Módulo 2 checkpoint 4: Preguntas teóricas

Pregunta 1

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

La primera diferencia es la sintaxis, las listas se declaran con corchetes [] y las tuplas se declaran con parentesis (), a partir de esta diferencia existe una diferencia de uso, las listas son mutables, por lo tanto nos permiten su modificación en la lista original, y las tuplas no lo son, son inmutables, lo cual no nos permite su modificación tal como las cadenas, pero si hay formas de realizar esta modificaciones, incorporar elementos, pero siempre sin modificar la tupla original.

La diferencia de uso se hace evidente cuando utilizamos el desempaquetado, que nos ofrece la manera de tratar como una variable a los elementos dentro de la lista y la tupla, el desempaquetado en una tupla nos ofrece la seguridad de que sus elementos no han cambiado, ni han modificado su orden, sin embargo el desempaquetado en una lista nos puede llevar a un error, dado que al ser modificables han podido cambiar el orden o el elemento de la lista original.

Pregunta 2

¿Cuál es el orden de las operaciones?

El orden de operaciones es el orden en el que python realizara los calculos, es sumamente importante ya que si no se tiene claro el orden, podemos obtener resultados de nuestras operaciones no esperados.

Como siglas para recordar se utiliza PEMDAS, aunque tambien exite PEDMAS, estas siglas nos recuerdan que el orden sería P de parentesis, E de exponentes, M de multiplicación, D de división, A de addicción o suma y S de sustracción o resta. El cambio de orden en la multiplicación o división M por D o D por M no tiene resultado diferente.

Como ejemplo del orden sería:

```
var_pemdas = 1 + 1 * 10 / 2 - (10 * 2)

El orden que seguirían las operaciones sería:

- parentesis (10 * 2) = 20 == 1 + 1 * 10 / 2 - 20
- multiplicacion 1 *10 = 10 == 1 + 10 / 2 - 20
- division 10 / 2 = 5 == 1 + 5 - 20
- suma 1 + 5 = 6 == 6 - 20
- resta 6 - 20 == -14
```

Pregunta 3

¿Qué es un diccionario Python?

Es una estructura de datos que podemos asignar a una variable a traves de la siguiente sintaxis, nuevo diccionario = {}, esta seria la sintaxis para crear un diccionario vacio. Al igual que las listas, el dicionario contiene elementos, en este caso los elementos estan estructurados como clave = valor, lo cual quiere decir, que para cada clave se le otorga un valor correspondiente, este valor puede ser de cualquier tipo, entero, flotante, cadenas incluso listas o tuplas, a más incluso otros diccionarios.

Al trabajar con varios elementos, las mejores prácticas serian ofrecerle varias lineas, ya que como he comentado pueden incluir listas, tuplas o diccionarios, por lo tanto, crearan varios elementos y puede ser complicado y tedioso comprender completamente el diccionario.

Ejemplo:

```
new_dictionary = {
    'name_1' : 'valor1',
    'name_2' : ['valor1', 'valor2', 'valor3'],
    'name_3' : ('valor1', 'valor2')
}
```

Podemos extraer todos las claves, todos los valores, utilizando las enseñanzas que he aprendido con listas, el manejos de esos elementos es igual en ambas estructuras, desmenuzar la complejidad del diccionario, utilizando cada elemento, te lleva a que esta biblioteca de elementos pueda ser un gran almacen de información.

Ejemplo:

Obtener la clave y valor del segundo valor de la segunda clave del diccionario.

```
new_exemple_key_second = list(new_dictionary.keys())[1]
new_exemple_value_second = list(new_dictionary.values())[1][1]
print(new_exemple_key_second)
print(new_exemple_value_second)
```

Pregunta 4

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

La método ordenado sort(), que tienen las listas de python ordena de menor a mayor los números, y alfabeticamente las cadenas, y una diferencia fundamental es que sort() modifica la cadena original in situ, osea la cambia de orden, podemos incluso pasarle el parametro reverse=True que nos devolveria el orden al reves, osea numeros de mayor a menos y cadenas desde la z a la a.

Ejemplo:

```
print()
first_list_number = [10, 40, 20, 153, 225]
first_list_string = ['casa', 'arbol', 'barco','peine']
print('List number: {}'.format(first_list_number))
```

```
print('List string: {}'.format(first_list_string))
print()
first_list_number.sort()
first_list_string.sort()
print('List number ordered: {}'.format(first_list_number))
print('List string ordered: {}'.format(first_list_string))
print()
first_list_number.sort(reverse=True)
first_list_string.sort(reverse=True)
print('List number reverse: {}'.format(first_list_number))
print('List string reverse: {}'.format(first_list_string))
```

Y la diferencia fundamental con la función de ordenación sorted() es que sorted(), no modifica la lista original crea una nueva y realiza sus acciones sobre la nueva lista, manteniendo intacta la lista original, tambien le podemos pasar el parametro reverse=True lo que nos devuelve las listas ordenadas al reves.

Ejemplo:

```
print()
first_list_number = [10, 40, 20, 153, 225]
first_list_string = ['casa', 'arbol', 'barco','peine']
print('List number: {}'.format(first_list_number))
print('List string: {}'.format(first_list_string))
print()
new_list_number_sorted = sorted(first_list_number)
new_list_string_sorted = sorted(first_list_string)
print('List number ordered: {}'.format(new_list_number_sorted))
print('List string ordered: {}'.format(new_list_string_sorted))
print('List number: {}'.format(first_list_number))
print('List string: {}'.format(first_list_string))
print()
new_list_number_sorted = sorted(first_list_number, reverse=True)
new_list_string_sorted = sorted(first_list_string, reverse=True)
print('List number reverse: {}'.format(new_list_number_sorted))
print('List string reverse: {}'.format(new_list_string_sorted))
print('List number: {}'.format(first_list_number))
print('List string: {}'.format(first_list_string))
```

Pregunta 5

¿Qué es un operador de asignación?

El operador básico de asignación es = , nos permite asignar un valor a una variable, luego tenemos los atajos que son operadores de asignación que nos permiten realizar un calculo mientras realizamos la asignación, su sintaxis es identica a los operadores estandar.

Como ejemplo serían:

total = 10 (En esta expresión esta el operador de asignación =)

Si quisieramos realizar la siguiente operación:

total = total + 333

Esperando el resultado de: 343, podemos realizar un atajo, utilizando los operadores de asignación aprendidos:

total += 333

Esta expresión daría como resultado 343 al ser la misma, el operador += sustituye a total +.

La misma operación se puede realizar con los operadores de resta, multiplicación, división, división por piso, exponente y modulo