Diccionarios

Un **diccionario** es una colección de pares formados por una clave y un valor asociado a la clave.

Se construyen poniendo los pares entre llaves separados por comas, y separando la clave del valor con dos puntos :.

Se caracterizan por:

- No tienen orden.
- Pueden contener elementos de distintos tipos.
- Son mutables, es decir, pueden alterarse durante la ejecución de un programa.
- Las claves son únicas, es decir, no pueden repetirse en un mismo diccionario, y pueden ser de cualquier tipo de datos inmutable.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
# Diccionario vacío
type({})
<class 'dict'>
# Diccionario con elementos de distintos tipos
{'nombre':'Gabriela', 'despacho': 213, 'email':'gabriela.b.arevalo@gmail.com'}
# Diccionarios anidados
{'nombre_completo':{'nombre': 'Gabriela', 'Apellidos': 'Arévalo'}}
```

---

1. Acceso a los elementos de un diccionario

- `d[clave]` devuelve el valor del diccionario `d` asociado a la clave `clave`. Si en el diccionario no existe esa clave devuelve un error.
- `d.get(clave, valor)` devuelve el valor del diccionario `d` asociado a la clave `clave`. Si en el diccionario no existe esa clave devuelve `valor`, y si no se especifica un valor por defecto devuelve `None`.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> a = {'nombre':'Gabriela', 'despacho': 213, 'email':'gabriela.b.arevalo@gmail
    .com'}
>>> a['nombre']
'Gabriela'
>>> a['despacho'] = 213
>>> a
{'nombre':'Gabriela', 'despacho': 213, 'email':'gabriela.b.arevalo@gmail.com'}
>>> a.get('email')
'gabriela.b.arevalo@gmail.com'
>>> a.get('Consultora', 'MindHub')
'MindHub'
```

---

2. Operaciones que no modifican un diccionario

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

- `min(d)`: Devuelve la mínima clave del diccionario `d` siempre que las claves sean comparables.
- `max(d)`: Devuelve la máxima clave del diccionario `d` siempre que las claves sean comparables.
- `sum(d)`: Devuelve la suma de las claves del diccionario `d`, siempre que las claves se puedan sumar.
- `clave in d`: Devuelve `True` si la clave clave pertenece al diccionario `d` y `False` en caso contrario.
- `d.keys()`: Devuelve un iterador sobre las claves de un diccionario.
- `d.values()`: Devuelve un iterador sobre los valores de un diccionario.
- `d.items()`: Devuelve un iterador sobre los pares clave-valor de un diccionario.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> a = {'nombre':'Gabriela', 'despacho': 213, 'email':'gabriela.b.arevalo@gmail
    .com'}
>>> len(a)
3
>>> min(a)
'despacho'
>>> 'email' in a
True
>>> a.keys()
dict_keys(['nombre', 'despacho', 'email'])
>>> a.values()
dict_values(['Gabriela', 213, 'gabriela.b.arevalo@gmail.com'])
>>> a.items()
dict_items([('nombre', 'Gabriela'), ('despacho', 213), ('email', 'gabriela.b.
    arevalo@gmail.com')])
```

---

3. Operaciones que modifican un diccionario

- `d[clave] = valor`: Añade al diccionario `d` el par formado por la clave `clave` y el valor `valor`.
- `d.update(d2)`: Añade los pares del diccionario `d2` al diccionario `d`.
- `d.pop(clave, alternativo)`: Devuelve del valor asociado a la clave `clave` del diccionario `d` y lo elimina del diccionario. Si la clave no está devuelve el valor alternativo.
- `d.popitem()`: Devuelve la tupla formada por la clave y el valor del último par añadido al diccionario `d` y lo elimina del diccionario.
- `del d[clave]`: Elimina del diccionario `d` el par con la clave `clave`.
- `d.clear()`: Elimina todos los pares del diccionario `d` de manera que se queda vacío.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> a = {'nombre':'Gabriela', 'despacho': 213, 'email':'gabriela.b.arevalo@gmail
    .com'}
>>> a['consultora'] = 'MindHub'
>>> a
{'nombre': 'Gabriela', 'despacho': 213, 'email': 'gabriela.b.arevalo@gmail.com',
     'universidad': 'CEU'}
>>> a.pop('despacho')
213
```

```
>>> a
{'nombre': 'Gabriela', 'email': 'gabriela.b.arevalo@gmail.com', 'consultora': '
    MindHub'}
>>> a.popitem()
('consultora', 'MindHub')
>>> a
{'nombre': 'Gabriela', 'email': 'gabriela.b.arevalo@gmail.com'}
>>> del a['email']
>>> a
{'nombre': 'Gabriela'}
>>> a.clear()
>>> a
{}
```

4. Copia de diccionarios

Existen dos formas de copiar diccionarios:

- Copia por referencia `d1 = d2`: Asocia la la variable `d1` el mismo diccionario que tiene asociado la variable `d2`, es decir, ambas variables apuntan a la misma dirección de memoria. Cualquier cambio que hagamos a través de `l1` o `l2` afectará al mismo diccionario.
- Copia por valor `d1 = list(d2)`: Crea una copia del diccionario asociado a `d2` en una dirección de memoria diferente y se la asocia a `d1`. Las variables apuntan a direcciones de memoria diferentes que contienen los mismos datos. Cualquier cambio que hagamos a través de `l1` no afectará al diccionario de `l2` y viceversa.

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> a = {1:'A', 2:'B', 3:'C'}
>>> # copia por referencia
>>> b = a
>>> b
{1:'A', 2:'B', 3:'C'}
>>> b.pop(2)
>>> b
{1:'A', 3:'C'}
>>> a
{1:'A', 3:'C'}
```

Ejercicio de Inducción: *Pruebe las siguientes líneas de código y verifique los resultados presentados:*

```
>>> a = {1:'A', 2:'B', 3:'C'}
>>> # copia por referencia
>>> b = dict(a)
>>> b
{1:'A', 2:'B', 3:'C'}
>>> b.pop(2)
>>> b
{1:'A', 3:'C'}
>>> a
{1:'A', 2:'B', 3:'C'}
```