Tuplas

Una tupla es un tipo de datos de Python que permite almacenar varios valores baj Hay varias formas de crear una tupla:

In [6]:
```
# Se definen diferentes tuplas
tupla1 = 0, 1, 2, 3, 4
tupla2 = (0, 1, 2, 3, 4)
tupla3 = tuple(range(0,5))
# Se imprimen las tuplas
print(tupla1)
print(tupla2)
print(tupla3)
# De que tipo de datos es la variable "tupla1"?
type(tupla1)
```

```
(0, 1, 2, 3, 4)
(0, 1, 2, 3, 4)
(0, 1, 2, 3, 4)
```

```
tuple
```

Dato curioso: ¿como hacer una tupla de un solo elemento?

```
t = 1    -> Esto es una variable int
t = (1)  -> Esto sigue siendo una variable int
```

Se puede hacer esto con una sintaxis que puede parecer un error:

In [10]:
```
tupla4 = (1,)
type(tupla4)
```

```
tuple
```

La ',' adicional al final indica que ese dato es una tupla y no un valor entero.

Para acceder a los datos de una tupla se utiliza el *índice* que indica la posición de cada dato. El indice inicia con cero, por lo que:

```
tupla = (10, 20, 30)
tupla[0] -> 10
tupla[1] -> 20
tupla[2] -> 30
```

Los índices pueden ser negativos. El índice -1 indica el último valor, el -2, en penúltimo, etc.

```
tupla[-1] -> 30
tupla[-2] -> 20
tupla[-1] -> 10
```

Por ejemplo, se pueden imprimir los datos de un tupla utilizando un lazo for que controle los índices:

In [24]:

```
tupla1 = 10, 20, 30, 40, 50
for i in range(len(tupla1)):
    print(tupla1[i])
```

```
10
20
30
40
50
```

La funcion *len()* retorna el número de datos de un iterable (en este caso, de una tupla).

Sin embargo, una tupla es un *iterable*, por lo que puede ser parte de un for de forma directa:

In [25]:

```
for num in tupla1:
    print(num)
```

```
10
20
30
40
50
```

Para imprimir tanto los índices como los datos se puede utilizar la funcion *enumerate()* que por cada elemento de un iterable, extrae el índice y el dato en si, en dos variables (a esto se le llama desempaquetado de un iterable).

In [26]:

```
for i, v in enumerate(tupla1):
    print("indice =", i,"\tvalor =",v)
```

```
indice = 0      valor = 10
indice = 1      valor = 20
indice = 2      valor = 30
indice = 3      valor = 40
indice = 4      valor = 50
```

Una tupla es "*inmutable*", esto quiere decir que no puede cambiar:

In [27]:

```
tupla1[0] = 0
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-cd68bbe4f592> in <module>()
----> 1 tupla1[0] = 0

TypeError: 'tuple' object does not support item assignment
```

Una tupla puede contener datos de tipos combinados:

In [29]:

```
tupla1 = (2000, 3.1415, 'Nombre')
for data in tupla1:
    print(data)
```

```
2000
3.1415
Nombre
```

¿Qué funciones se pueden aplicar sobre una tupla? Se puede consultar esto por medio de la instrucción dir():

In [30]:

```
tupla1 = (1, 2, 3)
dir(tupla1)
```

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'count',
 'index']
```

Como se observa básicamente son 2:

```
count(val)                    -> Cuenta cuantos elementos de un valor existen en una tupla
index(val,[inicio, [fin]])    -> Retorna el indice donde se enuentre un dato (entre inicio y fi
n)
```

In [36]:

```
tupla = (10, 11, 12, 13, 14, 15, 14, 13, 12, 11, 10)
print("¿Cuantos 13's hay en la tupla?:", tupla.count(13))
print("¿Donde estan ubicados los 13's de la tupla?", tupla.index(13))
```

```
¿Cuantos 13's hay en la tupla?: 2
¿Donde estan ubicados los 13's de la tupla? 3
```

Si hay dos ocurrencias iguales de un dato, la función index() solo retornará el índice de la primera ocurrencia. Sin embargo se puede utilizar un lazo que ajuste el rango de busqueda de la función index() y que realice tantas iteraciones como ocurrecias haya de un dato (revise el siguiente código y trate de interpretarlo):

In [40]:

```
ini = 0
for i in range(tupla.count(13)):
    index13 = tupla.index(13, ini)
    print("Indice 13 =", index13)
    ini = index13 + 1
```

```
Indice 13 = 3
Indice 13 = 7
```

¿Por que utilizar una tupla? Hay varias razones:

- Si se desea tener datos y no se quiere que estos sean modificados
- Si se desea procesar información y tener un mejor tiempo de respuesta.

Sobre esto último, hagamos una prueba (estas dos celdas de instrucciones pueden tomar algunos segundos):

In [5]:

```
timeit "x = tuple(range(100))"
```

```
10000000 loops, best of 3: 11.8 ns per loop
```

In [6]:

```
timeit "x = list(range(100))"
```

```
100000000 loops, best of 3: 15.7 ns per loop
```

La instruccion "*timeit*" realiza una instrucción (en este caso, la creacion de una tupla y de una lista) 1'000,000 de veces y toma el mejor tiempo de ejecución. Se observa que la creación de una lista es 3 ns más rápida. Puede parecer poca cosa, pero si se tiene una gran cantidad de datos (¡y una gran cantidad de realmente una gran cantidad!) se puede apreciar alguna diferencia.

Sin embargo, para un numero de datos que no sea mounstruoso, con una lista es suficiente. ¿Y que es una lista? Es básicamente una tupla con esteroides...