M2C4 Python Assignment – Preguntas teóricas

Las preguntas teóricas son:

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

La principal diferencia es que una tupla es un elemento inmutable como lo son las cadenas (‘*strings*’), es decir, no se puede alterar su contenido original o su estructura de datos, mientras que las listas son mutables. De este modo, los operadores para ordenar, añadir, eliminar o obtener items funcionarán en el caso de las listas pero no así en el caso de las tuplas. La sintaxis para la creación listas (que pueden nombrarse como '*arrays*' en otros lenguajes de programación) conlleva el uso de corchetes, mientras que las tuplas usan paréntesis. En ambos casos, la sintaxis para crear un conjunto de elementos usa cadena ('*string*') separado por coma. Uno de los atributos interesantes en las tuplas es la función de ‘*unpacking*’ que permite asignar un nombre a todos los elementos de la tupla (título, subtítulo, contenido) y usarlos como si se tratase de un índice en una lista.

¿Cuál es el orden de las operaciones?

El orden de las operaciones matemáticas en Python está dado por el acrónimo PEMDAS (en inglés) similar al orden matemático y que como ayuda mnemotécnica puede expresarse como sigue:

PEMDAS:

Please --> **Parentheses** = Paréntesis ()

Excuse --> **Exponents** = Exponentes **

My --> **Multiplication** = Multiplicación *

Dear --> **Division** = División /

Aunt --> **Addition** = Suma +

Sally --> **Subtraction** = Resta -

¿Qué es un diccionario Python?

Un diccionario es un elemento que almacena datos en pares del tipo clave:valor. Podemos crear no solo elementos como se hace con las listas, sino que también podemos crear una clave con un valor correspondiente. La sintaxis para crear un diccionario en Python es usar

llaves y dentro de estas listar la colección de datos del tipo clave+valor, ambos como elementos cadena (‘*strings*’) separados por dos puntos, por ejemplo un par clave valor es la posición de un jugador de fútbol y su apellido ‘DE’ : ‘Silva’, las diferentes entradas de pares clave+valor se separan por comas, así:

```
NK_Olimpija  = {
    'DF': 'Silva',
    'FW': 'Nukić',
    'MF': 'Elšnik',
    'GK': 'Vidovšek'
}
```

El diccionario se almacena en una variable para trabajar posteriormente con él, en el caso del ejemplo esta variable es “NK_Oljmpija” que es el nombre del equipo de fútbol.

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

La diferencia principal entre el método ordenado (***sort()***) y la función de ordenación (***sorted()***) en Python está en cómo se aplican y qué efectos tienen en las listas. El método ordenado (***sort()***) es un método que se usa en las listas de Python para ordenar la lista en su lugar, es decir, modifica la lista original.Este método ordena la lista original directamente y no devuelve nada (o más precisamente, devuelve ‘*None*’), muestra el resultado directo de la ordenación. Por su parte la función de ordenación (***sorted()***) es una función incorporada en Python que toma una secuencia (por ejemplo, una lista) y devuelve una nueva lista ordenada que contiene los elementos de la secuencia original en orden ascendente o descendente. La función *sorted()* no modifica la lista original; en su lugar, devuelve una nueva lista ordenada la cual puede almacenar en una nueva variable para usarla después en mi código.

Sort:

```
# Sort list

tags = ['python', 'development', 'tutorials', 'code']

print(tags) # ['python', 'development', 'tutorials', 'code']

tags.sort() #Alphabetical order

print(tags) # ['code', 'development', 'python', 'tutorials']
```

Sorted:

```python
# Sorted function
sale_prices = [
  100,
  83,
  220,
  40,
  100,
  400,
  10,
  1,
  3
]

sorted_list = sorted(sale_prices, reverse=True)

print(sorted_list) # [400, 220, 100, 100, 83, 40, 10, 3, 1]
print(sale_prices) # [100, 83, 220, 40, 100, 400, 10, 1, 3]
```

¿Qué es un operador de reasignación?

Un operador de reasignación se utiliza para actualizar o modificar el valor de una variable existente mediante una operación aritmética combinada con la reasignación del resultado a la misma variable. Esto proporciona una forma concisa de actualizar el valor de una variable basándose en su valor actual. El operador de reasignación más común es el operador de asignación aumentada (+=, -= , *= , /= , %=, **=, //=), por ejemplo:

```python
# Reassignation operator
post = ('Python Basics', 'Intro guide to python', 'Some cool python content',
'published')
print (post)

post += ('tags',)

first, second, third, fourth, fifth = post

print(first)
print(second)
print(third)
print(fourth)
print(fifth)
```