Estructuras de Datos II

Tuplas

Jonatan Gomez Perdomo, Ph. D.
jgomezpe@unal.edu.co

Arles Rodríguez, Ph.D.
aerodriguezp@unal.edu.co

Camilo Cubides, Ph.D. (c)
eccubidesg@unal.edu.co

Carlos Andrés Sierra, M.Sc.
casierrav@unal.edu.co

Research Group on Articial Life
Grupo de investigación en vida artificial (Alife)
Computer and System Department
Engineering School
Universidad Nacional de Colombia

Agenda

Hechos QUE CONECTAN

UNIVERSIDAD NACIONAL DE COLOMBIA un

Definición

Una tupla es una secuencia de elementos que puede almacenar datos heterogéneos tales como: enteros, reales, cadenas, listas, diccionarios y otros más, inclusive otras tuplas. Una tupla se escribe como la secuencia de datos a mantener, separados por una coma (`,`), secuencia delimitada por los paréntesis redondos. Como las cadenas de caracteres, las tuplas son inmutables, esto es, no se pueden modificar después de definidas.

`()` : La tupla vacía.

`("Este es un texto",)` : Una tupla con un elemento que es una cadena. Para el caso de tuplas de un sólo elemento Python requiere una coma final para considerarla una tupla.

`('Una cadena', 123)` : Una tupla de dos elementos, el primero una cadena y el segundo un número entero.

`(1, 2, 3, 4.5, 'hola', 'a')` : Una tupla de seis elementos.

Variables

Una tupla se puede asignar a una variable de dos maneras: Usando los paréntesis redondos o sólo la secuencia de datos separados por comas.

`x = ()` : Le asigna la tupla vacía a la variable `x`.

`tup = (1, 2, 3, 4.5, 'hola', 'a')` : Le asigna la tupla de seis elementos a la variable `tup`.

`a = 1, 2, 3` : Le asigna la tupla `(1, 2, 3)` a la variable `a`.

Tuplas anidadas

Tuplas con tuplas como elementos (anidadas). Para el programa

```
tuple1 = (0, 1, 2, 3)
tuple2 = ("A", "B", "C")
tuple3 = (tuple1, tuple2)
print(tuple3)
print(tuple3[0])
print(tuple3[1])
print(tuple3[1][0])
```

La salida obtenida es

```
((0, 1, 2, 3), ('A', 'B', 'C'))
(0, 1, 2, 3)
('A', 'B', 'C')
A
```

Agenda

Hechos QUE CONECTAN

UNIVERSIDAD NACIONAL DE COLOMBIA un

Concatenar +

Concatena dos tuplas. Para el programa

```
tup1 = ("A", "B", "C", "E")
tup2 = (1, 2, 3, 4, 5)
tup3 = tup1 + tup2
print(tup3)
```

La salida obtenida es

```
('A', 'B', 'C', 'E', 1, 2, 3, 4, 5)
```

Repetir *

Crea una tupla con múltiples copias de una tupla, tantas como se defina.
Para el programa

```
tup2 = (1, 2, 3, 4, 5)
tup3 = tup2 * 3
print(tup3)
tup4 = ("Abc", "Bcd")
tup5 = tup4 * 2
print(tup5)
```

La salida obtenida es

```
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
('Abc', 'Bcd', 'Abc', 'Bcd')
```

Comparar I

Se usan los operadores convencionales (`<`, `<=`, `>`, `>=`, `==`, `!=`) para comparar tuplas usando el orden **lexicográfico**, cuando es posible. En el orden lexicográfico, se tratan de comparar (si el operador $<$ está definido), de izquierda a derecha uno a uno los elementos de la tupla, mientras sean iguales. En el caso en el que no este definido el orden (por ejemplo entre cadenas y números), la comparación genera un error. En el caso que no sean iguales, si el elemento de la primera tupla es menor que el de la segunda, la primer tupla se considera la menor, si el elemento es mayor la primer tupla se considera la mayor. Si todos los elementos son iguales, se consideran iguales a las tuplas.

Comparar II

Para el programa

```
print(("Rojas", 123) < ("Rosas", 123))
print(("Rosas", 123) == ("rosas", 123))
print(("Rosas", 123) > ("Rosas", 23))
print(("Rojas", "123") > ("Rosas", 23))
print(("Rosas", "123") > ("Rosas", 23))
```

La salida obtenida es

```
True
False
True
False
error '<'
```

Subíndice [ ]

Accede los elementos de una tupla. Si la posición que se envía es negativa, lo considera desde el final. Si no es válida genera error. Para el programa

```
avengers = ("Ironman", "Thor", "Ant-man", "Hulk")
print(avengers[0])
print(avengers[3])
print(avengers[-1])
print(avengers[-3])
```

La salida obtenida es

```
Ironman
Hulk
Hulk
Thor
```

Hechos QUE CONECTAN

UNIVERSIDAD NACIONAL DE COLOMBIA un

Agenda

Hechos QUE CONECTAN

UNIVERSIDAD NACIONAL DE COLOMBIA un

Consultando una tupla

Es posible determinar si un elemento se encuentra en una tupla. Para el programa

```
text = ("cien", "años", "de", "soledad")
if "años" in text:
   print("Si está en la tupla")
else:
   print("No está en la tupla")
```

La salida obtenida es

```
Si está en la tupla
```

Consultando una tupla

Es posible determinar si un elemento no se encuentra en una tupla. Para el programa

```
text = ("cien", "años", "de", "soledad")
if "compañia" not in text:
    print("No está en la tupla")
else:
    print("Si está en la tupla")
```

La salida obtenida es

```
No está en la tupla
```

Iterando una tupla

Es posible iterar una tupla usando el ciclo `for`. Para el programa

```
s = ("hola", "amigos", "mios")
for palabra in s:     # para cada palabra de la tupla
   print(palabra, end = ", ")
```

La salida obtenida es

```
hola, amigos, mios,
```

Asignando múltiples variables

Es posible asignarle los valores a un grupo de variables usando la asignación y el concepto de tupla. Para el programa

```
tupla = (1, -2, 3)
a, b, c = tupla
print("a =", a)
print("b =", b)
print("c =", c)
```

La salida obtenida es

```
a = 1
b = -2
c = 3
```

Intercambiando variables

Es posible intercambiar los valores de un grupo de variables usando la asignación y el concepto de tupla. Para el programa

```
a = 1
b = 3
a, b = b, a
print("a =", a)
print("b =", b)
```

La salida obtenida es

```
a = 3
b = 1
```

Asignando múltiples variables desde una tupla

Es posible asignar los valores de un grupo de variables usando la asignación, el ciclo `for` y el concepto de tupla. Para el programa

```
tupla = (11, 9, -2, 3, 8, 5)
var1, var2, var3 = (tupla[i] for i in (1, 3, 5))
print("var1 =", var1, ", var2 =", var2, ", var3 =", var3)
var1, var2, var3 = (tupla[i] for i in range(0,6,2))
print("var1 =", var1, ", var2 =", var2, ", var3 =", var3)
```

Para el anterior programa, la salida obtenida es

```
var1 = 9 , var2 = 3 , var3 = 5
var1 = 11 , var2 = -2 , var3 = 8
```

Tuplas y funciones

Retornar más de un valor en una función usando el concepto de tupla.

```
def minmax(a, b):
    if a < b:
        return a, b
    else:
        return b, a
x, y = minmax(5, 13)
print("min =", x, ",", "max =", y)
x, y = minmax(12, -4)
print("min =", x, ",", "max =", y)
```

Para el anterior programa, la salida obtenida es

```
min = 5 , max = 13
min = -4 , max = 12
```

Agenda

Longitud (`len`)

la función `len` determina la dimensión (longitud) de una tupla.

```
tup = (1, 2, 3, 4)
nombre = ("Minch", "Yoda")
trabajo = ("Stars", "War", "Movie")
empty = ()
print(len(tup))
print(len(nombre))
print(len(trabajo))
print(len(empty))
```

Para el anterior programa, la salida obtenida es

```
4
2
3
0
```

Subtuplas (slice)

La función `slice` obtiene una porción (subtupla) de una tupla. La definición es similar a la función `slice`, de cadena `[inicio:fin:incremento]`. Para el programa

```
avengers = ("Ironman", "Thor", "Ant-man", "Hulk")
print(avengers[:2])
print(avengers[1:3])
print(avengers[3:3])
print(avengers[::-1])
```

La salida obtenida es

```
('Ironman', 'Thor')
('Thor', 'Ant-man')
()
('Hulk', 'Ant-man', 'Thor', 'Ironman')
```

Contando (count)

El método count obtiene las veces que un elemento se encuentra en una tupla. Para el programa

```
tupla = (4, 3, 8, 8, 2, 5, 4, 6, 8, 9)
print(tupla.count(2))
print(tupla.count(8))
print(tupla.count(5))
print(tupla.count(7))
```

La salida obtenida es

```
1
3
1
0
```

Buscando (index)

El método index obtiene la primera ocurrencia de un elemento en una tupla. Para el programa

```
tupla = (4, 3, 8, 8, 2, 5, 4, 6, 8, 9)
print(tupla.index(2))
print(tupla.index(8))
print(tupla.index(5))
```

La salida obtenida es

```
4
2
5
```

En caso de que el objeto que se esté buscando no se encuentre en la tupla, se generará una excepción.

Máximo y mínimo (`max, min`)

El método `max/min` obtiene el máximo/mínimo elemento de una tupla. Para el programa

```
t = (4, 5, -1, 6, 7)
print(max(t))
print(min(t))
```

La salida obtenida es

```
7
-1
```

De cadena a tupla (tuple)

El método `tuple` se usa para crear tuplas a partir de otros objetos, aquí se usa para convertir una cadena de caracteres a tupla. Para el programa

```
magician = "Dumbledore"
tm = tuple(magician)
print(tm)
```

La salida obtenida es

```
('D', 'u', 'm', 'b', 'l', 'e', 'd', 'o', 'r', 'e')
```

Desempacar variables (unpacking)

Una forma rápida de desempacar y asignar variables de una tupla es la siguiente:

```
tup1 = (1, 2, 3)
a, b, c = tup1
print("a:", a, "b:", b, "c:", c)
```

La salida obtenida es

```
a: 1 b: 2 c: 3
```

La función map (map)

Ejecuta una función para cada uno de los valores de una tupla o una lista, en el ejemplo siguiente convierte a `int` los valores leídos:

```
t = tuple(map(int, input().split(" ")))    # digite 1 2 3
print(t)
print(t[0] + t[1])
```

La salida obtenida es

```
1 2 3
(1, 2, 3)
3
```

Sugerencia

Se sugiere consultar un manual de Python o de sus librerías para determinar si ya existe un método para lo que se quiera realizar con una tupla.