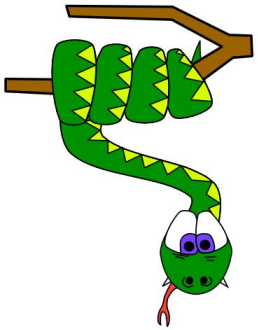

TUPLAS EN PYTHON

CTIC - UNI

Victor Melchor

Tuplas

- Son similares a las listas pero
  - Son Inmutables
  - Delimitadas por paréntesis
  - Una tupla con un solo elemento **debe** tener una coma dentro de los paréntesis:

    a = (11,)

Tupla Vacía

- Hay dos formas de inicializar una tupla vacía. Puede inicializar una tupla vacía escribiendo () sin ningún valor dentro.
  TuplaVacia1 = ()

- También puede inicializar una tupla vacía utilizando la función **tuple**.
  TuplaVacia2 = tuple ()

Ejemplos

- >>> mi_tupla = (11, 22, 33)
- >>> mi_tupla[0]
  11
- >>> mi_tupla[-1]
  33
- >>> mi_tupla[0:1]
  (11,)
- Se requiere la coma!

Por qué?

- No hay confusión posible entre **[11]** y **11**

- **(11)** es una expresión perfectamente aceptable
  - **(11) sin la coma es** el entero 11
  - **(11,) con la coma es** una lista que contiene el entero 11

Las tuplas son inmutables

- >>> mi_tupla = (11, 22, 33)
- >>> copia = mi_tupla
- >>> mi_tupla += (44,)
- >>> mi_tupla
  (11, 22, 33, 44)
- >>> copia
  (11, 22, 33)

Operaciones en Tuplas

- Las tuplas admiten todas las operaciones de secuencia estándar, en el que se incluyen:
  - Pruebas de membresía (usando la palabra clave **in**)
  - Comparación (elemento a elemento)
  - Iteración (por ejemplo, en un bucle for)
  - Concatenación y repetición

Prueba de Membresía (`in`)

- En Python, la palabra clave in se usa para probar si una secuencia (lista, tupla, cadena, etc.) contiene un valor.
  - Retorna ***`True`*** o ***`False`***

¿Qué produce esta salida?

```
a = [1, 2, 3, 4, 5]
print(5 in a)
print(10 in a)
```

```
True
False
```

Comparación

- En Python 3.3, podemos usar el operador de comparación, ==, para hacer la comparación de tuplas
  - Retorna ***True*** o ***False***

¿Qué produce esta salida?

```
tupla1, tupla2 = (123, 'xyz'), (456, 'abc')
tupla3 = (456, 'abc')
print (tupla1==tupla2)
print (tupla2==tupla3)
```

```
False
True
```

Iteracion

```
equipos = ((1, 'Argentina'),(2, 'Brasil'),
           (5, 'Ecuador'),(7, 'Suiza'))
```

tupla de tuplas

```
for (indice, nombre) in equipos:
   print(indice, nombre)
```

¿Qué produce esta salida?

```
1 Argentina
2 Brasil
5 Ecuador
7 Suiza
```

Iteracion

```
t = [('a', 0), ('b', 1), ('c', 2)]
for letra, numero in t:
    print (numero, letra)
```

tupla de tuplas

¿Qué produce esta salida?

```
0 a
1 b
2 c
```

Concatenación (+)

- El operador + devuelve una nueva tupla que es una concatenación de dos tuplas.

```
a = (1, 2, 3)
b = (4, 5, 6)
c = a + b
print (a, b, c)
```

¿Qué produce esta salida?

```
(1, 2, 3) (4, 5, 6) (1, 2, 3, 4, 5, 6)
```

Repetición (*)

- El operador * devuelve una nueva tupla que repite la tupla.

```
a = (1, 2, 3)
b = (4, 5, 6)
print (a*2, b)
```

¿Qué produce esta salida?

```
(1, 2, 3, 1, 2, 3) (4, 5, 6)
```

Función `len()`

- La función **`len()`** retorna el número de elementos en la tupla.

```
tupla0 = ()
print(len(tupla0))
tuplaA = ("UNI", "es", "la", "mejor")
print(len(tuplaA))
```

¿Qué produce esta salida?

0

4

Funciones `min()` y `max()`

- **`max(tupla)`**
  - Devuelve el item de la tupla con valor máximo
- **`min(tupla)`**
  - Devuelve el item de la tupla con valor mínimo

```
miTupla = tuple('margot')
print (miTupla)

print(min(miTupla))
print(max(miTupla))
```

¿Qué produce esta salida?

```
('m', 'a', 'r', 'g', 'o', 't')
a
t
```

Cosas que no funcionan

- **mi_tupla += 55**
  Traceback (most recent call last):Z
  …
  TypeError:
  can only concatenate tuple (not "int") to tuple

  – Solo podemos concatenar tuplas!

Ordenando tuplas

- >>> tupla = (33, 22, 11)
- >>> tupla.sort()
  Traceback (most recent call last):
  ...
  AttributeError:
  'tuple' object has no attribute 'sort'
- >>> lista = sorted(tupla)
- >>> lista
  [11, 22, 33]

Tuplas son inmutables!

sorted( ) retorna una lista!

¡La mayoría de las otras cosas funcionan!

- >>> tupla = (11, 22, 33)
- >>> len(tupla)
  3
- >>> 44 in tupla
  False
- >>> [i for i in tupla]
  [11, 22, 33]

Lo contrario no funciona

- >>> lista = [11, 22, 33]
- >>> (i for i in lista)
  <generator object <genexpr> at 0x02855DA0>
  - *No funciona!*

Convertir secuencias en tuplas

- >>> lista = [11, 22, 33]
- >>> tupla = tuple(lista)
- >>> tupla
  (11, 22, 33)
- >>> otra_tupla = tuple('Hello World!')
- >>> otra_tupla
  ('H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!')

Conversión entre tuplas y listas

- Para la conversión entre tuplas y listas use las funciones tuple y list:

- >>> lista = [11, 22, 33]
- >>> tu = tuple(lista)
- >>> li = list(tu)

Tuplas y funciones

- Las funciones de Python (como ocurre con la mayoría de lenguajes) solo pueden devolver un valor
  - Pero hemos devuelto múltiples valores antes!
- Si se agrupan varios objetos en una tupla, la función puede devolver los objetos como una sola tupla.
- Muchas funciones de Python devuelven tuplas.

Ejemplo: min_max.py

```
# Devuelve los elementos más pequeño y más grande
# de una secuencia como una tupla
def min_max(t):
    return min(t), max(t)

lista = [12, 98, 23, 74, 3, 54]
print (min_max(lista))

string = 'Este domingo fue un dia espectacular!'
print (min_max(string))
miMin, miMax = min_max(string)
```

¿Qué hace esta salida?

```
(3, 98)
(' ', 'w')
```

Paso de Tuplas como Parámetros

- Un nombre de parámetro que comienza con * reúne todos los argumentos en una tupla.
- Esto permite que las funciones tomen un número variable de parámetros.
  - ¡Entonces podemos llamar a la función con uno, dos o veinte parámetros!
- (Un parámetro actual también se llama argumento)

Ejemplo

```
def EscribeTodos(*args):
    print (args)

EscribeTodos(1, 2.0, 'three')
```

Ejemplo: Presentar.py

```
def Presentar(requerido, *args):
    print ('Requerido:', requerido)
       if args:
        print('Otros:  ', args)
```

¿Qué hace esta salida?

```
Presentar(1)
Presentar(1, 2)
Presentar(1, 2.0, 'tres')
Presentar(1, 2.0, 'tres', [4])
```

```
Requerido: 1
Requerido: 1
Otros:   (2,)
Requerido: 1
Others:   (2.0, 'tres')
Requerido: 1
Otros:   (2.0, 'tres', [4])
```

Eliminación de tuplas

Las tuplas son inmutables y no se pueden eliminar, pero es posible eliminarlas por completo mediante la palabra clave **del**

```
>>> t = (1, "abc", 3.0)
>>> del t
```

Métodos no soportados por tuplas

- Las tuplas no soportan los métodos:
  - append
  - remove
  - insert
  - reverse
  - sort