Tupla (tuple)

Las tuplas en Python son un tipo o estructura de datos que permite almacenar datos de una manera muy parecida a las listas, con la salvedad de que son inmutables.

Crear tupla Python

Las tuplas en Python o `tuples` son muy similares a las listas, pero con dos diferencias. Son **inmutables**, lo que significa que no pueden ser modificadas una vez declaradas, y en vez de inicializarse con corchetes se hace con `()`. Dependiendo de lo que queramos hacer, **las tuplas pueden ser más rápidas**.

```
tupla = (1, 2, 3)
print(tupla) #(1, 2, 3)
```

También pueden declararse sin `()`, separando por `,` todos sus elementos.

```
tupla = 1, 2, 3
print(type(tupla)) #<class 'tuple'>
print(tupla)       #(1, 2, 3)
```

Operaciones con tuplas

Como hemos comentado, las tuplas son tipos **inmutables**, lo que significa que una vez asignado su valor, no puede ser modificado. Si se intenta, tendremos un `TypeError`.

```
tupla = (1, 2, 3)
#tupla[0] = 5 # Error! TypeError
```

Al igual que las listas, las tuplas también pueden ser anidadas.

```
tupla = 1, 2, ('a', 'b'), 3
print(tupla)       #(1, 2, ('a', 'b'), 3)
print(tupla[2][0]) #a
```

Y también es posible convertir una lista en tupla haciendo uso de al función `tuple()`.

```
lista = [1, 2, 3]
tupla = tuple(lista)
print(type(tupla)) #<class 'tuple'>
print(tupla)       #(1, 2, 3)
```

Se puede **iterar** una tupla de la misma forma que se hacía con las listas.

```
tupla = [1, 2, 3]
for t in tupla:
    print(t) #1, 2, 3
```

Y se puede también asignar el valor de una tupla con `n` elementos a `n` variables.

```
l = (1, 2, 3)
x, y, z = l
print(x, y, z) #1 2 3
```

Aunque tal vez no tenga mucho sentido a nivel práctico, es posible crear una tupla de un solo elemento. Para ello debes usar `,` antes del paréntesis, porque de lo contrario `(2)` sería interpretado como `int`.

```
tupla = (2,)
print(type(tupla)) #<class 'tuple'>
```

Métodos tuplas

`count(<obj>)`

El método `count()` cuenta el número de veces que el objeto pasado como parámetro se ha encontrado en la lista.

```
l = [1, 1, 1, 3, 5]
print(l.count(1)) #3
```

`index(<obj>[,index])`

El método `index()` busca el objeto que se le pasa como parámetro y devuelve el índice en el que se ha encontrado.

```
l = [7, 7, 7, 3, 5]
print(l.index(5)) #4
```

En el caso de no encontrarse, se devuelve un `ValueError`.

```
l = [7, 7, 7, 3, 5]
#print(l.index(35)) #Error! ValueError
```

El método `index()` también acepta un segundo parámetro opcional, que indica a partir de que índice empezar a buscar el objeto.

```
l = [7, 7, 7, 3, 5]
print(l.index(7, 2)) #2
```