Checkpoint 4 – M2C4 Python Assignment

Tarea 1 - Ejercicios en Python.

Los ejercicios en Python de este Checkpoint 4 se encuentran bajo el link a mi GitHub:

https://github.com/aatxaerandio/DevCamp/blob/bd2d153464ef9ada04e285bf754193e92b034b56/Full-Stack/module_02_python/07_checkpoint_4.py

Tarea 2 – Preguntas teóricas

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

Las listas y las tuplas son un tipo de datos que son bastante similares en cuanto a que pueden contener números, cadenas, etc.

La principal diferencia entre las litas y las tuplas es que mientras que las listas son mutables, las tuplas son inmutables. Es decir, las listas son elementos mutables, que pueden ser cambiadas o modificadas. Por el contrario, las tuplas son elementos inmutables, y no pueden ser modificadas como las listas.

Por tanto, si a la hora de desarrollar aplicaciones hay datos que no van a cambiar o ser modificados, se deberían usar las tuplas en lugar de las listas. Además de ello, hay que tener en cuenta que la sintaxis es diferente, usando paréntesis () en las tuplas, mientras que corchetes [] en las listas.

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

Cuando hablamos del orden de las operaciones hablamos de qué operación se tiene que ejecutar en primer orden y cuales le siguen. El orden de las operaciones en Python sigue las reglas matemáticas. Por tanto, primero se ejecuta lo que está dentro de los paréntesis, para continuare con los exponentes, multiplicaciones, divisiones, y finalmente las sumas y restas. En el caso de las multiplicaciones y divisiones, o las sumas y restas, siempre se empieza de izquierda a derecha.

En caso de duda y con el fin de acordarse fácilmente del orden de las operaciones se puede recordar esta frase: “**P**lease **E**xcuse **M**y **D**ear **A**unt **S**ally”. Cada primera letra de cada palabra correspondería al orden de operaciones, de este modo; “**P**arenthesis”, “**E**xponents”, “**M**ultiplication”, “**D**ivision”, “**A**ddition”, “**S**ubstraction”.

3. **¿Qué es un diccionario Python?**

Un diccionario es una colección que datos, creado por “llaves” (key) y “valores” (values). Las llaves contienen los valores de las mismas, y pueden ser números, cadenas, etc.

La particularidad de los diccionarios es que son elementos que presentan un orden determinado y su contenido es modificable. En cuanto a la sintaxis, los diccionarios están recogidos por corchetes redondeados {}. Las llaves y los valores se escriben entre comillas simples o dobles. Por otro lado, las llaves de delimitan con dos puntos : y los calores con comas ,

Un ejemplo de diccionario es:

```
diccionario_uno = {
    "llave_01": "valor_01",
    "llave_02": "valor_02",
    "llave_03": "valor_03",
}

Imprimimos con print(diccionario_uno)
El resultado sería; {'llave_01': 'valor_01', 'llave_02': 'valor_02',
'llave_03': 'valor_03'}
```

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

   El método ordenado y la función de ordenación, como su nombre indica, es para ordenar los datos que contienen ciertos tipos de elementos.

   Para empezar, la sintaxis del método ordenado es sort(), mientras que el de la función de ordenación será sorted().

   La diferencia entre ambas es que si se usa sort(), se estará cambiando la lista, es decir, se ordena y se mantiene asi.

```
lista_clase = [
  "Miguel",
  "Juanjo",
  "Alejandro",
]
print(lista_clase)    # ['Miguel', 'Juanjo', 'Alejandro']
```

   El orden sería el que está por defecto, tal cual está escrita en la lista.

```
lista_clase.sort()
print(lista_clase)      #['Alejandro', 'Juanjo', 'Miguel']
```

Si se aplica sort(), se reordena alfabeticamente en este caso. **Y de este modo se ha cambiado el orden de la lista.**

```
lista_ordenada = lista_clase.sort()
print(lista_ordenada)    # None
```

Lo que ocurre con sort() es que en este caso no podemos guardarlo en una variable, y por tanto, al imprimir sale "None".

En este caso es muy útil usar la función sorted(), ya que permite tomar una copia ordenada pero no modifica la lista.

```
ista_ordenada = sorted(lista_clase)
print(lista_ordenada) # ['Alejandro', 'Juanjo', 'Miguel']

print(lista_clase) # ['Miguel', 'Juanjo', 'Alejandro']
```

5. ¿Qué es un operador de reasignación?

Un operador de reasignación en python hace referencia a la reasignación de valores a variables. Hay varios tipos de operadores de reasignación, de suma, resta, multiplicación, división, etc. Se usan tras declarar una variable y antes del signo igual (=). De este modo se declara que la variable será el resultado del efecto del operador a lo que contiene dicha variable. Por ejemplo, usando un operador de suma;

Dada una variable,

manzanas = 10

```
manzanas += 5
```

Se usa el operador suma +, de este modo **se estará reasignando la variable manzana**, añadiéndole el valor de 5. Así, el valor se ha reasignado y al imprimir son 15.

```
print(manzanas)      # 15
```

Continuando con el actual valor de la variable manzanas, si por ejemplo se usa el operador de multiplicación:

```
manzanas *= 3
```

El valor de la variable se reasigna, en este caso, 3 x 15 = 45.

```
print(manzanas)      # 45
```