

CHECKPOINT 4

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

Hay una diferencia fundamental entre una lista y una tupla, la modificación de los elementos que las componen. En el caso de la lista, podemos modificar los elementos que la componen así como añadir o eliminar dichos elementos. Por el contrario, una tupla no nos permite modificar, añadir o eliminar sus elementos.

Otra característica diferencial entre ambas es cómo hay que definir las, la lista se define utilizando corchetes [] y la tupla utilizando paréntesis ().

Ejemplo:

```
lista = [1, 2, 3]
tupla = (1, 2, 3)
```

En el caso de la lista, cuando queremos modificar uno de sus elementos tendríamos que hacerlo de la siguiente forma:

```
lista[0] = uno
```

En este caso concreto estaríamos cambiando el primer elemento de la lista '1' por 'uno'.

¿Cuál es el orden de las operaciones?

El orden de las operaciones en Python es el mismo que se utiliza en matemáticas. Por tanto, en caso de tener en una misma operación varios elementos como en el ejemplo que vemos a continuación, tendríamos que realizar en primer lugar el exponente (**), en segundo lugar la multiplicación (*) y la división (/) y en último lugar la suma(+) y la resta (-).

Ejemplo:

```
operacion_matematica= 5 + 2 * 3 ** 2 / 6 - 1
```

¿Qué es un diccionario Python?

Un diccionario Python es una estructura de datos que nos permite almacenar conjuntos de elementos, mejor dicho, la clave y el valor de diferentes elementos. Son útiles para almacenar y acceder a datos de manera eficiente, especialmente si queremos conocer el valor asociado a una clave específica.

Es importante que no haya dos claves iguales en el mismo diccionario, cada una tiene que ser única, pero los valores sí pueden repetirse en más de una ocasión y pueden ser desde cadenas, números hasta tuplas.

Para definir un diccionario tendremos que emplear llaves '{ }' y separar cada clave-valor con dos puntos ': '.

Ejemplo:

```
producto= {
    "nombre": "Vestido",
    "precio": 29.99,
    "disponibilidad": TRUE
```

```
}
```

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

En este caso estamos hablando de las variables 'sort()' y 'sorted()'. Dos variables que a simple vista parecen muy parecidas pero que tienen funciones diferentes.

La primera de ellas, la función 'sort()' es una función de ordenación que se utiliza para ordenar cualquier variable y nos devuelve una nueva lista ordenada sin modificar la lista original. Es decir, podríamos seguir utilizando ambas listas indistintamente. Por otro lado, la función 'sorted()' hace referencia al método de ordenación específico de las listas en Python. Esta función se utiliza para ordenar los elementos de la lista modificando la lista original.

Ejemplo:

```
lista_original = [5, 3, 2, 4, 1]
```

Si queremos tener una nueva lista con los elementos ordenados para poder seguir utilizando la anterior tendríamos que usar:

```
nueva_lista = sorted(lista_original)
```

Si por el contrario simplemente queremos tener esa lista ordenada tendríamos que utilizar:

```
lista_original.sort()
```

En el caso de la primera variable tendríamos tanto la 'lista_original' (5, 3, 2, 4, 1) como nueva_lista (1, 2, 3, 4, 5). En el caso de sort() tan solo tendríamos la 'lista_original' (1, 2, 3, 4, 5)

¿Qué es un operador de reasignación?

Un operador de reasignación en Python es un atajo que nos permite realizar una operación aritmética para modificar la variable original. Para ello tendríamos varias opciones dependiendo de la operación que queramos realizar: suma ('+='), resta ('-='), multiplicación ('*='), división ('/='), exponente ('**=') y módulo ('%='). Es decir, consiste en añadir antes del símbolo '=' el símbolo de la operación que queramos realizar.

Ejemplo:

```
ejemplo_reasignacion = 10  
ejemplo_reasignacion += 5
```

Si ejecutásemos print nos daría la suma de ambos elementos, es decir, 15.

```
print(ejemplo_reasignacion)  
15
```