TEL101 - Iniciación a la Programación

Clase 8: Tuplas

Prof. Nicolás Torres
nicolas.torresr@usm.cl
Ingeniería Civil Telemática
Departamento de Electrónica
Universidad Técnica Federico Santa María

Tuplas

Secuencia **inmutables**, y generalmente contienen elementos heterogéneos (del mismo tipo). Aunque las tuplas pueden parecer similares a las listas, a menudo se usan en diferentes situaciones y para diferentes propósitos.

Creación de tuplas

Las tuplas se pueden declarar con o sin paréntesis `( )`, y sus elementos se separan por comas. Si no se escriben los paréntesis, de todas formas aparecerán con ellos. Sin embargo, en algunos casos es obligatorio escribir los paréntesis (si la tupla es parte de una expresión más grande). Por lo tanto, se recomienda siempre declarar una tupla entre paréntesis.

In [1]:
```
x = 1,2
print(x)
```

```
(1, 2)
```

In [2]:
```
coordenadas = (-33.491076, -70.618951)
```

In [3]:
```
fecha = (2010, 5, 12)
```

In [4]:
```
carta = (5, 'corazones')
```

Indexación

Al igual que las listas y los strings, las tuplas tienen elementos ordenados posicionalmente desde el primero hasta el último, con índices positivos $(0, 1, 2, \ldots)$, y desde el último hasta el primero, a través de índices negativos $(-1, -2, -3, \ldots)$.

In [5]:
```
x = ('a', 'b', 'c', 'd', 'e')
```

In [6]:
```
print(x[1])
```

```
b
```

In [7]:
```
print(x[-1])
```

e

Además, soportan todas las operaciones de *Slicing*.

Inmutabilidad

A diferencia de las listas, las tuplas son un tipo de dato **inmutable**. Por lo tanto, sus elementos **no se pueden reemplazar**.

In [8]:
```
x = (1, 2, 3)
x[0] = 4
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [8], in <cell line: 2>()
      1 x = (1, 2, 3)
----> 2 x[0] = 4

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

Operaciones sobre tuplas

Las operaciones elementales de concatenación, repetición y membresía, son compatibles con las tuplas.

In [9]:
```
(1, 2, 3) + (4, 5, 6)
```

Out[9]: (1, 2, 3, 4, 5, 6)

In [10]:
```
('a', 'b') * 3
```

Out[10]: ('a', 'b', 'a', 'b', 'a', 'b')

In [11]:
```
2 in (1, 2, 3)
```

Out[11]: True

In [12]:
```
'2' in (1, 2, 3)
```

Out[12]: False

Funciones sobre tuplas

Las funciones `len()`, `min()`, `max()` y `sum()`, se pueden aplicar sobre tuplas.

In [13]:
```
len(('a', 'b', 'c'))
```

Out[13]: 3

In [14]:
```
min((6, 4, 1, 9, 5))
```

Out[14]: 1

In [15]:
```
max((6, 4, 1, 9, 5))
```

Out[15]:
```
9
```

In [16]:
```
sum((6, 4, 1, 9, 5))
```

Out[16]:
```
25
```

Desempaquetado

El desempaquetado (en inglés *unpacking*), es una operación que consiste en asignar cada elemento de una secuencia, a una sola variable.

In [17]:
```
tupla = (1, 2, 3, 4, 5)
a,b,c,d,e = tupla

print(a)
print(c)
print(e)
```

```
1
3
5
```

Esto se llama, apropiadamente, "desempaquetado de secuencias" y funciona para cualquier secuencia en el lado derecho. Sin embargo, no es común desempaquetar listas o strings, ya que, son secuencias de muchos elementos y, en el caso de las listas, su longitud puede variar durante la ejecución.

`ValueError: too many values to unpack`

La cantidad de variables al lado izquierdo debe ser **exactamente igual al largo de la secuencia**.

In [18]:
```
tupla = (1, 2, 3, 4, 5)
a,c,e = tupla
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [18], in <cell line: 2>()
      1 tupla = (1, 2, 3, 4, 5)
----> 2 a,c,e = tupla

ValueError: too many values to unpack (expected 3)
```

Convención

Para no definir variables innecesarias, que luego no se usarán, se utiliza un guion bajo para descartar esos elementos.

In [19]:
```
tupla = (1, 2, 3, 4, 5)
a,_,c,_,e = tupla
print(a,c,e)
```

```
1 3 5
```

Iteraciones

Las tuplas no se suelen recorrer a través de un ciclo porque, generalmente, tienen pocos elementos. Por lo tanto, se accede a ellos mediante su posición o desempaquetando.

Listas de tuplas

En general, como las tuplas son inmutables, se utilizan para relacionar conceptos o entidades, como por ejemplo una persona y su edad, género, etc. Sin embargo, para representar a un grupo de personas, las tuplas se agrupan dentro de una estructura que permita agregar, eliminar y ordenar elementos, como una lista.

Las listas de tuplas son iterables, por lo tanto, se pueden iterar usando un ciclo `for` o `while`.

In [20]:
```
edades = [('juan', 55), ('andres', 40), ('juana', 12), ('andrea', 45)]
```

In [21]:
```
for x in edades:
    print("La edad de",x[0],"es",x[1])
```

```
La edad de juan es 55
La edad de andres es 40
La edad de juana es 12
La edad de andrea es 45
```

In [22]:
```
for x in edades:
    nombre, edad = x
    print("La edad de",nombre,"es",edad)
```

```
La edad de juan es 55
La edad de andres es 40
La edad de juana es 12
La edad de andrea es 45
```

In [23]:
```
for nombre,edad in edades:
    print("La edad de",nombre,"es",edad)
```

```
La edad de juan es 55
La edad de andres es 40
La edad de juana es 12
La edad de andrea es 45
```

Ordenamiento

Una lista de tuplas se ordena lexicográficamente. Por lo tanto, el primer elemento de cada tupla es el valor que determina la disposición de los elementos en la lista.

In [24]:
```
edades = [('juan', 55), ('andres', 40), ('juana', 12), ('andrea', 45)]
edades.sort()
print(edades)
```

```
[('andrea', 45), ('andres', 40), ('juan', 55), ('juana', 12)]
```

Ordenar por edad

Si el objetivo no es ordenar en base al primer elemento de cada tupla es necesario reestructurar los elementos dentro de cada tupla.

In [25]:
```
nueva = []
# Se invierte el orden de cada tupla dentro de la lista
for nombre,edad in edades:
    nueva.append((edad, nombre))
# Se ordena la lista
nueva.sort()
# Las tuplas vuelven a su orden original, pero la lista se mantiene ordenada
final = []
for edad,nombre in nueva:
    final.append((nombre,edad))
print(final)
```

```
[('juana', 12), ('andres', 40), ('andrea', 45), ('juan', 55)]
```

Retorno múltiple

Las funciones pueden retornar más de un valor, separándolos por comas. Sin embargo, en Python, varios elementos separados por comas corresponden a una tupla.

In [26]:
```
def operaciones(a, b):
    return a+b, a-b, a*b, a/b
```

In [27]:
```
print(operaciones(4, 2))
print("4 + 2 =",operaciones(4,2)[0])
print("4 - 2 =",operaciones(4,2)[1])
print("4 * 2 =",operaciones(4,2)[2])
print("4 / 2 =",operaciones(4,2)[3])
```

```
(6, 2, 8, 2.0)
4 + 2 = 6
4 - 2 = 2
4 * 2 = 8
4 / 2 = 2.0
```