Diccionarios

Es un tipo contenedor *mutable* que almacena ***pares*** de valores que se denominan *clave* y *valor*:
En general sirven para *agrupar o contabilizar* determinados datos *por alguno* de los distintos valores *de los elementos* de un contenedor.

Se inicializan con dict() o { }, esta última ({ }) es la misma sintaxis que para los conjuntos. Por defecto define un diccionario y sólo será un conjunto si se inicializa como conjunto incluyendo elementos separados por coma (,).

En un diccionario las *claves* se separan de los *valores* por dos puntos (:)

Así tenemos que:
dic=dict() → aux es un diccionario
dic={ } → aux es un diccionario
dic={"TI4": 90, "TI2": 51, "IS3": 84, "TI3": 72,…} → dic es un diccionario (*claves* los grupos, *valores* el número de alumnos)
dic={"Pepe", "Juana", "Margarita", "Antonio", …} → dic es un conjunto

Diccionarios

Las *claves* y los *valores* pueden ser de *cualquier tipo* de datos estudiados con la restricción que las *claves* *no pueden ser contenedores* *(listas, conjuntos o diccionarios*, aunque si pueden ser tuplas).

Ejemplo 1: que asocia a personas con su edad.

dicci1={"Eva": 41, "Manuel": 11, "Javier": 24, "Ana": 28}

- Las *claves* aparecen a la izquierda de los dos puntos (:) →Eva, Manuel,…
- Los *valores* aparecen a la derecha 41, 11, …

Ejemplo 2: Asocia a personas con tuplas que contienen su edad y provincia de nacimiento

dicci2={"Eva": (41, "Cádiz"), "Manuel": (11, "Sevilla"), "Javier": (24, "Murcia"), "Ana": (28, "Lugo")}

- Las *claves* aparecen a la izquierda de los dos puntos (:) →Eva, Manuel,…
- Los *valores* aparecen a la derecha y en este caso son tuplas (41, "Cádiz"), (11, "Sevilla"), …

Importante: Los diccionarios no tienen *claves* repetidas

Diccionarios

Acceso a los valores de un diccionario. Existen dos formas:

Si dicci2={"Eva": (41, "Cádiz"), "Manuel": (11, "Sevilla"), "Javier": (24, "Murcia"), "Ana": (28, "Lugo")}

1. Con el operador *[ ]* través de las *claves*
   - dicci2["Manuel"]→(11,"Sevilla")
   - dicci2["Javier"]→(24,"Murcia")
   - dicci2["José"]→ KeyError (*devuelve error porque no existe la clave "José". El programa aborta)*

2. Con el método *get( )* a través de las *claves*
   - dicci2.get ("Manuel")→(11,"Sevilla")
   - dicci2.get ("Javier")→(24,"Murcia")
   - dicci2.get ("José")→ Por defecto devuelve *None* y no aborta.

   No obstante, se puede indicar un valor (*como 2º parámetro*), para el caso de que no exista la *clave*.

   get (clave, valor por si no existe la clave) :
   - dicci2.get ("Manuel", False)→(11,"Sevilla")
   - dicci2.get ("José", 0)→ 0
   - dicci2.get ("José", False)→ False

Diccionarios

Inserción o modificación de un valor en un diccionario

Para inserta una nueva pareja o modificar el *valor* de una *clave* ya existente, se usa el operador [nueva_clave] =valor o [clave_existente] =nuevo_valor

Ejemplo:

Si dicci1={"Eva": 41, "Manuel": 11, "Javier": 24, "Ana": 28}

- dicc1["Isa"]=20→{"Eva": 41, "Manuel": 11, "Javier": 24, "Ana": 28, **"Isa": 20**}
- dicc1["Javier"]=50→{"Eva": 41, "Manuel": 11, "**Javier**": **50**, "Ana": 28, "Isa": 20}

Otra forma de crear es utilizando el método *update( )*

diccionario2.*update(*diccionario1*)* → Hace una copia del diccionario 1 en el 2. Resultado final: dos diccionarios iguales.

Diccionarios

Borrar una pareja de un diccionario

Para borrar una pareja se utiliza la función *del()*, con la siguiente sintaxis:

del(diccionario[*clave*]).

Importante si la clave no existe el programa aborta, por lo que es recomendable utilizar previamente el método *get()* para comprobar si existe o no la clave.

Ejemplo:

Si dicci1={"Eva": 41, "Manuel": 11, "Javier": 24, "Ana": 28}

- *del*(dicci1["Manuel"]) → {"Eva": 41, "Javier": 24, "Ana": 28}
- *del*(dicci1["José"]) → KeyError: 'José'

Esquema de borrado que no aborta el programa

```
if dicci1.get(clave, False) != False:
    del(dicci1[clave])
…
```

Diccionarios

Limpiar un diccionario

Para limpiar y dejar vacío un diccionario se utiliza el método *clear()*, con la siguiente sintaxis:

diccionario. *clear()*

***Ejemplo**:*

Si dicci1={"Eva": 41, "Manuel": 11, "Javier": 24, "Ana": 28}

- *dicc1.clear()* → { }

Diccionarios

- *Función len()*: Devuelve el número de parejas que hay en el diccionario (*coincide con el número de claves*)

Ejemplo:

Si dicci1={"Eva": 41, "Manuel": 11, "Javier": 24, "Ana": 28}

- *len(*dicci1*)* → 4

Diccionarios

- *Método items()*: Devuelve *una tupla* con un solo elemento con *namedtuple dict_ítems. Dicho elemento es* una lista formada por todas las parejas de diccionario en formas de tuplas

Ejemplo:

Si dicci1={"Eva": 41, "Manuel": 11, "Javier": 24, "Ana": 28}

- *dicci1.items()* → *dict_ítems([('Eva', 41), ('Manuel', 11), ('Javier', 24), ('Ana', 28)]) observar los paréntesis externos que indica que es una tupla.*

Se puede utilizar para obtener una lista (con list()) o un conjunto (con set()):

- *list(dicc1.ítems()) → [('Eva', 41), ('Manuel', 11), ('Javier', 24), ('Ana', 28)] !lista!*
- *set(dicc1.ítems()) → {('Ana', 28),('Eva', 41), ('Manuel', 11), ('Javier', 24)} ¡conjunto!*

ítems() permite recorrer directamente el diccionario como si fuese una lista con tuplas de dos elementos: el primero la clave y el segundo el valor

Diccionarios

- *Método keys()*: Devuelve *una tupla* con un solo elemento con *namedtuple dict_keys* con una lista formada por todas las claves

Ejemplo:

Si dicci1={"Eva": 41, "Manuel": 11, "Javier": 24, "Ana": 28}

- *dicci1.keys() → dict_keys(['Eva', 'Manuel', 'Javier', 'Ana']) observar los paréntesis externos*

Se puede utilizar para obtener una lista o un conjunto (con list() o con set()):

- *list(dicc1.keys()) → ['Eva', 'Manuel', 'Javier', 'Ana'] !lista!*
- *set(dicc1. keys()) → {'Ana', 'Javier', 'Eva', 'Manuel'} ¡conjunto!*

keys() permite recorrer directamente las claves como una lista

Diccionarios

- *Método values()*: Devuelve una tupla con un solo elemento con *namedtuple dict_values* con una lista formada por todos los valores

Ejemplo:

Si mi_dicci1={"Eva": 41, "Manuel": 11, "Javier": 24, "Ana": 28}

- *mi_dicci1.values() → dict_values([41, 11, 24, 28]) observar los paréntesis externos*

Se puede utilizar para obtener una lista o un conjunto (con list() o con set()): :

- *list(mi_dicc1.values()) → [41, 11, 24, 28] !lista!*
- *set(mi_dicc1. values()) → {24, 41, 11, 28} ¡conjunto!*

No obstante, values() permite recorrer directamente los valores como una lista

Diccionarios: COUNTER

La función *Counter* (de la librería *collections*) permite crear un diccionario que asigna a cada elemento de un contenedor la frecuencia con la que aparece en mismo.

Si, lista=[12, 2, 4, 2, 12, 4, 5, 7, 8, 5, 4, 3, 10, 10, 10, 3, 3, 34, 7, 8, 9]

from *collections* import *Counter*
diccionario2=*Counter* (lista)

diccionario2 → {12: 2, 2: 2, 4: 3, 5: 2, 7: 2, 8: 2, 3: 3, 10: 3, 34: 1, 9: 1}

Hemos dicho que para cualquier diccionario
diccionario2[4] → 3 pero diccionario2[6] → *Key Error*

Sin embargo, cuando el diccionario se crea con *Counter*():
diccionario2[4] → 3 pero diccionario2[6] → *0 (no da error, devuelve que hay 0 elementos)*