

NIVEL 4

TUPLAS



TUPLAS, ¿CUÁNDO USARLAS Y CÓMO SON?

- ✓ Cuando necesitamos agrupar cualquier cantidad de ítems (datos) en un solo valor compuesto
- ✓ Sintácticamente, una tupla es una secuencia de valores separados por comas
- ✓ Aunque no es obligatorio, es una convención encerrar las tuplas entre paréntesis

(,)

EJEMPLO - TUPLA

Una tupla nos permite "juntar" información relacionada y utilizarla como un solo elemento

No hay descripción de lo que significa cada uno de los campos, pero se puede "adivinar"

```
Terminal de IPython
Terminal 1/A x
In [3]: gabo = ("Gabriel", "García Márquez", 1927, "Aracataca, Colombia", 2014, "México, México", "Premio Nobel", 1982)
In [4]: gabo
Out[4]:
('Gabriel',
 'García Márquez',
 1927,
 'Aracataca, Colombia',
 2014,
 'México, México',
 'Premio Nobel',
 1982)
In [5]: type(gabo)
Out[5]: tuple
```

Una tupla es una estructura de datos y a la vez un tipo

OPERACIONES SOBRE TUPLAS

- ✓ Las tuplas soportan las mismas operaciones de secuencia que los strings. El operador de indexación selecciona un elemento de una tupla:

```
Terminal 1/A [X]
In [6]: gabo[2]
Out[6]: 1927
```

- ✓ Tampoco podemos modificar un elemento de una tupla (**inmutables** como los strings):

```
Terminal 1/A [X]
In [7]: gabo[0] = "X"
Traceback (most recent call last):

File "<ipython-input-7-b15afa2f7b35>", line 1, in <module>
    gabo[0] = "X"

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

LAS TUPLAS SON INMUTABLES



- Como los strings, las tuplas son **inmutables**. Una vez que Python ha creado una tupla en memoria, esta no puede ser modificada. Si queremos modificar una tupla debemos crear una nueva tupla, que contenga información de la tupla original modificada (usando slices):

```
Terminal de IPython
Terminal 1/A x
In [13]: gabo = gabo[:2] + ("Cien años de soledad", 1967) + gabo[2:]

In [14]: gabo
Out[14]:
('Gabriel',
 'García Márquez',
 'Cien años de soledad',
 1967,
 1927,
 'Aracataca, Colombia',
 2014,
 'México, México',
 'Premio Nobel',
 1982)
```

TUPLAS

VACIA

Una tupla vacía es una tupla con 0 componentes, y se indica como ()

```
Terminal 1/A x
In [45]: z = ()
In [46]: type(z)
Out[46]: tuple

In [47]: len(z)
Out[47]: 0
```

UNITARIA

Para crear una tupla con un solo elemento, debemos incluir la coma final, porque sin la coma final, Python trata el valor como un dato básico entre paréntesis:

```
Terminal 1/A x
In [15]: tupla = (5,)
In [16]: type(tupla)
Out[16]: tuple

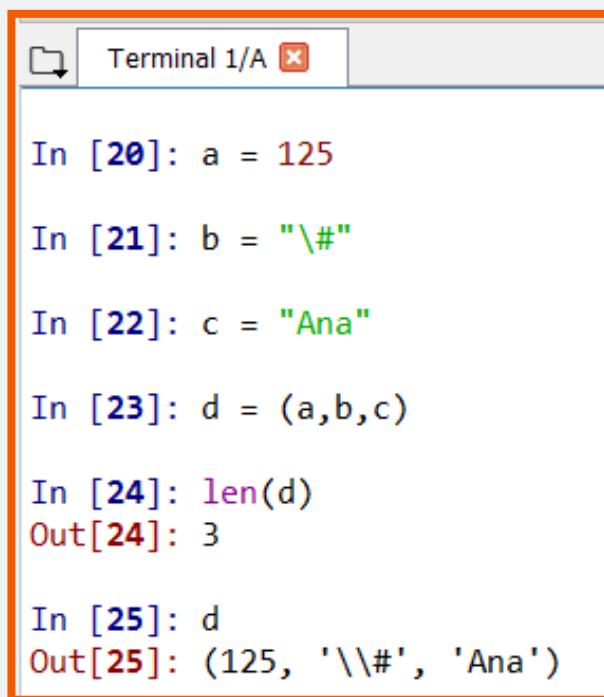
In [17]: x = (5)
In [18]: type(x)
Out[18]: int
```

ES POSIBLE ANIDAR TUPLAS

```
Terminal 1/A x
In [39]: gabo = (("Gabriel", "García Márquez"), (1927, "Aracataca, Colombia"), (2014, "México, México"), ("Premio Nobel", 1982))
In [40]: gabo
Out[40]:
(('Gabriel', 'García Márquez'),
 (1927, 'Aracataca, Colombia'),
 (2014, 'México, México'),
 ('Premio Nobel', 1982))
In [41]: len(gabo)
Out[41]: 4
In [42]: gabo[0]
Out[42]: ('Gabriel', 'García Márquez')
In [43]: gabo[1]
Out[43]: (1927, 'Aracataca, Colombia')
```

EMPAQUETADO DE TUPLAS

Si a una variable se le asigna una secuencia de valores separados por comas, el valor de esa variable será la tupla formada por todos los valores asignados. A esta operación se la denomina **empaquetado de tuplas**



```
Terminal 1/A x

In [20]: a = 125

In [21]: b = "\\#"

In [22]: c = "Ana"

In [23]: d = (a,b,c)

In [24]: len(d)
Out[24]: 3

In [25]: d
Out[25]: (125, '\\#', 'Ana')
```


DESEMPAQUETADO DE TUPLAS

```
Terminal 1/A x
In [27]: gabo = ("Gabriel", "García Márquez", 1927, "Aracataca, Colombia", 2014, "México, México", "Premio Nobel", 1982)
In [28]: (nombre, apellido, anio_nac, lugar_nac, anio_muerte, lugar_muerte, premio, anio_premio) = gabo
In [29]: nombre
Out[29]: 'Gabriel'
In [30]: apellido
...:
Out[30]: 'García Márquez'
In [31]: anio_nac
Out[31]: 1927
In [32]: lugar_nac
Out[32]: 'Aracataca, Colombia'
In [33]: anio_muerte
Out[33]: 2014
In [34]: lugar_muerte
Out[34]: 'México, México'
In [35]: premio
Out[35]: 'Premio Nobel'
In [36]: anio_premio
Out[36]: 1982
```



Si se tiene una tupla de longitud k , se puede asignar la tupla a k variables distintas y en cada variable quedará uno de los componentes de la tupla. A esta operación se la denomina **desempaquetado de tuplas**



Si las variables no son distintas, se pierden valores. Y si las variables no son exactamente k se produce un error.

TUPLAS COMO VALORES DE RETORNO

- ✓ Las funciones solo pueden devolver un único valor, pero al convertir ese valor en una tupla, podemos agrupar efectivamente todos los valores que deseemos y devolverlos juntos
- ✓ Esto es muy útil: a menudo queremos conocer el valor más alto y más bajo de una secuencia, o queremos encontrar la media y la desviación estándar, o queremos saber el año, el mes y el día de una fecha, etc

Ejemplo

```
EjemploTuplaComoRetorno.py x
1 import math
2
3 def funcion(radio: float)->tuple:
4     perimetro = 2 * math.pi * radio
5     area = math.pi * radio * radio
6     return (perimetro, area)
7
```

Resultado de ejecución

```
Terminal 2/A x
In [2]: funcion(10)
Out[2]: (62.83185307179586, 314.1592653589793)
```