

Bloque 1

Contenedores

Fundamentos de Programación
Departamento de Lenguajes y Sistemas Informáticos



Contenedores

Los contenedores son *estructura de datos* que permiten, por su tipología, semejanza, funcionalidad, etc., agruparlos bajo un único tipo de datos: Estos tipos pueden ser en Python:

- Tuplas →: (...) **p.e.:** (12,55,60,--) *tupla de números*
- Listas → [...] **p.e.:** ["pepe", "ana", 'Isabel',...] *lista de nombres*
- Conjuntos → {...} **p.e.:** {"pepe", "ana", 'Isabel',...} *conjunto de nombre*
- Diccionesarios → {.: ., . : ...} **p.e.:** {"pepe": 5, "ana":3, 'Isabel':1,...} *diccionario con
¿cuántas veces
aparece un nombre?*



Contenedores: **tuple (tupla)**

Es una variable que, generalmente, se usa almacena más de un dato sobre un mismo objeto y referenciarlos bajo una misma denominación.

Sintaxis: se encierran entre paréntesis y separados por coma los valores del objeto en cuestión.

Por ejemplo:

- tuplas con datos de personas –*con 6 datos por persona*:-
 - persona1=('12345678A', 'Arancha', 'López Martín', 18, 'Sevilla', 'Sevilla')
 - Persona2=('111222333B', 'Antonio', 'Gómez Benítez', 19, 'Lepe', 'Huelva')
- tuplas con datos sobre admisión a grados –*con 4 datos por grado*:-
 - grado1=('Ingeniería del Software', 11.184, 174, '814502')
 - grado2=('Tecnología Informática', 9,941, 124, '823504')
- **Las tuplas son inmutables** → Una vez creada una tupla, no se puede añadir, suprimir o modificar sus datos. Es necesario, generar una nueva a partir de sus datos.



Contenedores: **tuple (tupla)**

Ejemplo de acceso a los elementos de una tupla **sin nombre**

Si se definen de las siguientes tuplas:

- persona1=('12345678A', 'Arancha', 'López Martín', 'M', 18, 'Sevilla', 'Sevilla')
- persona2=('111222333B', 'Antonio', 'Gómez Benítez', 19, 'Lepe', 'Huelva')
- grado1=('Ingeniería del Software', 11.184, 174, '814502')
- grado2=('Tecnología Informática', 9,941, 124, '823504')

Se accede a sus elementos con el operador slicing (*¡OJO! Se empieza por 0*). (*Desde el final con -1*):

- persona1[1] → 'Arancha' ; persona1[0] → '12345678A' ; persona2[5] → 'Huelva'
- grado1[1] → 11,184 ; grado1[3] → '814502' ; grado1[-1] → '814502'
- grado2[2] = 180 --- Proporciona un **error** de compilación (*inmutabilidad de las tuplas*)

Acceso a un carácter de una cadena que, a su vez, es un elemento de una tupla:

(de los más general a lo más particular)

- persona2[2][6] → 'B' ; persona2[-1][-1] → 'a'



Tuplas con nombres (namedtuple –contrucción de la tupla-)

*Para no tener que usar referencias indexadas (los corchetes y la posición) para acceder a los elementos de una tupla, se puede “nombrar” los campos mediante la función **NamedTuple** que asigna un nombre descriptivo a cada campo de una tupla y además especificamos el tipo de dato.*

NOTA:-→ Es necesario importar la función desde la biblioteca *typing*
`from typing import NamedTuple`

Sintaxis:

Nombre de la tupla=**NamedTuple**("nombre", [("nombrecampo1", tipo), ("nombrecampo2", tipo),...])

¡Ojo!: **NamedTuple** sólo tiene **2 parámetros** separados por una coma.

- El primero es el nombre con que se visualizará la tupla por la consola.
- El segundo, encierra entre corchetes las asociaciones de cada campo con su tipo.



Tuplas con nombres (namedtuple –contrucción de la tupla-)

Sintaxis:

Nombre de la tupla=**NamedTuple**("nombre", [("nombrecampo1", tipo), ("nombrecampo2", tipo),...])

Por ejemplo: Se definen los tipos *Persona* y *Grado*

- *Persona*=**NamedTuple**('persona', [("dni",str), ("nombre",str), ("apellidos",str), ("edad",int), ("localidad",str), ("provincia",str)])
- *Grado*=**NamedTuple**('grado', [("nombre",str), ("nota_corte",float), ("plazas", int), ("código",str)])

Cuando se crean las tuplas se antepone el nombre de la tupla

- persona1=*Persona*('12345678A', 'Arancha', 'López Martín',18, 'Sevilla', 'Sevilla')
- persona2=*Persona*('111222333B', 'Antonio', 'Gómez Benítez', 19, 'Lepe', 'Huelva')
- grado1=*Grado*('Ingeniería del Software', 11.184, 174, '814502'),
- grado2=*Grado*('Tecnología Informática', 9.941, 124, '823504')

Acceso a los campos de la tupla:

- persona1.nombre → 'Arancha' ; persona1.dni → '12345678A' ; persona2.provincia → 'Huelva'
- Grado1.nota_corte → 11,184 ; grado1.código → '814502'



Contenedores: “list()” (lista)

Es una variable que almacena todo tipo de datos, en un orden determinado.

Por ejemplo: una lista de números enteros, una lista de nombre de ciudades o una lista de tuplas que contienen determinados datos sobre equipos de futbol,...

Sintaxis para crear una lista: con la función *list()* o con corchetes *[]* y, en su caso, con los elementos separado por coma:

Ejemplos:

- lista_vacia = *list()* (este es *mi forma favorita* ★)
- lista_vacia2 = *[]*
- edades = *[23,17,21,17,30,23,11,7]* ← Es una lista de número enteros
- grados = *[Grado('Ingeniería del Software', 11.184, 174, '814502'), Grado('Tecnología Informática', 9.941, 124, '823504'), Grado('Ingeniería de Computadores, 9.425, 102, '815001'), Grado(...),...]* ← Es una lista de tuplas de Grados

Es muy normal utilizar una lista almacenar las tuplas que forman los registros de un fichero.



Contenedores: “list()” (lista)

Las listas son *mutables*: se puede añadir, eliminar un dato o modificarlo

- Se *añade* un *dato* al *final* de una lista con *append(dato)*
- Se *inserta* un *dato* en una *posición* de una lista con *insert(posición,dato)*
- Se *borra* un *dato* de una lista con *remove(dato)*. Borra la primera vez que lo encuentre
- Se *borra* un dato de una *posición* de una lista con *pop(posición)*
- Se *accede* a un dato en determinada posición con el operador slicing *[posición]*. ¡Empieza en cero!
- Se *conoce el número de elementos* de una lista con *len(nombre_lista)*

Ejemplo de acceso a un elemento:

edades=[23,17,21,17,30,23,11,7]

grados=[Grado('Ingeniería del Software', 11.184, 174, '814502'), Grado('Tecnología Informática', 9.941, 124, '823504'), Grado('Ingeniería de Computadores', 9.425, 102, '815001'), Grado(...),...]

- edades[1]→17 ; edades[-2]→11
- grados[1][1] o grados[1].*nota_corte* → 9.941
- grados[2][0] o grados[2].*nombre* → 'Ingeniería de Computadores'




Sentencia for (recorriendo una lista)

Permite ejecutar de forma iterada un bloque de sentencias desde el primero, hasta el último, de los elementos de una lista

Sintaxis :

for *variable* *in* nombre_lista:
 bloque sentencias




La *variable* va tomando, en cada iteración, el valor de cada elemento del contenedor

Ejemplo:

Se dispone de una lista de números *mi_lista*=[62,56,90,-22,20]

for *v* *in* *mi_lista*:
 print (*v*+1)



63
57
91
-21
21



Sentencia for (recorriendo una lista)

Ejemplo:

Se dispone de una lista denominada `grados`, con 3 tuplas de tipo *Grado*:

```
grados = [Grado('Ingeniería del Software', 11.184, 174, '814502'), Grado('Tecnología Informática', 9.941, 124, '823504'), Grado('Ingeniería de Computadores', 9.425, 102, '815001')]
```

Visualizando las tuplas completas:

```
for g in grados:  
    print(g)
```

```
grado(nombre='Ingeniería del Software', nota_corte=11.184,plazas=174,código='814502')  
grado(nombre='Tecnología Informática', nota_corte=9.941,plazas=124,código='823504')  
grado(nombre='Ingeniería de Computadores', nota_corte=9.425,plazas=102,código='815001')
```

Visualizando dos campos

```
for g in grados:  
    print(g.nombre, ': ', g.plazas)
```

```
for nom, not, pla, cód in grados:  
    print(nom, ': ', pla)
```

```
for nom, _, pla, _ in grados:  
    print(nom, ': ', pla)
```

```
'Ingeniería del Software' : 174  
'Tecnología Informática' : 124  
'Ingeniería de Computadores' : 102
```



Ejercicio

Se trata de hacer un proyecto “*T06_Datos_Personales*” con dos archivos:

- El primer archivo/módulo *datos_personales.py* contendrá dos funciones
 - “*filtra_por_edad*” que recibiendo como parámetros una lista de tuplas con los datos de personas y una edad, devuelva otra lista con las tuplas de las personas con menos edad que la dada.
 - “*obtiene_dni_y_nombres*” que recibiendo como parámetro una lista de tuplas con los datos de personas, devuelva otra lista con los nombres y los dni de todas las personas.
- El segundo archivo/módulo *test_datos_personales.py* contendrá las instrucciones necesarias para crear una lista “*lista_personas*” y probar las dos funciones.

Para crear la lista use los siguientes datos. (*tenga cuidado al copiar de ver como quedan los apóstrofes*)

Importante: Use el siguiente tipo y los siguientes datos

Persona=*NamedTuple*('persona', [(*"dni"*,str), (*"nombre"*,str), (*"apellidos"*,str), (*"edad"*,int), (*"localidad"*,str), (*"provincia"*,str)])

Datos para crear las personas	{	('12345678A', 'JUAN', 'AFAN POSTIGO', 22, 'SEVILLA', 'SEVILLA')
		('12345678B', 'NICOLAS', 'AGUILAR SAUCEDO', 20, 'DOS HERMANAS', 'SEVILLA')
		('12345678C', 'LUCAS', 'ACEJO GARCÍA', 20, 'UTRERA', 'SEVILLA')
		('12345678D', 'CLAUDIA', 'ÁLVAREZ GARCÍA', 21, 'VISO DEL ALCOR', 'SEVILLA')
		('12345678E', 'PAULA', 'ALBENDÍN CAMINO', 19, 'TOMARES', 'SEVILLA')
		('12345678F', 'ANA', 'LOBATO ÁLVAREZ', 18, 'PUNTA UMBRÍA', 'HUELVA')
		('12345678G', 'ANTONIO', 'DÍAZ NARANJO', 18, 'CHIPIONA', 'CADIZ')
		('12345678H', 'SOFÍA', 'GUERRERO CANTARERO', 20, 'CHIPIONA', 'CADIZ')