ESTUDIOS SUPERIORES DE BUENOS AIRES

ESBA

Tuplas

Tuplas

Son una secuencia de valores agrupados.

Una **tupla** sirve para agrupar, como si fueran un único valor, varios valores que, por su naturaleza, deben ir juntos.

Las tuplas son inmutables, es decir que **no puede ser modificada** una vez que ha sido creada.

Son más eficientes que las listas para algunas operaciones, especialmente cuando se trata de recorrer elementos y acceder a ellos.

Para poder crear las mismas, se deben asignar a la variable los valores separados por comas y entre paréntesis. Por ejemplo, podemos crear una tupla que tenga el apellido y nombre de una persona:

```
persona = ('Guido', 'van Rossum')
```

En otros lenguajes, las tuplas reciben el nombre de **registros**.

Tuplas

También es posible crear una tupla sin el uso de los paréntesis, es decir, solo separando los valores por coma. Sin embargo, para aportar mayor claridad al código, es preferible hacer uso de los mismos.

```
persona = 'Guido', 'van Rossum'
```

Una tupla puede incluir un único elemento, pero para que Python entienda que nos estamos refiriendo a una tupla es necesario escribir al menos una coma.

```
persona = ('Guido',)
```

Se puede convertir una tupla en una lista usando la función **list**()

Y viceversa, una lista se puede convertir en una tupla usando la función **tuple**()

```
tupla = (1, 2, 3)
tuplaAListа = list(tupla)
print(tuplaAListа) #[1, 2, 3]

lista = [5, 8, 13]
listaATupla = tuple(lista)
print(listaATupla) #(5, 8, 13)
```

Operadores

El operador de pertenencia **in** se utiliza para saber si un elemento está contenido en una tupla, así como su complemento el **not in.**

```
vocales = ('a', 'e', 'i', 'o', 'u')
if ('b' in vocales):
    print("Es una vocal")
elif ('b' not in vocales):
    print("No es una vocal")
```

Para concatenar 2 tuplas y generar una nueva tupla se puede utilizar el operador **+**

```
tupla1 = ("Moe", "Larry", "Curly")
tupla2 = ("Laurel", "Hardy")
tupla_unida = tupla1 + tupla2
print(tupla_unida)
# ('Moe', 'Larry', 'Curly', 'Laurel', 'Hardy')
```

Operadores relacionales

Dos tuplas **son iguales** cuando tienen el mismo tamaño y cada uno de sus elementos correspondientes tienen el mismo valor:

```
print((3, 5) == (33 / 11, 2 + 3)) #True
print((1, 3) == (3, 1)) #False
print((3, 3) == (3, 3)) #True
print((1, 2) == (0, 1, 2)) #False
```

Para determinar si una tupla **es menor** que otra, se utiliza lo que se denomina **orden lexicográfico.** Si los elementos en la primera posición de ambas tuplas son distintos, ellos determinan el ordenamiento de las tuplas:

```
print((1, 2, 3) < (2, 3, 5, 8)) #True
print((1, 2, 3) < (1, 2, 4)) #True
print((8, 13) < (0, 21)) #False
print((1, 2) < (1, 2, 4)) #True
```

La primera comparación es True porque 1 es menor que 2. No importa el valor que tengan los siguientes valores, o si una tupla tiene más elementos que la otra.

Si los elementos en la primera posición son iguales, entonces se usa el valor siguiente para hacer la comparación, es por eso que la segunda comparación da como resultado True.

Si a una tupla se le acaban los elementos para comparar antes que a la otra, entonces es considerada menor que la otra (es por eso que la última comparación da True).

Métodos y Funciones

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

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

```
numeros = (1, 1, 1, 3, 5)
print(numeros.count(1)) #3


print(numeros.index(5)) #4
```

En el caso de no existir el índice, se devuelve el error **ValueError**.

La función **len**() devuelve el número de elementos que contiene una tupla.

Al igual que las listas, las tuplas son **iterables**:

```
print(len(vocales)) # 5
```

```
for vocal in vocales:
    print(vocal)
for i in range(0, len(vocales)):
    print(vocales[i])
```

Desempaquetado

Los valores individuales de una tupla pueden ser recuperados asignando cada elemento a su variable respectiva. Esto se llama **desempaquetar** (**unpack**) la tupla:

```
persona = ('Guido', 'van Rossum')
nombre, apellido = persona
print("Nombre: ", nombre, " - Apellido: ", apellido)
```

La cantidad de variables que se va a asignar el contenido de la tupla, debe ser coincidente; en caso contrario, ocurrirá un error de asignación.

También es posible acceder a cada elemento de la tupla por medio de sus índices, tal como lo hacíamos al momento de trabajar con listas:

```
nombre = persona[0]
apellido = persona[1]
```

Como se mencionó y a diferencia de las listas, por ser un tipo inmutable no es posible editar el contenido de los elementos.