Bloque 1

Contenedores: Algoritmo de recorridos, Conjuntos y Operaciones

Fundamentos de Programación
Departamento de Lenguajes y Sistemas Informáticos

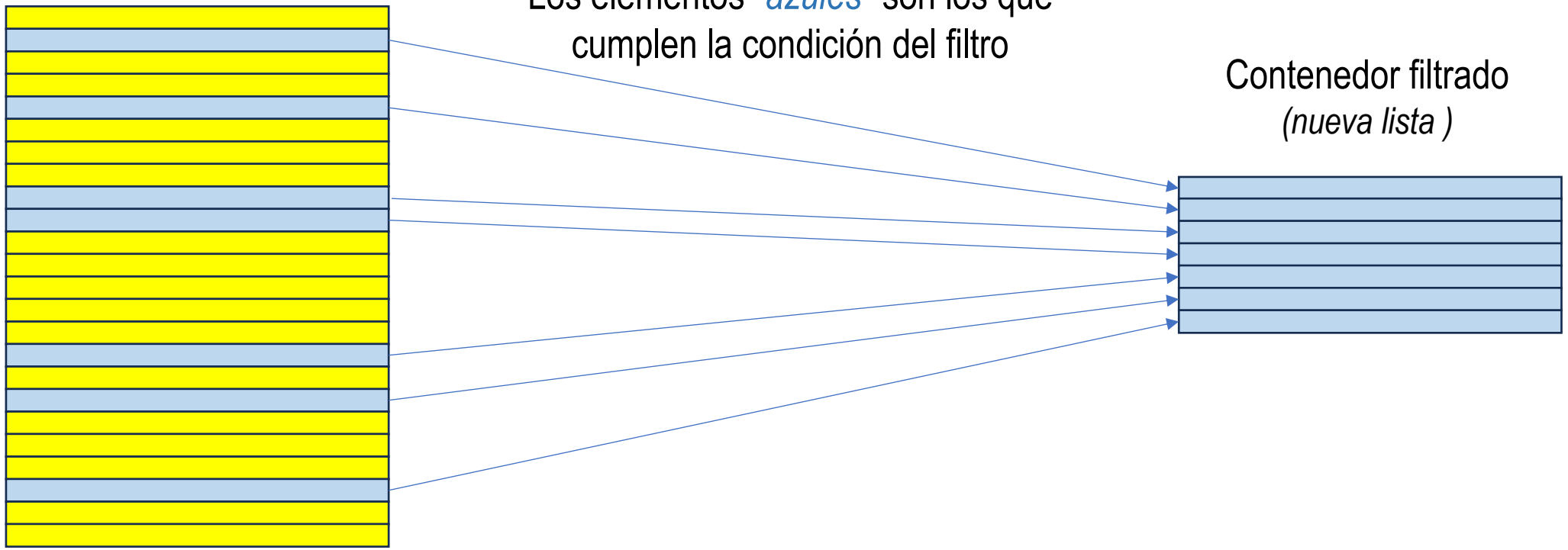
2024/2025

Algoritmia: Filtrar un contenedor (algunos elementos)

Contenedor a filtrar
(Por ejemplo: una lista)

Los elementos "*azules*" son los que cumplen la condición del filtro

Contenedor filtrado
(nueva lista)



Algoritmia: Filtrar un contenedor (algunos elementos)

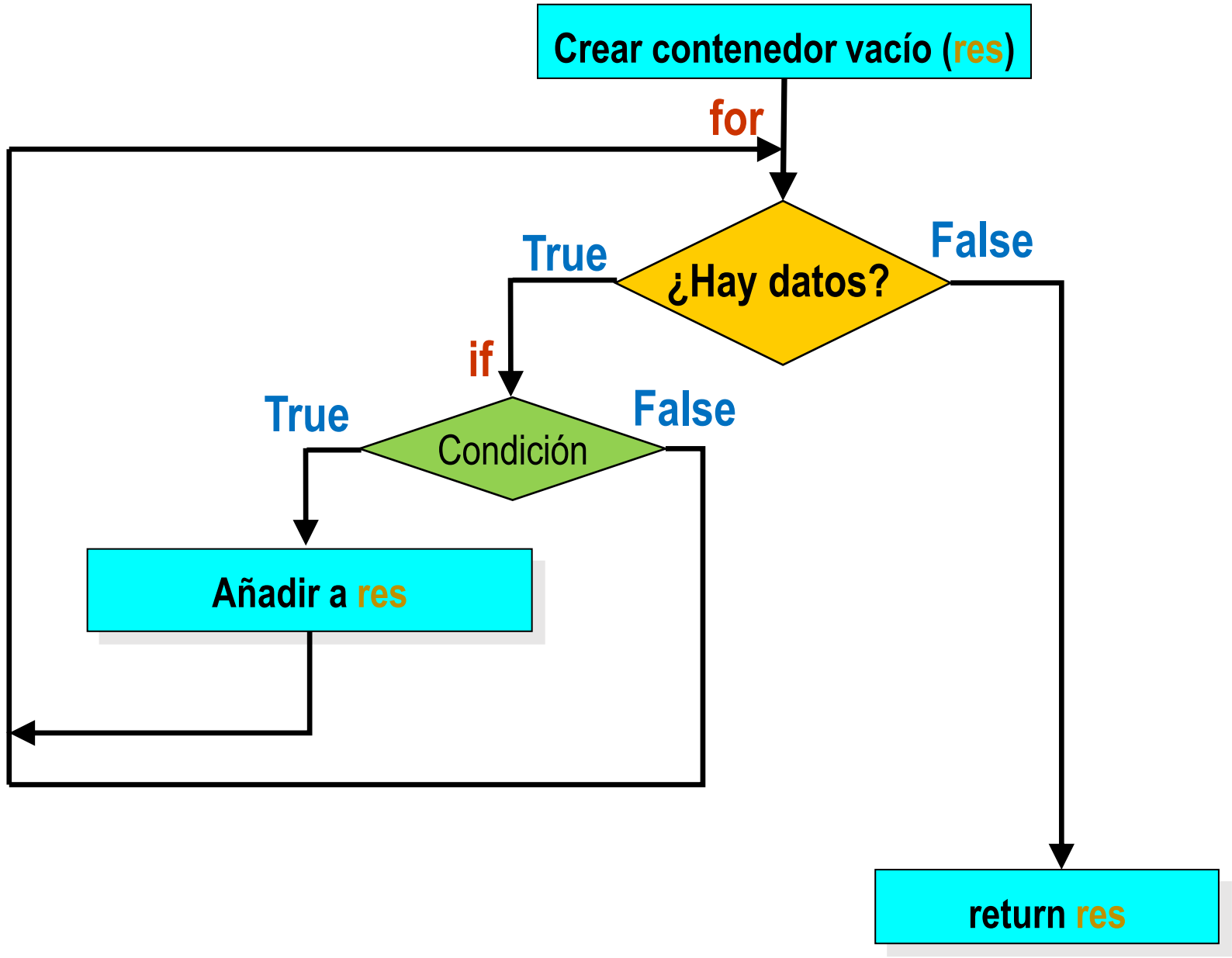
Denominamos normalmente *filtro* al proceso por el que se selecciona determinados elementos de un contenedor que cumplan determinada *condición*. Generalmente lista o conjuntos.

El proceso es sencillo y consiste en:

1. *Definir una función* que reciba en los parámetros formales:
   - El *contenedor*
   - El/los parámetro/s en los que recibir los valores por los que hay que filtrar
2. *Crear un nuevo contenedor vacío* (*res*), en el que se almacenarán los elementos que cumplen la *condición*.
3. *Ejecutar* una sentencia *for* sobre el *contenedor* de entrada,
4. Dentro del bloque de sentencias del for implementar una sentencia if en la que se evalúe la *condición* para filtrar los elemento que la cumplen y en ese caso *añadir* a la variable (*res*) el elemento.
5. *Por último, devolver* el contenedor (*res*): *return res*

Algoritmia: Filtrar un contenedor

for que recorre el contenedor a filtrar



Algoritmia: Filtrar un contenedor. Ejercicio

Se trata de realizar el proyecto "*T07_Población*" que contenga en carpeta *src* los dos archivos:

- El primer archivo/módulo *población.py* contendrá dos funciones
  - *lee_poblaciones* que cree y devuelva una lista de tuplas de tipo *Población* con los datos de la siguiente diapositiva. Esta función no recibe parámetros y devuelve el tipo List[*Población*]
  - *filtra_por_país* que reciba dos parámetros: una lista de tuplas de tipo *Población* con los datos de poblaciones, un país de tipo *str*, y devuelva otra lista con las tuplas de las poblaciones que son del país dado..
  - *filtra_por_año_y_umbral_de_censo* que reciba tres parámetros: una lista de tuplas de tipo *Población* con los datos de poblaciones, un año de tipo *int* y un umbral de tipo *int*, y devuelva otra lista con las tuplas de las poblaciones que en el año dado tuvieron un censo superior al umbral dado..
- El segundo archivo/módulo *test_población.py* contendrá, las instrucciones necesarias para probar las funciones del módulo *población.py* . *(Copie el test de dos diapositivas más adelante)*

Algoritmia: Filtrar un contenedor

Debe definir el tipo *Población* con los siguientes campos y tipos para poder nombrar las tuplas: *país* de tipo *str*, *código* de tipo *str*, *año* de tipo *int*, *censo* de tipo *int*. Importe también de *typing List*

```
[Población('Austria','AUT',2000,8011566),
Población('Austria','AUT',2001,8042293),
Población('Austria','AUT',2002,8081957),
Población('Haiti','HTI',2000,8549200),
Población('Haiti','HTI',2001,8692567),
Población('Haiti','HTI',2002,8834733),
Población('Monaco','MCO',2000,32082),
Población('Monaco','MCO',2001,32360),
Población('Monaco','MCO',2002,32629),
Población('Spain','ESP',2000,40567864),
Población('Spain','ESP',2001,40850412),
Población('Spain','ESP',2002,41431558),
Población('Spain','ESP',2003,42187645),
Población('Portugal','PRT',2000,10289898),
Población('Portugal','PRT',2001,10362722),
Población('Portugal','PRT',2002,10419631),
Población('United States','USA',2000,282162411),
Población('United States','USA',2001,284968955),
Población('United States','USA',2002,287625193)]
```

Estructura estándar de un módulo test

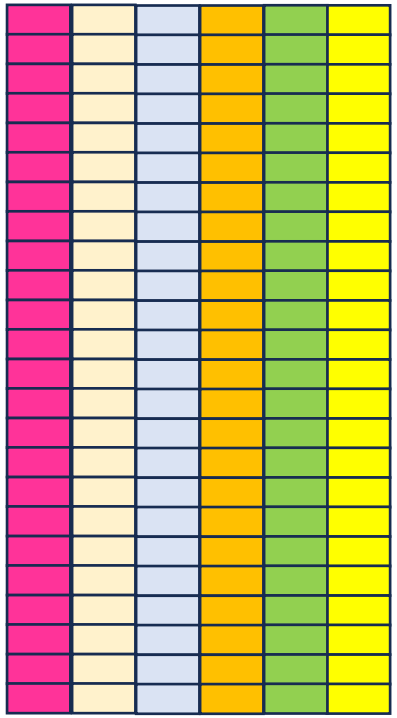
```
from población import lee_poblaciones, filtra_por_país, filtra_por_año_y_umbral_de_censo
def test_lee_poblaciones(datos:List[Población])->None:
  print("\nTotal poblaciones leídas:",len(datos))
  print("Las dos primeras: ",datos[:2])
  print("Las dos últimas:",datos[-2:])

def test_filtra_por_país(datos:List[Población])->None:
  print("\nfiltra_por_país")
  poblaciones_filtradas=filtra_por_país(datos,"Spain")
  for p in poblaciones_filtradas:
    print(p)

def test_filtra_por_año_y_umbral_de_censo(datos:List[Población])->None:
  print("\nfiltra_por_año_y_umbral_de_censo")
  poblaciones_filtradas=filtra_por_año_y_umbral_de_censo(datos,2002,40000000)
  for p in poblaciones_filtradas:
    print(p)
if __name__=='__main__':
  poblaciones=lee_poblaciones()
  test_lee_poblaciones(poblaciones)
  test_filtra_por_país(poblaciones)
  test_filtra_por_año_y_umbral_de_censo(poblaciones)
```

El módulo empieza a ejecutarse por aquí y continua hacía abajo

"poblaciones" almacena los datos leídos

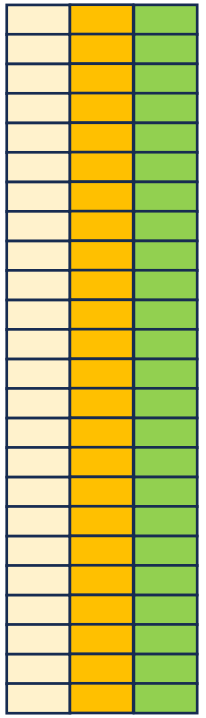
Algoritmia: Obtener una vista (algunos campos)

Contenedor
(p.e.: una lista con 6 campos)



Se trata de obtener un nuevo contendor con el mismo número de elementos, pero con determinados campos *(p.e.: tres)*

Contenedor con la vista
(p.e.:nueva lista con tres campos)



Algoritmia: Obtener una vista (algunos campos)

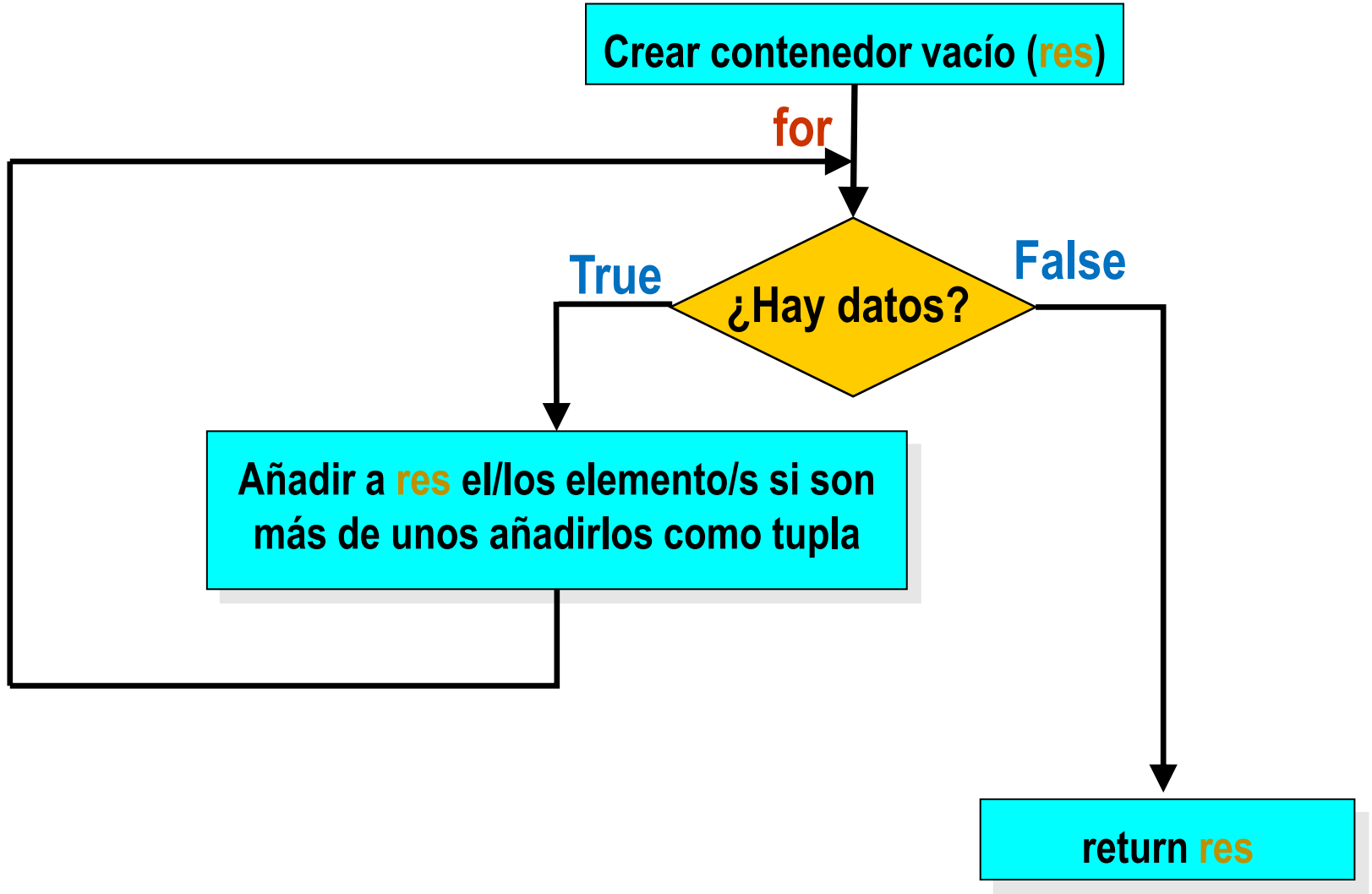
Denominamos normalmente "*obtener una vista*" al proceso por el que se selecciona determinados campos de los elementos de un contenedor. Generalmente lista o conjuntos.

El proceso es sencillo y consiste en:

1. *Definir una función* que reciba en los parámetros formales:
   - El *contenedor*
2. *Crear un contenedor* (*res*), en el que se almacenarán el/los campo/s que hay que devolver en la vista. *Si hay más de un campo se agrupan en una tupla*
3. *Ejecutar* una sentencia *for* sobre el *contenedor*.
4. En el bloque de sentencias del for, *añadir* al contendor (*res*) el/los elemento/s que corresponda. Si son más de uno agruparlos en una tupla antes de *añadir*
5. *Por último, devolver* el contenedor (*res*): *return res*

Algoritmia: Obtener una vista (algunos campos)

for que recorre el contenedor con todos los campos



Algoritmia: Obtener una vista. Ejercicio

Se trata de modificar el proyecto "*T07_Población*" :

- Añadir al módulo *población.py* la función
  - *obtiene_país_y_censo,* que recibiendo como parámetro una lista de tuplas de tipo *Población* con los datos de poblaciones, y devuelva otra lista con tuplas que contengas el nombre del país y el censo..
  - *obtiene_código_y_censo_de_año,* que reciba dos parámetros: una lista de tuplas de tipo *Población* con los datos de poblaciones, un año de tipo *int*, y devuelva otra lista con tuplas que contenga el código y el censo del año dado . *Nota.- Esta función mezcla filtrado y vista*
- Modificar *test_población.* para probar las dos funciones anteriores.

Tipos contenedores: “set()” (conjuntos)

Variable que almacena todo tipo de datos (como las listas), pero con las siguientes diferencias:

- ***No se puede acceder*** a los datos por *la posición* (*no existe orden*)
- *No* puede haber elementos *repetidos*

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

Ejemplos:

- conjunto_vacío =*set()* *(este es mi forma favorita*  *)*
- conjunto_vacío2 =*{ }* ← *No vale. Interpreta que es un diccionario*
- edades=*{* 23,17,21,17,30,23,11,7*}* ← *Es un conjunto de número enteros. No entran el segundo 17 y 23*
- 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 un conjunto de tuplas de Grados*

Tipos contenedores: “set()” (conjuntos)

Los conjuntos son *mutables*: se puede añadir o eliminar un dato.

- *Se añade un dato a un conjunto con add(dato)*
- *Se borra un dato de un conjunto con remove(dato). Borra una ocurrencia*
- *Se conoce el número de elementos de un conjunto con len(nombre_conjunto)*
- *Para acceder a los elementos de un conjunto hay que recorrerlo mediante un for*

Observar:

```
conjunto={1,3,4,7,3,8,7}
print(len(conjunto))
```

→ visualiza 5 (*los repetidos no entran*)

edades[1] ← *ERROR No se puede acceder por la posición a un elemento de un conjunto*

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

Operadores sobre contenedores:

- Operador *in*: *Devuelve True o False según el contenedor contenga, o no, al elemento*

Sintaxis:

elemento *in* *contenedor:*

Ejemplos

- *Si, mi_lista=[('a',62),('b',56),('c',90),('d',-22),('e',20)]*
  ('c',90) in mi_lista → True
  ('d',21) in mi_lista → False
- *Si, mi_tupla=('a',62,'b',56,'c',90)*
  56 in mi_tupla → True
  'A' in mi_tupla → False
- *Si, mi_conjunto={"gallo",62,"perro","foca",'c',2.89}*
  "foca" in mi_conjunto → True
  2 in mi_conjunto → False

Operadores sobre contenedores

Operador **+**: *Al igual que se puede aplicar a las string para concatenarlas, se puede aplicar a las tuplas, o a las listas (para conjuntos no está permitido este operador).*

¡OJO! tiene que ser de mismo tipo (string con string, tupla con tupla o lista con lista)

Ejemplos:

Concatenando tuplas:

```
tupla1=('a',62,'b',56,'c',90)
tupla2=(20,'h','p',"perro","foca")
print(tupla1 + tupla2) → ('a',62,'b',56,'c',90,20,'h','p','perro','foca')
```

Observar los paréntesis de tupla

Concatenando listas:

```
l1=[1,'Antonio',23,'perro']
l2=[23.7,22.3,"gato","Ana"]
l3=l1+l2
print(l3) → [1, 'Antonio', 23, 'perro', 23.7, 22.3, 'gato', 'Ana']
```

Observar los corchetes de lista

Algoritmia: Obtener una vista. Ejercicio (para casa)

Se trata de modificar el proyecto "*T07_Población*" :

- Añadir al módulo *población.py* la función
  - *suma_población_de_año*, que reciba dos parámetros: una lista de tuplas de tipo *Población* con los datos de poblaciones y un año de tipo *int*, devuelva la suma del censo de ese año.
  - *promedio_ población_de_año*, que reciba dos parámetros: una lista de tuplas de tipo *Población* con los datos de poblaciones y un año de tipo *int*, devuelva el promedio del censo de ese año . *Nota.- Si no hubiese población para el año dado, debe devolver None*
- Modificar *test_población.* para probar las dos funciones anteriores.. Pruebe ambas funciones con dos años: primero con el *2000* y después con el *2025*.