
# Operador de reasignación
my_var = 20
my_var += 7  # Es lo mismo que: my_var = my_var + 7
print(my_var)
```
✓ 0.0s
```
27
```

Buenas prácticas:

Se recomienda el uso de operadores de reasignación para simplificar el código.