DICCIONARIOS

Los diccionarios son tipos de datos que nos permiten guardar valores, a los que se puede acceder por medio de una clave. Son tipos de datos mutables y los campos no tienen asignado orden.

Definición de diccionarios. Constructor dict

```
>>> a = dict(one=1, two=2, three=3)
>>> b = {'one': 1, 'two': 2, 'three': 3}
>>> c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
>>> d = dict([('two', 2), ('one', 1), ('three', 3)])
>>> e = dict({'three': 3, 'one': 1, 'two': 2})
>>> a == b == c == d == e
True
```

Si tenemos un diccionario vacío, al ser un objeto mutable, también podemos construir el diccionario de la siguiente manera.

```
>>> dict1 = {}
>>> dict1["one"]=1
>>> dict1["two"]=2
>>> dict1["three"]=3
```

Operaciones básicas con diccionarios

```
>>> a = dict(one=1, two=2, three=3)
```

- `len()`: Devuelve número de elementos del diccionario.

```
>>> len(a)
3
```

- Indexación: Podemos obtener el valor de un campo o cambiarlo (si no existe el campo nos da una excepción `KeyError`):

```
>>> a["one"]
1
>>> a["one"]+=1
>>> a
{'three': 3, 'one': 2, 'two': 2}
```

- `del()`:Podemos eliminar un elemento, si no existe el campo nos da una excepción `KeyError`:

```
>>> del(a["one"])
>>> a
{'three': 3, 'two': 2}
```

- Operadores de pertenencia: `key in d` y `key not in d`.

```
>>> "two" in a
True
```

- `iter()`: Nos devuelve un iterador de las claves.

```
>>> next(iter(a))
'three'
```

Los diccionarios son tipos mutables

Los diccionarios, al igual que las litas, son tipos de datos mutable. Por lo tanto podemos encontrar situaciones similares a las que explicamos en su momentos con las listas.

```
>>> a = dict(one=1, two=2, three=3)
>>> a["one"]=2
>>> del(a["three"])
>>> a
{'one': 2, 'two': 2}


>>> a = dict(one=1, two=2, three=3)
>>> b = a
>>> del(a["one"])
>>> b
{'three': 3, 'two': 2}
```

En este caso para copiar diccionarios vamos a usar el método `copy()`:

```
>>> a = dict(one=1, two=2, three=3)
>>> b = a.copy()
>>> a["one"]=1000
>>> b
{'three': 3, 'one': 1, 'two': 2}
```

Métodos de eliminación: clear

```
>>> dict1 = dict(one=1, two=2, three=3)
>>> dict1.clear()
>>> dict1
{}
```

Métodos de agregado y creación: copy, dict.fromkeys, update, setdefault

```
>>> dict1 = dict(one=1, two=2, three=3)
>>> dict2 = dict1.copy()

>>> dict.fromkeys(["one","two","three"])
{'one': None, 'two': None, 'three': None}
>>> dict.fromkeys(["one","two","three"],100)
{'one': 100, 'two': 100, 'three': 100}

>>> dict1 = dict(one=1, two=2, three=3)
>>> dict2 = {'four':4,'five':5}
>>> dict1.update(dict2)
>>> dict1
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}

>>> dict1 = dict(one=1, two=2, three=3)
>>> dict1.setdefault("four",4)
4
>>> dict1
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
>>> dict1.setdefault("one",-1)
1
>>> dict1
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

setdefault: devuelve el valor de la clave si ésta existe en el diccionario y si no la inserta con el valor indicado por parámetro.

Métodos de retorno: get, pop, popitem, items, keys, values

```
>>> dict1 = dict(one=1, two=2, three=3)
>>> dict1.get("one")
1
>>> dict1.get("four")
>>> dict1.get("four","no existe")
'no existe'

>>> dict1.pop("one")
1
>>> dict1
{'two': 2, 'three': 3}
>>> dict1.pop("four")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'four'
>>> dict1.pop("four","no existe")
'no existe'

>>> dict1 = dict(one=1, two=2, three=3)
>>> dict1.popitem()
('one', 1)
>>> dict1
{'two': 2, 'three': 3}

>>> dict1 = dict(one=1, two=2, three=3)
>>> dict1.items()
dict_items([('one', 1), ('two', 2), ('three', 3)])

>>> dict1.keys()
dict_keys(['one', 'two', 'three'])
```

El tipo de datos dictviews

Los tres últimos métodos devuelven un objeto de tipo `dictviews`.

Esto devuelve una vista dinámica del ciccionario, por ejemplo:

```
>>> dict1 = dict(one=1, two=2, three=3)
>>> i = dict1.items()
>>> i
dict_items([('one', 1), ('two', 2), ('three', 3)])
>>> dict1["four"]=4
>>> i
dict_items([('one', 1), ('two', 2), ('three', 3), ('four', 4)])
```

Es este tipo de datos podemos usar las siguientes funciones:

- `len()`: Devuelve número de elementos de la vista.
- `iter()`: Nos devuelve un iterador de las claves, valores o ambas.
- `x in dictview`: Devuelve True si x está en las claves o valores.

Recorrido de diccionarios

Podemos recorrer las claves:

```
>>> for clave in dict1.keys():
...     print(clave)
one
two
three
```

Podemos recorrer los valores:

```
>>> for valor in dict1.values():
...     print(valor)
1
2
3
```

O podemos recorrer ambos:

```
>>> for clave,valor in dict1.items():
...    print(clave,"->",valor)
one -> 1
two -> 2
three -> 3
```

EJERCICIOS PROPUESTOS DE LISTAS.

1. Escribe un programa que lea una cadena y devuelva un diccionario con la cantidad de apariciones de cada palabra en la cadena. Por ejemplo, si recibe “Qué lindo día que hace hoy” debe devolver: ‘que’: 2, ‘lindo’: 1, ‘día’: 1, ‘hace’: 1, ‘hoy’: 1
2. Tenemos guardado en un diccionario los códigos morse corespondientes a cada caracter. Escribir un programa que lea una palabra y la muestre usando el código morse.

```
codigo = { 'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E':
'.', 'F': '..-.', 'G': '--.', 'H': '....', 'I': '..', 'J': '.---',
'K': '-.-', 'L': '.-..', 'M': '--', 'N': '-.', 'O': '---', 'P': '.-
--.', 'Q': '--.-', 'R': '.-.', 'S': '...', 'T': '-', 'U': '..-',
'V': '...-', 'W': '.--', 'X': '-..-', 'Y': '-.--', 'Z': '--..',
'1': '.----', '2': '..---', '3': '...--', '4': '....-', '5':
'.....', '6': '-....', '7': '--...', '8': '---..', '9': '----.',
```

```
'0': '-----', '.': '.-.-.-', ',': '--..--', ':': '---...', ';': '-
.-.-.', '?': '..--..', '!': '-.-.--', '"': '.-..-.', "'": '.----.',
'+': '.-.-.', '-': '-....-', '/': '-..-.', '=': '-...-', '_': '..--
.-', '$': '...-..-', '@': '.--.-.', '&': '.-...', '(': '-.--.',
')': '-.--.-' }
```

3. Continuar el programa: ahora nos pide un código morse donde cada letra está separada por espacios y nos da la cadena correspondiente.
4. Suponga un diccionario que contiene como clave el nombre de una persona y como valor una lista con todas sus “gustos”. Desarrolle un programa que agregue “gustos” a la persona:

- Si la persona no existe la agregue al diccionario con una lista que contiene un solo elemento.
- Si la persona existe y el gusto existe en su lista, no tiene ningún efecto.
- Si la persona existe y el gusto no existe en su lista, agrega el gusto a la lista.

Se deja de pedir personas cuando introducimos el carácter “*”.