Tuplas

Una tupla es similar a una lista, una estructura de datos ordenada para almacenar multiple información, permite valores duplicados, la diferencia es que no puede modificada, es decir es inmutable.

A estas, y otras, estructuras se les conoce tipo de secuencia, que por definición es un tipo de dato en Python el cual es capaz de almacenar más de un valor (o ninguno si la secuencia esta vacía), los cuales pueden ser secuencialmente (de ahí el nombre) examinados, elemento por elemento.

Y como se ha visto hasta el momento una secuencia es un tipo de dato que puede ser escaneado por el bucle for.

Del primer párrafo se resalta el término mutabilidad que por definición es una propiedad de cualquier tipo de dato en Python que describe su disponibilidad para poder cambiar libremente durante la ejecución de un programa.

Si un tipo de dato es mutable, puede ser modificado libremente en cualquier momento, si es inmutable, no pueden ser modificados directamente.

Las tuplas pueden ser accedidas igual que las listas y almacenar diferentes tipos de datos mezclados, manejar índices negativos, rangos de índices y demás.

Lo primero que distingue una lista de una tupla es la sintaxis empleada para crearlas - las tuplas utilizan paréntesis (), mientras que las listas usan corchetes []. Aunque también pueden ser creadas solo indicando comas ,. Cada elemento de la tupla puede ser de un tipo de dato diferente.

```
tuple_1 = (1, 2, 4, 8)

tuple_2 = 1., .5, .25, .125
```

Aquí puede encontrar el listado de métodos de las tuplas

```
[ ]  tuple_1 = (1, 2, 4, 8)
     tuple_2 = 1., .5, .25, .125

     print(tuple_1)
     print(tuple_2)
```

```
(1, 2, 4, 8)
(1.0, 0.5, 0.25, 0.125)
```

Ejercitación Tupla vacía

Cómo puede crear:

- Una tupla vacía
- Una tupla de un solo elemento

Debido a su naturaleza inmutable pero su similitud con listas, para modificar una tupla lo que se pude hacer es convertirla a lista, aplicar métodos de listas y nuevamente convertirla a una tupla.

```
convertir tuppla a lista

aplicar cambios

convertir lista a tupla
```

```
[ ]  #Partiendo de que los estudiantes del curso no van a cambiar
     #Es posible crear una tupla con el listado
     estudiantes_tupla = ('Juan', 'Pedro', 'Andrés', 'María', 'Johanna')


     print(estudiantes_tupla)
     estudiantes_temp_list = list(estudiantes_tupla);
     print(estudiantes_temp_list)

     estudiantes_temp_list[1] = 'Cesar'
     print(estudiantes_temp_list)

     estudiantes_tupla = tuple(estudiantes_temp_list)
     print(estudiantes_tupla)
```

```
('Juan', 'Pedro', 'Andrés', 'María', 'Johanna')
['Juan', 'Pedro', 'Andrés', 'María', 'Johanna']
['Juan', 'Cesar', 'Andrés', 'María', 'Johanna']
('Juan', 'Cesar', 'Andrés', 'María', 'Johanna')
```

Ejercitación Tuplas o listas?

- Se ha mencionado en diferentes momentos, la similitud entre listas y tuplas, en ese orden de ideas, cómo puede acceder a los elementos almacenados en una tupla? por ejemplo al segundo elemento de la tupla anterior.

- Qué sucede con las siguientes instrucciones

```
my_tuple = (1, 10, 100, 1000)


my_tuple.append(10000)
del my_tuple[0]
my_tuple[1] = -10
```

- Cómo puede utilizar los métodos in y not in?
- Qué ocurre al utilizar el operador + entre tuplas?
- Qué ocurre al utilizar el operador * entre una tupla y un entero?

Las tuplas soportan algo llamado el desempaquetado, que consiste en asignar los valores de la tupla a nuevas variables, puede ser uno a uno, es decir, una variable por cada elemento en la tupla, o en una sola variable almacenar diferentes valores, lo que convierte dicha variable en una lista.

```
[ ]  juan, cesar, andres, *mujeres = estudiantes_tupla

     print(juan)
     print(mujeres)
```

```
Juan
['María', 'Johanna']
```

Ejercitación Multiplicando tuplas

- Cree una tupla con 3 elementos cada una y haga la multiplicación elemento a elemento entre las dos tuplas, es decir tupla_1[0] x tupla_1[0]. El resultado de cada operación debe ser almacenado en una nueva estructura que estará ordenada de menor a mayor.

- Qué hacen los métodos index() y count()?