M2C4

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

Las tuplas son colecciones ordenadas e inmutables de elementos. Son como listas, pero con una diferencia clave:

- **Inmutables**: No se pueden modificar después de crearlas (no puedes añadir, eliminar o cambiar elementos).
- **Eficientes**: Usan menos memoria que las listas.
- **Seguras**: Ideales para datos que no deben cambiar (días de semana, coordenadas).

Se crean con **( )** o solo **,** Si se crea con un solo elemento, hay que añadir una coma **,**

2. ¿Cuál es el orden de las operaciones?

Please – **P** – Paréntesis **( )**
Excuse – **E** – Exponentes ******
My – **M** – Multiplicación *****
Dear – **D** – División **/**
Aunt – **A** – Adición **+**
Sally – **S** – Sustración **-**

```
calculo = 8 + 2 * 5 – (9 + 2) ** 2

print(calculo) # -103
```

```
8 + 2 * 5 - (9 + 2) ** 2
8 + 2 * 5 - 11 ** 2
8 + 2 * 5 - 121
8 + 10 - 121
-103
```

3. ¿Qué es un diccionario en Python?

Es una estructura de datos que almacena información en pares de clave-valor (key-value). Es mutable, no ordenada (hasta Python 3.7+) y muy eficiente para buscar, insertar o eliminar datos. **Sintaxis:**

```
diccionario = {
“key1” = “value1”,
“key2 = “value2”
}
```

Características clave:

- **Claves únicas:** No puede haber duplicados (si repites una clave, se sobrescribe el valor).
- **Mutabilidad:** Puedes añadir, modificar o eliminar elementos después de crearlo.
- **Flexibilidad:** Las claves pueden ser strings, números o tuplas (si son inmutables), y los valores pueden ser cualquier tipo de dato (incluso otros diccionarios).

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

Sorted( ) función y **sort( )** método son usados para ordenar elementos pero tienen diferentes usos.

- **Sorted( )** Función: Devuelve una lista ordenada a partir de un iterable v (lista, tupla..), sin modificar la original. Devuelve una lista, incluso si el iterable no es una lista y no modifica el iterable original. Ej:

```
numeros = [2, 1, 3,]
numeros_ordenados = sorted(numeros) # Devuelve nueva lista

print(numeros)                # [2, 1, 3] original sin cambio
print(numeros_ordenados) # [1, 2, 3] ordenada
```

Uso: Necesidad de mantener el iterable original. Ordenar algo que no sea una lista (tupla, string).

- **sort( )** Método: Ordena la lista original directamente. Sólo funciona con tuplas, strings..). No retorna nada (None), pero modifica la lista original y es más eficiente en memoria para listas grandes sin crear copias. Ej:

```
numeros = [3, 2, 1]
```

numeros.sort() # Modifica lista original

print(numeros) # [1, 2, 3] cambió el original

Uso: Trabajas con una lista y no necesitas conservar el orden original. Ahorro de memoria (evita crear copias).

Iterable: Objeto que puede recorrerse elemento por elemento (listas, strings..)

5. **¿Qué es un operador de reasignación?**

(=) Es el operador que asigna un valor a una variable. Guarda (o reasigna) un dato en un espacio de memoria asociado a un nombre (variable). Significado: “almacenar en”.

Funcionamiento:

1. **Creación de la variable:**
2.

Cuando escribes **x = 10**

Reserva memoria para el valor 10.
Asocia el nombre x a ese espacio de memoria.

3. **Reasignación:**

Si después escribes **x = 20**

Crea el valor 20 en otro espacio de memoria.
Desvincula x de 10 y lo relaciona con 20.
El 10 queda “huérfano” y será eliminado por el recolector de basura si nadie más lo referencia.
Se puede acortar operaciones de asignación y calculo con operadores como **+=** , **-=** los que modifican el valor actual de la variable. Ej:

x = 5
x += 2   # Equivale a x = x+2 cambia a 7
x -= 2   # Equivale a x = x-2 cambia a 3

= (asignación): Guarda el valor en una variable: x = 5 # Asigna 5 a x

== (comparación): Verifica si dos valores son iguales.