Lección 10: Diccionarios

Índice

Anterior

Definición

Hay cuatro estructuras de datos compuestas en Python: *Listas*, *Tuplas*, *Conjuntos* y *Diccionarios*.

Los diccionarios : - Son una colección de elementos organizados por pares clave, valor. - Son mutables, se pueden añadir y quitar elementos. - Estan ordenados desde Python 3.6. - Pueden estar vacíos. - Los valores pueden ser elementos de distintos tipos, es decir cualquier tipo de objeto.

Se pueden crear vacíos con el contructor dict() o con llaves {}, o con elementos utilizando llaves {}.

Nota: En Python las llaves {} se utilizan para conjuntos y diccionarios. Unas llaves sin elementos dentro creará un diccionario y no un conjunto.

```
# Creación de diccionarios

diccionario1 = dict() # Esto es un diccionario vacio.
diccionario2 = {} # # Esto es otro diccionario vacío.

print(diccionario1)
print(diccionario2)

print(type(diccionario1))
print(type(diccionario2))

# Creación de diccionarios con valores iniciales

diccionario3 = {"clave1": "valor1", "clave2": " ", "clave3": 3}  # Los valores pueden conten

print(diccionario3)

print(type(diccionario3))

persona = {
   'nombre':'Manuel',
   'apellido':'Ejemplar',
   'edad':26,
   'país':'España',
   'casado':True,
   'conocimientos':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
```

```
    'direccion':{
        'calle':'De la protección de datos',
        'cp':'28000'
    }
    }

print(persona)
```

Se puede utilizar la función built-in *len()* para saber el número de pares clave,valor de un diccionario.

```
# Número de elementos de un diccionario

print("Diccionario:", diccionario3)
print(f"Número de pares clave,valor: {len(diccionario3)}")

print("Persona", persona)
print(f"Número de pares clave,valor: {len(persona)}")
```

Accediendo a los elementos de un diccionario

Se puede acceder a los elementos de un conjunto referenciándolos por la clave.

```
# Accediendo a los elementos de un diccionario

print(diccionario3['clave1'])
print(diccionario3['clave2'])
print(diccionario3['clave3'])

print(persona['apellido'])
```

Acceder por clave dará un error cuando la clave no existe. Para evitarlo, se puede comprobar la existencia de la clave antes de intentar acceder o utilizar el método get(), que devuelve el valor si la clave existe o *None* si la clave no existe.

> **Nota: NoneType** es un tipo de dato especial de variable en Python cuyo único valor posible es *None*. Se puede asignar a cualquier tipo de objeto para indicar que está definido pero no tiene ningún valor.

```
print(persona.get('nombre'))
print(persona.get('apellido'))
print(persona.get('hijos')) # Daría error si se intenta hacer persona['hijos']
```

Comprobando si una clave está en un diccionario

- Se puede utilizar el operador *in*.

```
# Comprobando si una clave está en un diccionario
```

```
print('apellido' in persona)
print('titulación' in persona)

# Nos podemos ayudar de un condicional (que veremos en detalle más adelante)

if 'apellido' in persona:
    print(f"El apellido de esta persona es {persona['apellido']}") # Ojo con comillas dobles

if 'titulación' in persona:
    print(f"La titulación de esta persona es {persona['titulación']}") # Ojo con comillas do
```

Añadiendo o modificando elementos de un diccionario

- Se pueden añadir pares clave,valor a un diccionario simplemente referenciando una nueva clave. Ej: *diccionario['nuevaclave']=nuevovalor*
- Se puede modificar el valor asignado a una clave, referenciándola y asignando un nuevo valor. Ej: *diccionario['clave1']=nuevovalor1*

```
## Añadiendo o modificando elementos de un diccionario

print(persona)
persona['titulación'] = 'Ingeniero Aeroespacial'
persona['conocimientos'] = ['aerodinámica', 'cálculo', 'física', 'álgebra', 'aviónica']
print(persona)
```

Eliminando pares clave,valor de un diccionario

- *pop(clave)*: Elimina el elemento cuya clave es *clave*.
- *popitem()*: Elimina el último elemento y lo devuelve.

```
# Eliminando pares clave,valor de un diccionario

print(persona)
persona.pop('casado')
print(persona)
eliminado = persona.popitem()
print(persona)
print(eliminado)
```

Obteniendo una lista de elementos de un diccionario

- El método *items()* devuelve una vista iterable con los elementos (pares clave,valor) del diccionario. Una colección de tuplas (clave,valor) que puede convertirse en una lista con *list()*, aunque no siempre es necesario convertirla.
- El método *keys()* devuelve una vista iterable con las claves del diccionario. Una colección de claves que puede convertirse en una lista con *list()*, aunque

no siempre es necesario convertirla.

- El método *values()* devuelve una vista iterable con los valores del diccionario. Una colección de valores que puede convertirse en una lista con *list()*, aunque no siempre es necesario convertirla.

```
# Obteniendo una lista de elementos de un diccionario

lista_elementos = list(persona.items())
print(persona)
print(type(persona))
print(lista_elementos)
print(type(lista_elementos))
print(lista_elementos[1])

lista_claves = list(persona.keys())
print(type(lista_claves))
print(lista_claves)

lista_valores = list(persona.values())
print(type(lista_valores))
print(lista_valores)
```

Copiando un diccionario

- *copy()*: Devuelve una copia del diccionario.

  Nota: Para copiar un diccionario hay que utilizar el método *copy()*. Si se hace una asignación a otra variable no estaremos creando un nuevo diccionario sino una referencia al diccionario, y cualquer cambio en uno de los diccionarios se verá en el otro.

```
persona = {
    'nombre':'Manuel',
    'apellido':'Ejemplar',
    'edad':26,
    'país':'España',
    'casado':True,
    'conocimientos':['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'direccion':{
        'calle':'De la protección de datos',
        'cp':'28000'
    }
    }

print(f"Esto es persona: {persona}")

persona2 = persona # Esto no crea una copia sino una nueva referencia
```

```
persona2['casado'] = False
print(f"Esto es persona: {persona}")
```

Siguiente