**Tuplas**

**Qué son las tuplas**

En Python, una **tupla** es un conjunto ordenado e inmutable de elementos del mismo o diferente tipo.

Las tuplas se representan escribiendo los elementos entre paréntesis y separados por comas.

>>> (1, "a", 3.14)

(1, 'a', 3.14)

En realidad no es necesario escribir los paréntesis para indicar que se trata de una tupla, basta con escribir las comas, pero Python escribe siempre los paréntesis:

>>> 1, "a", 3.14

(1, 'a', 3.14)

La función len() devuelve el número de elementos de una tupla:

>>> len((1, "a", 3.14))

3

Una tupla puede no contener ningún elemento, es decir, ser una tupla vacía.

>>> ()

()

>>> len(())

0

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.

El ejemplo siguiente muestra la diferencia entre escribir o no una coma. En el primer caso Python interpreta la expresión como un número y en el segundo como una tupla de un único elemento.

>>> (3)

3

>>> (3,)

(3,)

Python escribe una coma al final en las tuplas de un único elemento para indicar que se trata de un tupla, pero esa coma no indica que hay un elemento después:

>>> (3,)

(3,)

>>> len((3,))

1

# Operaciones con tuplas

Las operaciones sobre tuplas son las mismas que vimos con las listas. El operador corchete selecciona un elemento de la tupla.

>>> tupla = (’a’, ’b’, ’c’, ’d’, ’e’)

>>> tupla[0]= ’a’

Y el operador segmento selecciona un rango de elementos:

>>> tupla[1:3]

(’b’, ’c’)

Pero si intentamos modificar un elemento de la tupla obtenemos un error:

>>> tupla[0] = ’A’

TypeError: object doesn’t support item assignment

Aunque no podemos modificar los elementos, sí podemos modificar toda la tupla:

>>> tupla = (’A’,) + tupla[1:]

>>> tupla

(’A’, ’b’, ’c’, ’d’, ’e’)

>>> tupla = (1,2,3)

>>> tupla

**Asignación de tuplas**

De vez en cuando necesitamos intercambiar los valores de dos variables. Con el operador de asignación normal tenemos que usar una variable temporal. Por ejemplo, para intercambiar a y b:

>>> temp = a

>>> a = b

>>> b = temp

Si tenemos que intercambiar variables muchas veces, el código tiende a ser engorroso. Python proporciona una forma de **asignación de tuplas** que resuelve este problema:

>>> a, b = b, a

## Tuplas como valores de retorno

El lado izquierdo es una tupla de variables; el derecho es una tupla de valores. Cada valor se asigna a su respectiva variable en el orden en que se presenta. Las expresiones en el lado derecho se evalúan antes de cualquier asignación. Esto hace a la asignación de tuplas una herramienta bastante versátil.

Naturalmente, el número de variables a la izquierda y el número de valores a la derecha deben coincidir.

>>> a, b, c, d = 1, 2, 3

ValueError: unpack tuple of wrong size

### Métodos

Son muy similares a las *listas* y comparten varias de sus funciones y métodos integrados, aunque su principal diferencia es que son inmutables. El objeto de tipo *tupla* integra una serie de métodos integrados a continuación:

#### count()

Este método recibe un elemento como argumento, y cuenta la cantidad de veces que aparece en la tupla.

**>>>** valores = ("Python", True, "Zope", 5)

**>>> print** "True ->", valores.count(True)

True -> 1

**>>> print** "'Zope' ->", valores.count('Zope')

'Zope' -> 1

**>>> print** "5 ->", valores.count(5)

5 -> 1

#### index()

Comparte el mismo método *index()* del tipo lista. Este método recibe un elemento como argumento, y devuelve el índice de su primera aparición en la tupla.

**>>>** valores = ("Python", True, "Zope", 5)

**>>> print** valores.index(True)

1

**>>> print** valores.index(5)

3

**3.10. Tipo tuplas**

El método devuelve una excepción *ValueError* si el elemento no se encuentra en la tupla, o en el entorno definido.

**>>>** valores = ("Python", True, "Zope", 5)

**>>> print** valores.index(4)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: tuple.index(x): x not in tuple

### Convertir a tuplas

Para convertir a *tipos tuplas* debe usar la función *tuple()* la cual *está integrada* en el intérprete Python.

De la misma forma que str(42) devuelve '42', esto es, la forma cadena del entero 42, las funciones list() y tuple() devuelven las versiones tipo lista y tipo tupla de los valores que les pasemos. Escribe lo siguiente en el shell interactivo, y observa que el valor de retorno es de un tipo de datos distinto al valor pasado:

### 

### Ejemplos

A continuación, se presentan algunos ejemplos de su uso: Ejemplo simple de tupla

tupla = 12345, 54321, 'hola!'

Ejemplo de tuplas anidadas

otra = tupla, (1, 2, 3, 4, 5)

Operación asignar de valores de una tupla en variables

x, y, z = tupla

Cuidar seguimiento del número de la numeración

Una tarea común es iterar sobre una secuencia mientras cuidas el seguimiento de la numeración de un elemento.

Podría usar un bucle while con un contador o un bucle for usando la función *range()* y la función *len()*:

**>>>** tecnologias = ('Zope', 'Plone', 'Pyramid') **>>> for** i **in** range(0, len(tecnologias)):

**... print** i, tecnologias[i]

**...**

1. Zope
2. Plone
3. Pyramid

Pero, Python provee la palabra reservada enumerate para esto:

**for** index, item **in** enumerate(conexion\_completa): **print** index, item

### Ayuda integrada

Usted puede consultar toda la documentación disponible sobre las tuplas desde la *consola interactiva* de la siguiente forma:

**>>>** help(tuple)