Bottega DevCamp

Full Stack Developer

Checkpoint 4: Ejercicios teóricos

Fecha: 1 de marzo de 2025

Autor: Domínguez Becerril, Jorge DNI: 72607195J

Índice

1. Cuestiones teóricas

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

Una **lista** es un tipo de dato de Python que contiene un conjunto de cualquier tipo de dato. Se representa por corchetes, separando cada elemento con una coma y tiene la propiedad de ser **mutable**, es decir, sus elementos pueden ser modificados. Por otro lado, una **tupla** es un tipo de dato de Python que permite almacenar un conjunto de elementos de manera similar a una lista. Se representa por paréntesis, separando cada elemento con una coma. Sin embargo, a diferencia de una lista, es **inmutable**, esto es, sus elementos no pueden ser modificados.

1.2. ¿Cuál es el orden de las operaciones?

El orden de las operaciones en Python (y en cualquier otro lenguaje de programación) sigue el criterio PEMDAS. A continuación se listan las operaciones siguiendo el orden de más importante a menos importante:

- Paréntesis (**P**)
- Exponentes (**E**)
- Multiplicaciones (**M**)
- Divisiones (**D**)
- Sumas (**A**, de addition)
- Restas (**S**, de subtraction)

El nombre de este criterio proviene del inglés y está constituido por las iniciales de cada una de las operaciones.

1.3. ¿Qué es un diccionario Python?

Un diccionario de Python es un conjunto de elementos, cada uno de los cuales tiene una llave key y un valor value. Se representan con llaves, separando con una coma cada par. Al igual que las listas, tienen la propiedad de mutabilidad. En la figura 1 se muestra un ejemplo de un diccionario llamado **datos_personales** que contiene las llaves 'nombre', 'apellido', 'edad', 'sexo' y 'estudios' así como los valores 'Jorge', 'Domínguez', 27, 'hombre' y 'universitarios' asociados con cada una de las llaves.

```
datos_personales = {
    'nombre': 'Jorge',
    'apellido': 'Domínguez',
    'edad': 27,
    'sexo': 'hombre',
    'estudios': 'universitarios'
}
```

Figura 1: Ejemplo de diccionario en Python.

Para acceder a una valor concreto se debe escribir el nombre del diccionario seguido de unos corchetes con la llave asociada a ese valor. En el ejemplo anterior, para acceder a la cadena 'Jorge' se debe escribir datos_personales['nombre'].

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

Los métodos **sort**() y **sorted**() sirven para ordenar los elementos de una lista tanto alfabéticamente como numéricamente. La diferencia fundamental entre ambos es que el método **sort**() cambia la lista original y no devuelve ningún valor mientras que la función **sorted**() produce una nueva lista con los elementos ordenados.

```
lista = ['sandias', 'avellanas', 'cerezas', 'platanos', 'melocotones']

# Método sort()
print(lista.sort())

# Función sorted()
print(sorted(lista))
```

```
None
['avellanas', 'cerezas', 'melocotones', 'platanos', 'sandias']
```

Figura 2: Diferencia entre los métodos **sort**() y **sorted**().

En la figura 2 se puede apreciar esta diferencia. Este fragmento de código define una lista con varios elementos e imprime el resultado de utilizar las funciones **sort**() y **sorted**() sobre la lista. Nótese como el método **sort**() no produce ningún valor mientras que el método **sorted**() devuelve la lista ordenada. Ahora bien, si se imprime la lista original se verá que está ordenada alfabéticamente, tal y como cabe esperar de la función **sort**(). En la figura 3 se puede ver el resultado.

```
lista = ['sandias', 'avellanas', 'cerezas', 'platanos', 'melocotones']

# Método sort()
print(lista.sort())

# Función sorted()
print(sorted(lista))

# Lista original
print(lista)
```

```
None
['avellanas', 'cerezas', 'melocotones', 'platanos', 'sandias']
['avellanas', 'cerezas', 'melocotones', 'platanos', 'sandias']
```

Figura 3: Diferencia entre los métodos **sort**() y **sorted**().

1.5. ¿Qué es un operador de reasignación?

Un operador de reasignación es un tipo de operador que se usa para modificar el valor de una variable existente en lugar de asignarle un nuevo valor. Existen varios tipos de operadores de reasignación:

- Suma y asignación, denotado por +=
- Resta y asignación, denotado por -=
- Multiplicación y asignación, denotado por *=
- División y asignación, denotado por /=
- División entera y asignación, denotado por //=
- Potencia y asignación, denotado por **=
- Módulo y asignación, denotado por %=

En la figura 4 se muestra un ejemplo para el operador de suma y asignación. Inicialmente se tiene la variable **total** con un valor de 1000. Se desea sumar 200 al valor inicial, de manera que se utiliza el operador de reasignación +=.

```
total = 1000
total += 200
print(total)
```

```
1200
```

Figura 4: Ejemplo de suma y asignación.