M2C4 TEORÍA

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

La principal diferencia entre una lista y una tupla es que una lista se puede modificar y una tupla no. La primera es mutable se puede transformar, es decir ampliar, reducir y también ordenar y la segunda no al menos no como la lista.

Ambas tanto listas como tuplas son conjuntos o colecciones de datos pueden estar formados por cadenas, integar, floats. Las tuplas su longitud es fija, pero las listas su tamaño puede ser variable ya que se pueden modificar, aumentar y disminuir el número de elementos que la componen

Ejemplos:

Lista

directors = ['lang', 'villeneuve', 'hitchtcock', 'donen', ‘nolan’]

Tuple

actors = ('gable', 'grant', 'power', 'firth', ‘holden’)

Otra diferencia es referente a su sintaxis las listas usa corchetes y las tuplas paréntesis. A continuación indico alguns de las operaciones/funciones y métodos que transforman las listas.

Las listas pueden utilizar métodos para ser ordendas como son sort() y .sorted() y para ser ampliadas append() extend() , para quitar elementos remove() pop(). Las tuplas no pueden ser modificadas ni ordenadas son inmutables. (se puede utilizar la función .sorted() pero sigue mantiene la tupla original). La gran diferencia entre ambas es que la tupla es inmutable y la lista como he indicado anteriormente no.

Ejemplos

Lista

directors = ['lang', 'villeneuve', 'hitchtcock', 'donen', ‘nolan’]

directors.append(‘eastwood’)

print (directors) ['lang', 'villeneuve', 'hitchtcock', 'donen', 'eastwood']

Lista

```
directors = ['lang', 'villeneuve', 'hitchtcock', 'donen']
directors.extend ('eastwood')
print(directors) ['lang', 'villeneuve', 'hitchcock', 'donen', 'e', 'a', 's', 't', 'w', 'o', 'o', 'd']
```

Lista

```
directors = ['lang', 'villeneuve', 'hitchtcock', 'donen']
directors.remove ('hitchtcok')
print(directors) ['lang', 'villeneuve', 'donen']
```

Las utilización de uno u otro de tipo de colección o agrupación de datos dependenerá de la finalidad del trabajo a realizar. Es una ventaja que la tupla no se pueda modficar ya que permite que el contenido no se cambie, si se quiere manipular se puede convertir en una lista y trabajar con esa nueva lista, pero conservado la tupla original.

2. ¿Cuál es el orden de las operaciones?

El orden de las operaciones siguen el orden del acronimo **PEDMAS** ó también se puede usar tip nemotécnico de **P**lease, **e**xcuse **m**y **d**ear **a**unt **S**ally. Es decir, que el orden sería

1. Paréntesis
2. Exponenentes
3. Multiplicaciones
4. División
5. adición (suma)
6. sustracción (restas)

Ejemplos

tag= 15+16*(25+36)+26-43**3 /25 ==> tag= 15+16*(61)+26-79.507/25

print (tag)

-2163,28

3. ¿Qué es un diccionario Python?

Es una colección, un conjunto de datos que están agrupados por parejas es decir, clave o Key y valor.

La sintaxis correspondiente a ambos es clave y valor están entre comillas y separados por dos puntos. El final y el principio del diccionario , es decir se abre y se cierra con llaves {}

Ejemplo:

```
dictionary = { "table" : "mesa" , "chair":"silla" , "Lamp" :"lampara", "door" :"puerta" }
```

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

La diferncia entre .sort() y .sorted().

El primero crea una nueva lista ordenada puede ser en orden creciente o decreciente al igual que alfabeticamente que puede ser ascedente o descendente ; esto puede ocasionar que sea más lento y consuma más memoria; en cambio .sorted () crea una nueva lista o colección de datos es decir puede ser utilizado en listas, tuplas, diccionarios. y conserva el conjunto o colección original

Ejemplos:

```
numers = [45,26,23,89,15]
numers.sort()
print(numers)  [15, 23, 26, 45, 89]
```

```
numers = [45,26,23,89,15]
numers.sort()
print(numers)  [15, 23, 26, 45, 89]
```

```
directors = ['lang', 'scott', 'hitchtcock', 'donen']
directors.sort()
print(directors)  ['donen' ,'hitchcock','lang', 'scott']
```

```
actors = ('gable', 'pacino', 'bardem', 'isaac')
actors_sorted= sorted(actors)
print(actors_sorted) ['bardem' ,'gable','isaac', 'pacino']
```

5. ¿Qué es un operador de reasignación?

El operador de asignación = nos permite directamente una serie de operaciones como sumar o añadir un elmento "+=" ó realizar una operación aritmética este es el caso de "*=", "/=", "%="

Ejemplos

```
x=8
y+=1
print(x)  9

y=10
y-=1
print(y)  9

z=30
z/=3
print (z) 10,0

m=25
m*5
print (m) 125
```