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

En **Python**, tanto las listas como las tuplas son secuencias de elementos, pero tienen una diferencia clave: las listas son mutables (puedes cambiar, agregar o eliminar elementos), mientras que las tuplas son inmutables (no se pueden modificar una vez creadas).

Mutabilidad:

Lista: Las listas son mutables, lo que significa que puedes modificar, agregar o eliminar elementos después de su creación.

Tupla: Las tuplas son inmutables, una vez creadas no pueden modificarse. No puedes agregar, eliminar ni cambiar elementos en una tupla.

Sintaxis:

Lista: Se define utilizando corchetes [].

Tupla: Se define utilizando paréntesis ().

Aplicaciones:

Lista: Las listas son más adecuadas cuando necesitas una colección de elementos que pueda cambiar a lo largo del tiempo, como una lista de tareas pendientes o los elementos de un carrito de compras.

Tupla: Las tuplas son útiles cuando deseas garantizar que los datos no cambien accidentalmente, por ejemplo, cuando estás representando coordenadas geográficas o la fecha de nacimiento de una persona.

```
# Lista (mutable)
lista = [1, 2, 3, 4]
lista.append(5)  # Agrega un elemento al final
lista[0] = 0     # Modifica un elemento
print(lista)     # Output: [0, 2, 3, 4, 5]
# Tupla (inmutable)
tupla = (1, 2, 3, 4)
# Intentar cambiar un elemento genera un error
# tupla[0] = 0   # Genera un TypeError
```

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

En **Python**, las operaciones se ejecutan siguiendo las reglas estándar de precedencia de operadores. El orden de las operaciones sigue la regla **PEMDAS**, que significa: Paréntesis, Exponentes, Multiplicación y División (de izquierda a derecha), Adición y Sustracción (de izquierda a derecha).

PEMDAS:

Paréntesis (Parentheses): Las operaciones dentro de paréntesis se realizan primero.

Exponentes (Exponents): Las operaciones de exponentes se evalúan a continuación.

Multiplicación y División (Multiplication and Division): Las operaciones de multiplicación y división se realizan en el orden en que aparecen de izquierda a derecha.

Adición y Sustracción (Addition and Subtraction): Las operaciones de adición y sustracción se realizan en el orden en que aparecen de izquierda a derecha.

```
resultado = 10 + 2 * 3  # Multiplicación se evalúa primero
print(resultado)        # Output: 16
```

3. ¿Qué es un diccionario Python?

Un diccionario en **Python** es una estructura de datos que permite almacenar pares de clave-valor. Cada elemento dentro de un diccionario consiste en una clave única y su correspondiente valor. Esto significa que se puede acceder rápidamente a un valor utilizando su clave, lo que hace que los diccionarios sean muy eficientes para buscar y recuperar datos.

```
diccionario = {"nombre": "Juan", "edad": 25, "ciudad": "Madrid"}
print(diccionario["nombre"])  # Accede al valor de la clave "nombre"
```

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

En **Python**, sort()es un método de lista que modifica la lista original en su lugar, mientras que sorted() es una función incorporada que devuelve una nueva lista ordenada sin modificar el iterable original. La elección entre usar sort() y sorted() depende de si deseas mantener la lista original sin cambios o si estás bien con modificar la lista directamente. **sorted**() –

Función de ordenación:

La función sorted() es una función incorporada en **Python** que se utiliza para ordenar cualquier iterable, como listas, tuplas, cadenas, etc.

sorted() devuelve una nueva lista ordenada sin modificar la lista original.

Se puede utilizar con cualquier iterable y admite argumentos opcionales para personalizar el ordenamiento, como el argumento clave y el argumento inverso.

```
lista = [3, 1, 2]
lista_ordenada = sorted(lista)
print(lista_ordenada)  # Output: [1, 2, 3]
print(lista)           # Output: [3, 1, 2] (la lista original no cambia).
```

5. ¿Qué es un operador de reasignación?

Un operador de reasignación se utiliza para cambiar el valor de una variable en **Python**. El operador más común para reasignar valores es el signo de igual (=). los operadores de asignación o *assignment operators* **permiten realizar una operación y almacenar su resultado en la variable inicial**.

Ejemplo:

```
x = 5        # Se asigna el valor 5 a la variable x
x = x + 1    # Se incrementa el valor de x en 1 (ahora x es 6)
print(x)     # Output: 6
```

Ejemplo con x=7

Operador	Ejemplo	Equivalente
=	x=7	x=7
+=	x+=2	x=x+2 = 7
-=	x-=2	x=x-2 = 5
*=	x*=2	x=x*2 = 14