Tema 7.
Tipos estructurados (diccionarios)

Curso 2020/2021

Programación

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

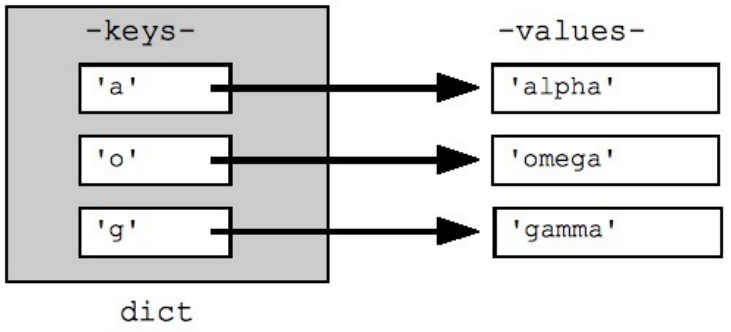
Diccionarios

- Un ***diccionario*** es una correspondencia entre ***claves*** y ***valores***. Ejemplos:
  - Un diccionario español-inglés asocia palabras españolas (claves) con palabras inglesas (valores)
  - Un listín telefónico asocia nombres (claves) con números de teléfono (valores)
  - Una agenda asocia fechas (claves) con citas (valores)
- Las claves de un diccionario han de ser valores de algún tipo *inmutable*: cadenas, números enteros, números reales o tuplas
- Un diccionario vacío (sin ninguna asociación de momento) se define con un par de llaves:

```
d = {}
```

- `Ejemplo:`

```
dict = {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}
```



Crear un diccionario

```
#directamente
diccionario = {'nombre' : 'Carlos',
               'edad' : 22,
               'cursos': ['Python','Django','JavaScript']
               }

#uno por uno
diccionario2 = {}
diccionario2['nombre'] = 'Carlos'
diccionario2['edad'] = 22
diccionario2['cursos'] = ['Python','Django','JavaScript']
```

Diccionarios

- Para añadir elementos a un diccionario, la notación es similar a la de las listas (solo que no es preciso que los índices sean números enteros dentro de un rango)
- Ejemplo: listín telefónico

```
d = {}
d['Pepe'] = '640 123 456'
d['Ana'] = '640 234 567'
d['Luis'] = '640 345 678'
```

- Si deseamos consultar un teléfono, solo tenemos que indexar por el nombre (clave):

```
print(d['Ana'])
```

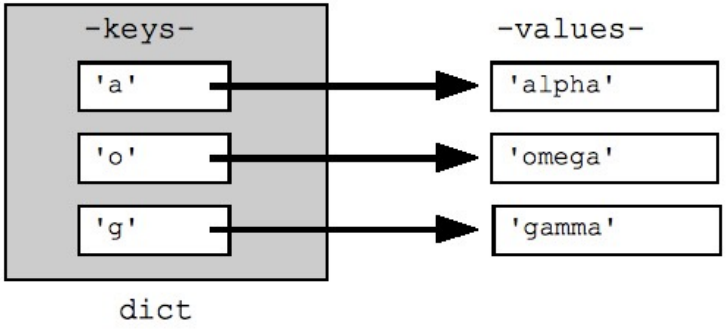
```
640 234 567
```

Diccionarios

- Si tratamos de acceder a un elemento inexistente se produce un error de tipo *KeyError*

```
>>> dict = {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}
>>> dict['a']
'alpha'

>>>
>>> dict['b']
  Traceback (most recent call last):
    File "<pyshell>", line 1, in <module>
  KeyError: 'b'
```



Diccionarios

- Podemos usar el operador **`in`** para averiguar si una clave existe o no en un diccionario:

```
'Luis' in d
```

```
True
```

```
'Pedro' in d
```

```
False
```

- También podemos usar `in` en los bucles `for` para recorrer un diccionario:

```
for clave in d:
    print(clave, '->', d[clave])
```

```
Pepe -> 640 123 456
Ana -> 640 234 567
Luis -> 640 345 678
```

Diccionarios

- Podemos obtener la lista de claves de un diccionario con la función *list()*:

```
list(d)
```

```
['Pepe', 'Ana', 'Luis']
```

- Al igual que en las listas, el borrado de elementos se realiza con la sentencia `del`:

```
del d['Ana']

for clave in d:
    print(clave, '->', d[clave])
```

Borra tanto el nombre 'Ana' como su número de teléfono: una clave y su valor están siempre relacionados entre sí

```
Pepe -> 640 123 456
Luis -> 640 345 678
```

Eliminar una clave del diccionario

- **del** d[clave]
- Lanza KeyError si la clave no existe

```
>>> d = {}
>>> d[1] = "uno"
>>> d[2] = "dos"
>>> print(d)
 {1: 'uno', 2: 'dos'}
>>> del d[2]
>>> print (d)
 {1: 'uno'}
>>> del d[3]
 Traceback (most recent call last):
   File "<pyshell>", line 1, in <module>
 KeyError: 3
>>>
```

Iterable and subscriptable

```
>>> list(123)
  Traceback (most recent call last):
    File "<pyshell>", line 1, in <module>
  TypeError: 'int' object is not iterable
>>> 123[0]
  Traceback (most recent call last):
    File "<pyshell>", line 1, in <module>
  TypeError: 'int' object is not subscriptable
>>>
```

```
>>> list("hello")
['h', 'e', 'l', 'l', 'o']
>>> "hello"[0]
'h'
>>>
```