BOTTEGA DEVCAMP. CHECKPOINT 4

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

Ambas son colecciones de datos que podemos crear para distintas situaciones. En la lista se crean con corchetes “[ ]”, mientras que en las tuplas se crean con “( )”. La principal diferencia es que una vez creada la colección, en el caso de las listas, los elementos son mutables, es decir, se pueden agregar, eliminar, reordenar sus elementos, etc. Sin embargo en el caso de las tuplas, una vez creadas no se permite la modificación de sus elementos. No se pueden reordenar, eliminar o agregar sus elementos.

Esta mutabilidad del caso de las listas y la inmutabilidad de las tuplas, es el comportamiento por el cual se usan en casos diferentes, donde necesitamos un comportamiento más flexible de sus elementos, o lo contrario en el caso de las tuplas.

Ejemplos:

```
# Ejemplo de lista
lista_compras = ["manzanas", "leche", "huevos"]
lista_compras.append("pan")
print(lista_compras)
# Resultado: ['manzanas', 'leche', 'huevos', 'pan']

# Ejemplo de tupla
lista_compras = ("manzanas", "leche", "huevos")
lista_compras.append("pan")
```

El código anterior provocará el siguiente error:
TypeError: 'tuple' object does not support item assignment

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

Cuando tenemos operaciones numéricas encadenadas con distintos operadores, se necesita seguir una regla de jerarquías en las operaciones para que matemáticamente den un resultado correcto. En el lenguaje Python se siguen exactamente las mismas reglas matemáticas las cuales son:

1. Paréntesis.
2. Potencias.
3. Productos.
4. Divisiones.
5. Sumas.
6. Restas.

Ejemplo:

```
resultado = (2 ** 3) * (4 / 2) + (3 - 1)  - 1
```

Primero se resuelven los paréntesis:

- 2 elevado a 3 = 8.
- 4 dividido por 2 = 2.
- 3 – 1 = 2.

resultado = 8 * 2 + 2 – 1

Luego potencias:

- 8 por 2 = 16.

resultado = 64 + 2 – 1

Sumas y restas: primero sumas y luego restas.

resultado = 65

3- ¿Qué es un diccionario en Python?

Un diccionario es otro tipo de colección donde podemos almacenar conjuntos de "clave-valor", es decir, poner un nombre de referencia y su valor correspondiente. Las claves son identificadores únicos que se utilizan para acceder a los valores asociados. Los valores pueden ser de cualquier tipo de dato en Python, como cadenas, números, listas e incluso otros diccionarios.

Características de los diccionarios:

- Ordenados: A partir de Python 3.7, los diccionarios mantienen el orden de inserción de los elementos.
- Mutables: Los diccionarios se pueden modificar después de su creación. Puedes agregar, eliminar o cambiar pares de clave-valor.
- Claves únicas: Cada clave en un diccionario debe ser única.
- Representación: Los diccionarios se escriben entre llaves {} y los pares de clave-valor se separan por comas. Cada par se escribe como clave:valor.

Por ejemplo:

```
top_3_ventas = {
        1 : 'Bad Bunny',
        2 : 'David Bisbal',
        3 : 'Pablo Alborán'
}
```

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

Los métodos sorted() y sort() son empleados para ordenar los elementos de una colección, aunque sort() sólo puede usarse para listas.

La diferencia fundamental es que sort() modifica la lista original, por lo que no hay que crear otra variable para almacenar el resultado. Sin embargo sorted(), reordena los elementos de la lista sin modificar la variable original, y si queremos almacenar los elementos ordenados, necesitamos crear otra variable.

Ejemplos:

```
lista_desordenada = [5, 1, 3, 4, 2]
# Usando sorted()
lista_ordenada_sorted = sorted(lista_desordenada)
print(lista_desordenada) # Salida: [5, 1, 3, 4, 2] (lista original sin cambios)
print(lista_ordenada_sorted) # Salida: [1, 2, 3, 4, 5] (nueva lista ordenada)

# Usando sort()
lista_desordenada.sort()
print(lista_desordenada) # Salida: [1, 2, 3, 4, 5] (lista original modificada)
```

5- ¿Qué es un operador de reasignación?

Los operadores de reasignación son utilizados para modificar el valor de una variable. Los hay que son simplemente un signo de igualdad “=”, y otros que unen un operador matemático con un nuevo signo de igualdad. Algunos ejemplos son:

```
total = 100             # Valor inicial = 100

total = total + 10      # total = 110 (suma 10 y reasigna a total)
total += 10             # total = 120 (expresión anterior abreviada)
total -= 10             # total = 110 (resta 10 y reasigna)
total *= 2              # total = 220 (multiplica por 2 y reasigna)
total /= 10             # total = 22 (divide entre 10 y reasigna)
total //= 10            # total = 2 (cociente entero de dividir entre 10 y reasignación)
total **= 2             # total = 4 (potencia de total elevado a 2, y reasignación)
total %= 2              # total = 0 (resto de la división de total entre 2 y reasignación)
```