


# Diccionarios Operadores Esquema de construcción



Fundamentos de Programación  
Departamento de Lenguajes y Sistemas Informáticos



## 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=**{"T14": 90, "T12": 51, "IS3": 84, "T13": 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, ...

*dicci1*

| Claves   | Valores<br>(números) |
|----------|----------------------|
| "Eva"    | 41                   |
| "Manuel" | 11                   |
| "Javier" | 24                   |
| "Ana"    | 28                   |



## Diccionarios

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"), ...

dicci2

| Claves   | Valores<br>(tuplas) |
|----------|---------------------|
| "Eva"    | → (41, "Cádiz")     |
| "Manuel" | → (11, "Sevilla")   |
| "Javier" | → (24, "Murcia")    |
| "Ana"    | → (28, "Lugo")      |

**Importante:** Los diccionarios no tienen **claves** repetidas



## Diccionarios (operaciones)

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 (operaciones)

### 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 dicc1={"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}



## Diccionarios (operaciones)

### 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 conveniente utilizar previamente el método *get()* para asegurar 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 (operaciones)

### 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}

- `dicci1.clear()` → {}

### Número de elementos (parejas) de un diccionario

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 (recorrido)

Método **items()**: Devuelve una tupla con un solo elemento con *namedtuple* `dict_items`. 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_items([('Eva', 41), ('Manuel', 11), ('Javier', 24), ('Ana', 28)])` observar los *paréntesis* externos que indica que es una tupla. No obstante, esa tupla `( )` contiene una lista – observar `[ ]` – de tuplas de las parejas. Por tanto, se pueden recorrer como iterable (con `for` o `while`)

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

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

**items()** 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 (recorrido)

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 *-( )*- y a su vez, los *[ ]* que permite iterar sobre las claves (con un for o while)

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 (recorrido)

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 `-( )` y a su vez, los `[ ]` que permite iterar sobre los valores (con un for o while)

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!

**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.

### Ejemplo:

Si, `mi_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  
mi_dicc=Counter(mi_lista)
```

`mi_dicc` → {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 por defecto cuando no existe la clave devuelve error

`mi_dicc[4]` → 3                      pero `mi_dicc[6]` → *Key Error*

Sin embargo, cuando el diccionario se crea con *Counter*():

`mi_dicc[4]` → 3                      pero `mi_dicc[6]` → 0 (no da error, devuelve que hay 0 elementos)



# Diccionarios (uso habitual).

## ¿En qué nos ayuda el uso de diccionarios?: A agrupar información

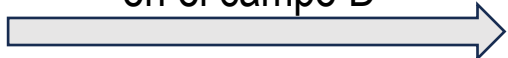
Contenedor  
(p.e.: una lista con 6 campos)

|  |  |    |  |      |      |
|--|--|----|--|------|------|
|  |  | D1 |  | v11  | v12  |
|  |  | D2 |  | v21  | v22  |
|  |  | D3 |  | v31  | v32  |
|  |  | D4 |  | v41  | v42  |
|  |  | D2 |  | v51  | v52  |
|  |  | D2 |  | v61  | v62  |
|  |  | D4 |  | v71  | v72  |
|  |  | D1 |  | v81  | v82  |
|  |  | D5 |  | v91  | v92  |
|  |  | D2 |  | v101 | v102 |
|  |  | D1 |  | v111 | v112 |
|  |  | D3 |  | v121 | v122 |
|  |  | D2 |  | v131 | v132 |
|  |  | D3 |  | v141 | v142 |
|  |  | D1 |  | v151 | v152 |
|  |  | D4 |  | v161 | v162 |
|  |  | D4 |  | v171 | v172 |
|  |  | D2 |  | v181 | v182 |
|  |  | D5 |  | v191 | v192 |
|  |  | D1 |  | v201 | v202 |
|  |  | D3 |  | v211 | v212 |
|  |  | D4 |  | v221 | v222 |
|  |  | D2 |  | v231 | v232 |
|  |  | D2 |  | v241 | v242 |

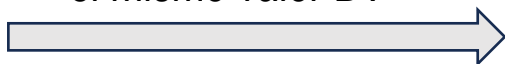
Obtener **una lista** con los registros que tienen el mismo valor en el campo D



Obtener **una lista** con los campos v\*1 que tienen el mismo valor en el campo D



¿Cuántos elementos tienen el mismo valor D?



### Diccionario

Claves

Valores (listas con el registro completo)

|    |  |    |  |     |     |  |    |  |      |      |  |    |  |      |      |     |
|----|--|----|--|-----|-----|--|----|--|------|------|--|----|--|------|------|-----|
| D1 |  | D1 |  | v11 | v12 |  | D1 |  | v81  | v82  |  | D1 |  | v111 | v112 | ... |
| D5 |  | D5 |  | v91 | v92 |  | D5 |  | v191 | v192 |  | D5 |  | v191 | v192 | ... |
| D4 |  | D4 |  | v41 | v42 |  | D4 |  | v71  | v72  |  | D4 |  | v161 | v162 | ... |
| D2 |  | D2 |  | v21 | v22 |  | D2 |  | v51  | v52  |  | D2 |  | v61  | v62  | ... |
| D3 |  | D3 |  | v31 | v32 |  | D3 |  | v121 | v122 |  | D3 |  | v141 | v142 | ... |

### Diccionario

Claves

Valores (listas con un campo)

|    |  |     |      |      |      |      |
|----|--|-----|------|------|------|------|
| D1 |  | v11 | v81  | v111 | v151 | v201 |
| D5 |  | v91 | v191 |      |      |      |
| D4 |  | v41 | v71  | v161 | v171 | v221 |
| D2 |  | v21 | v51  | v61  | v101 | v131 |
| D3 |  | v31 | v121 | v141 | v211 |      |
|    |  |     |      |      | v181 | v231 |
|    |  |     |      |      |      | v241 |

### Diccionario

Valores  
Claves (números)

|    |   |   |
|----|---|---|
| D1 | → | ⑤ |
| D5 | → | ② |
| D4 | → | ⑤ |
| D2 | → | ⑧ |
| D3 | → | ④ |



## Construcción de Diccionarios (contar)

Algoritmo de construcción de diccionarios a partir de un contenedor (**Esquema 1**):

Los pasos son los siguientes:

1. Se va recorriendo el contenedor (con un `for`)
2. Se consulta si el elemento que será la clave ya está en el diccionario (con un `if ... not in...`)  
Si no está se inserta una nueva pareja clave-valor con el valor neutro de la operación  
`diccionario[clave]=valor_neutro`
3. En todo caso, se actualiza el valor: `diccionario[clave]=actualización del valor`

Ejemplo: Para la lista `lista=['a','b','c','a','b','b','z','a','b','c']`, contar cuantas veces aparece cada valor

```
dic=dict()
```

```
for letra in lista:
```

```
    if letra not in dic:
```

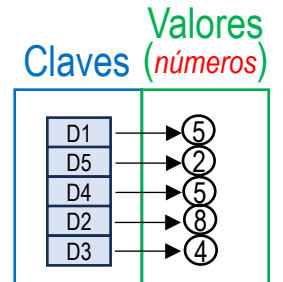
```
        dic[letra]=0
```

//Comienzo la cuenta. Elemento neutro de la suma

```
        dic[letra]+=1
```

//actualiza la cuenta. Es lo mismo que `dic[letra]=dic[letra]+1`

Resultado: `dic` → { 'a': 3, 'b': 4, 'c': 2, 'z': 1 }





## Construcción de Diccionarios (contar)

Algoritmo de construcción de diccionarios a partir de un contenedor (**Esquema 2**):

Los pasos son los siguientes:

1. Se va recorriendo el contenedor (con un `for`)
2. Se consulta si el elemento que será la clave ya está en el diccionario (con un `if ... not in...`)
  - a) Si no está se inserta una nueva pareja clave-valor : `diccionario[clave]=valor`
  - b) Si está se actualiza el valor: `diccionario[clave]=nuevo_valor`

Ejemplo: Para la lista=['a','b','c','a','b','b','z','a','b','c'], contar cuantas veces aparece cada valor

```
dic=dict()
```

```
for letra in lista :
```

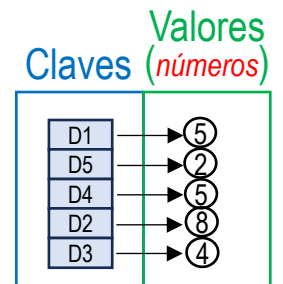
```
    if letra not in dic:
```

```
        dic[letra]=1           //Inicializa la cuenta
```

```
    else:
```

```
        dic[letra]+=1         //actualiza la cuenta. Es lo mismo que dic[letra]=dic[letra]+1
```

Resultado: `dic` → { 'a': 3, 'b': 4, 'c': 2, 'z': 1 }





## Construcción de Diccionarios (listas)

Algoritmo de construcción de diccionarios a partir de un contenedor (**Esquema 1**):

### Ejemplo:

```
Par=NamedTuple("pareja", "letra, número")
```

```
lista=[Par("z",21), Par("a",62), Par("J",7), Par("b",56), Par("c",90), Par("z",21), Par("a",1),  
        Par("b",10), Par("b",2), Par("a",21)]
```

Se trata de obtener un diccionario que a cada letra **le haga corresponder una lista** con los valores asociados

```
dic=dict()
```

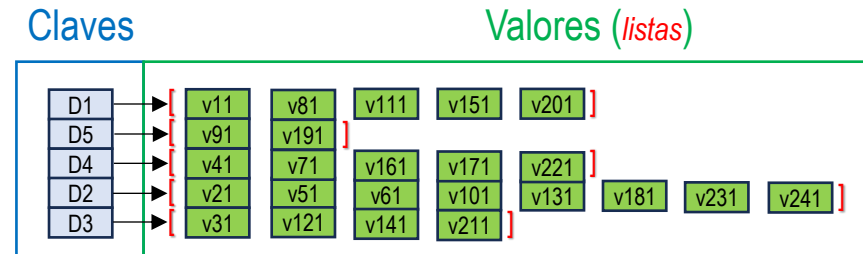
```
for p in lista:
```

```
    if p.letra not in dic:
```

```
        dic[p.letra]=list()           // o bien =[ ]. Valor neutro de listas (una lista vacía)
```

```
        dic[p.letra].append(p.número) // esto es lo mismo que dic[p.letra]+=[p.número]
```

Resultado: `dic`  $\rightarrow$  `{'z': [21, 21], 'a': [62, 1, 21], 'J': [7], 'b': [56, 10, 2], 'c': [90]}`







## Construcción de Diccionarios (listas)

Algoritmo de construcción de diccionarios a partir de un contenedor (**Esquema 2**):

### Ejemplo:

```
Par=NamedTuple("pareja", "letra, número")
```

```
lista=[Par("z",21), Par("a",62), Par("J",7), Par("b",56), Par("c",90), Par("z",21), Par("a",1),  
        Par("b",10), Par("b",2), Par("a",21)]
```

Se trata de obtener un diccionario que a cada letra **le haga corresponder una lista** con los valores asociados

```
dic=dict()
```

```
for p in lista:
```

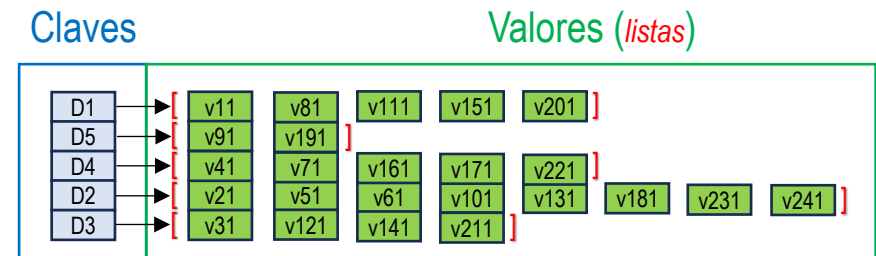
```
    if p.letra not in dic:
```

```
        dic[p.letra]=[p.número]           // o bien =[ ]. Valor neutro de listas (una lista vacía)
```

```
    else:
```

```
        dic[p.letra].append(p.número)    // esto es lo mismo que dic[p.letra]+=[p.número]
```

Resultado: `dic` → `{'z': [21, 21], 'a': [62, 1, 21], 'J': [7], 'b': [56, 10, 2], 'c': [90]}`





## Ejercicio:

*Proyecto L09\_ITV:*

- Descargue el fichero de enunciados “*README\_v2.md*”
- *Cópielo en el proyecto.*
- *Realice los ejercicios 7, 8 y 9*