



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 → (...)
- Listas → [...]
- Conjuntos → {...}
- Diccionarios → {.: ..., .: ...}



Tipos contenedores: “set()” (conjuntos)

Variable que almacena todo tipo de datos

- No se puede acceder a los datos por su posición (*ya que no existe orden*)
- No puede haber elementos *repetidos*
- *Son mutables: Se puede añadir y eliminar un dato*

La sintaxis: Se crean con la función **set()** o usando **{}** pero debe contener elementos

Operaciones:

- *Se añade un dato* con `nombre_del_conjunto.add(dato)`
- *Se elimina un dato* con `nombre_del_conjunto.remove(dato)`
- *Para obtener los datos* de un conjunto hay que *recórrelo elemento a elemento* mediante `for`.



Tipos contenedores: “set()” (conjuntos)

Ejemplo de creación de un conjunto con elementos:

- conjunto_vacío = **set()**
- conjunto_vacío2={ } ← es un **ERROR** porque cuando se quiera añadir un elemento dirá que es un diccionario
- edades={ 23,17,21,17,30,23,11,7} → Realmente el conjunto almacena {17, 23, 21, 7, 11, 30}
- 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**(...),... }

Acceso a los elementos de un conjunto

- edades[1] ← **ERROR** No se puede acceder por la posición a un conjunto
- grados[0][1] ← **ERROR** No se puede acceder por la posición a un conjunto



Sentencia for (recorriendo un conjunto)

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_conjunto:
 bloque sentencias




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

Ejemplo:

Si se crea un conjunto de edades={ 23,17,21,17,30,23,11,7}

for *edad* *in* edades:
 print (*edad*)



17
23
21
7
11
30



Sentencia for (recorriendo un conjunto)

NO se puede recorrer un conjunto por la posición de sus elementos

Sintaxis:

(aunque aparentemente el esquema siguiente parecería correcto)

```
for pos in range(0,len(nombre_conjunto)):  
    bloque sentencias
```

La variable **pos** va tomando, en cada iteración, el valor que devuelve la secuencia

Ejemplo:

Se dispone de un conjunto de números `mi_conjunto=[62,56,90,-22,20]`

```
for pos in range(0,len(mi_conjunto)):
```

```
    print (mi_conjunto[pos])
```

← **ERROR** No se puede acceder por la posición a un conjunto



Sentencia for (recorriendo un conjunto)

Ejemplo:

Se dispone de un conjunto de 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')}
```

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

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

```
for nom, not, pl, código in grados:  
    print (nom, ': ', pl)
```

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

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



Ejercicio: Se trata de modificar el proyecto “*T08_Datos_Personales*” para:

- Añadir una función nueva al archivo/módulo *datos_personales.py* “*obtiene_número_edades_distintas*” que recibiendo como parámetros una lista de tuplas con los datos de personas, devuelva el número de edades distintas que hay en la lista.
- Modificar el archivo/módulo *test_datos_personales.py* para probar la nueva función