

# Diccionarios

Los diccionarios en Python, representados por el tipo *dict*, son colecciones desordenadas de pares **clave-valor**. Cada clave es única y se utiliza para acceder a su valor asociado. Se definen utilizando llaves { } con pares separados por dos puntos (*clave: valor*).

## Creación de diccionarios

En este fragmento de código, se muestra cómo definir y trabajar con un diccionario en Python.

### Crear el Diccionario:

- Un diccionario en Python es una colección de pares clave-valor, donde cada clave es única y está asociada a un valor específico. Los diccionarios se definen usando llaves {}.
- En el siguiente código, se crea un diccionario llamado `dicc` con cuatro pares clave-valor
  - "Clave1" es una clave en el diccionario, y 1 es el valor asociado a esa clave.
  - "Clave2" es otra clave, con 2 como su valor.
  - "Clave3" y "Clave4" siguen el mismo patrón, con 3 y 4 como valores correspondientes.
- El diccionario `dicc` está compuesto por pares clave-valor, lo que permite asociar una clave con un valor específico.
- Las claves en un diccionario deben ser únicas y pueden ser de cualquier tipo inmutable, como cadenas de texto o números.
- Los valores asociados pueden ser de cualquier tipo de datos, incluyendo listas, tuplas o incluso otros diccionarios.
- La impresión del diccionario muestra su contenido, mientras que el uso de `type()` permite confirmar que `dicc` es un objeto del tipo `dict`.

```
dicc = {"Clave1": 1,  
        "Clave2": 2,  
        "Clave3": 3,  
        "Clave4": 4}
```

```
print(f"El diccionario es: {dicc}")
print(f"El tipo del diccionario es: {type(dicc)}")
```

El diccionario es: {'Clave1': 1, 'Clave2': 2, 'Clave3': 3, 'Clave4': 4}  
El tipo del diccionario es: <class 'dict'>

En este fragmento de código, se utiliza la función `dict()` para crear un diccionario en Python.

### Crear el Diccionario:

- La función `dict()` puede utilizarse para crear diccionarios de manera alternativa a la notación de llaves `{}`.
- En el siguiente código, se usa `dict()` para definir un diccionario llamado `dicc` con cuatro pares clave-valor:

```
dicc = dict(Clave1 = 1,
            Clave2 = 2,
            Clave3 = 3,
            Clave4 = 4)
```

- Aquí, `Clave1`, `Clave2`, `Clave3`, y `Clave4` son las claves del diccionario.
- Los valores asociados a estas claves son 1, 2, 3, y 4, respectivamente.

```
dicc = dict(Clave1 = 1,
            Clave2 = 2,
            Clave3 = 3,
            Clave4 = 4)
print(f"El diccionario es: {dicc}")
print(f"El tipo del diccionario es: {type(dicc)}")
```

El diccionario es: {'Clave1': 1, 'Clave2': 2, 'Clave3': 3, 'Clave4': 4}  
El tipo del diccionario es: <class 'dict'>

### Crear el Diccionario Vacío:

- En Python, se puede crear un diccionario vacío usando llaves `{}` sin ningún contenido dentro de ellas.
- En el siguiente código, `dicc` se define como un diccionario vacío:

```
dicc = {}
```

```
# Definir un diccionario vacío
dicc = {}
print(dicc)
print(type(dicc))
```

```
{}
```

```
<class 'dict'>
```

## Crear diccionarios mediante conversión

### 1. Definir la Lista de Listas:

- En Python, puedes tener una lista que contiene otras listas, donde cada lista interna tiene dos elementos: una clave y un valor.
- En el siguiente código, `lista_de_listas` es una lista de listas que define pares clave-valor:

```
lista_de_listas = [{"clave1", 1}, {"clave2", 2}, {"clave3", 3}]
```

### 2. Convertir la Lista de Listas en un Diccionario:

- Puedes usar la función `dict()` para convertir una lista de listas en un diccionario. Cada sublista debe tener exactamente dos elementos: el primero se usa como clave y el segundo como valor.
- El siguiente código crea un diccionario `dicc` a partir de `lista_de_listas`:

```
dicc = dict(lista_de_listas)
```

Lo anterior aplica para listas de listas, listas de tuplas, tuplas de listas y tuplas de tuplas.

```
# Lista de listas a diccionario
lista_de_listas = [{"clave1",1}, {"clave2",2}, {"clave3",3}]
dicc = dict(lista_de_listas)
print(dicc)
```

```
{'clave1': 1, 'clave2': 2, 'clave3': 3}
```

```
# Lista de tuplas a diccionario
lista_de_tuplas = [("clave1",1), ("clave2",2), ("clave3",3)]
dicc = dict(lista_de_tuplas)
print(dicc)
```

```
{'clave1': 1, 'clave2': 2, 'clave3': 3}
```

```
# Tupla de tuplas a diccionario
tupla_de_tuplas = (("clave1",1),("clave2",2),("clave3",3))
dicc = dict(tupla_de_tuplas)
print(dicc)
```

```
{'clave1': 1, 'clave2': 2, 'clave3': 3}
```

```
# Tupla de listas a diccionario
tupla_de_listas = (["clave1",1],["clave2",2],["clave3",3])
dicc = dict(tupla_de_listas)
print(dicc)
```

```
{'clave1': 1, 'clave2': 2, 'clave3': 3}
```

## Obtener elementos de un diccionario

- **Acceder a los Valores Utilizando las Claves:**

- Puedes acceder a los valores del diccionario utilizando las claves correspondientes.
- El siguiente código muestra cómo imprimir los valores de las claves "nombre", "edad" y "estatura":

```
print(dicc["nombre"])
print(dicc["edad"])
print(dicc["estatura"])
```

- **Salida esperada:**

```
José Ortega
29
1.74
```

- **Explicación:**

- \* `dicc["nombre"]` accede al valor asociado con la clave "nombre", que es "José Ortega".
- \* `dicc["edad"]` accede al valor asociado con la clave "edad", que es 29.
- \* `dicc["estatura"]` accede al valor asociado con la clave "estatura", que es 1.74.

```
dicc = {"nombre": "José Ortega",  
        "edad": 29,  
        "estatura": 1.74}  
print(dicc["nombre"])  
print(dicc["edad"])  
print(dicc["estatura"])
```

```
José Ortega  
29  
1.74
```

## Obtener elementos de un diccionario mediante la clave `get()`

### Utilizar el Método `get()` para Acceder a los Valores:

- El método `get()` de un diccionario permite acceder al valor asociado con una clave específica de manera segura.
- La sintaxis básica del método `get()` es:
  - `clave`: La clave cuyo valor deseas obtener.
  - `valor_default` (opcional): El valor que se devolverá si la clave no existe en el diccionario. Si no se proporciona, el método devuelve `None` si la clave no está presente.
- En el siguiente código, se utiliza `get()` para acceder al valor asociado con la clave `"nombre"`. Si la clave no existe, se devuelve un mensaje predeterminado
- **Explicación:**
  - `dicc.get("nombre")` busca la clave `"nombre"` en el diccionario `dicc`. La clave existe, por lo que se devuelve el valor asociado, que es `"José Ortega"`.
  - Si se intentara acceder a una clave que no existe en el diccionario, el método devolvería el valor por defecto proporcionado

```
dicc = {"nombre": "José Ortega",  
        "edad": 29,  
        "estatura": 1.74}  
dicc.get("nombre", "La clave no existe en el diccionario")
```

```
'José Ortega'
```

## Obtener todos los elementos de un diccionario

### 1. Utilizar el Método `keys()` para Obtener Todas las Claves:

- El método `keys()` de un diccionario devuelve una vista de todas las claves presentes en el diccionario.
- La sintaxis básica del método `keys()` es:

```
dicc.keys()
```

- Este método no toma ningún argumento y devuelve un objeto de vista de claves (`dict_keys`), que se puede convertir a una lista si se necesita:

```
claves = list(dicc.keys())
```

- **Explicación:**

- `dicc.keys()` devuelve un objeto `dict_keys` que muestra todas las claves en el diccionario `dicc`.
- El objeto `dict_keys` es una vista dinámica de las claves del diccionario y se actualiza automáticamente si se modifican las claves del diccionario.

### 2. Convertir a Lista (Opcional):

- Si se necesita una lista de las claves en lugar de una vista, se puede convertir el objeto `dict_keys` a una lista utilizando `list()`:

```
lista_claves = list(dicc.keys())  
print(lista_claves)
```

```
# Obtener todas las claves  
dicc = {"nombre": "José Ortega",  
       "edad": 29,  
       "estatura": 1.74}  
dicc.keys()
```

```
dict_keys(['nombre', 'edad', 'estatura'])
```

### 1. Utilizar el Método `values()` para Obtener Todos los Valores:

- El método `values()` de un diccionario devuelve una vista de todos los valores presentes en el diccionario.
- La sintaxis básica del método `values()` es:

```
dicc.values()
```

- Este método no toma ningún argumento y devuelve un objeto de vista de valores (`dict_values`), que se puede convertir a una lista si se necesita:

```
valores = list(dicc.values())
```

- En el siguiente código, se utiliza `values()` para obtener todos los valores del diccionario `dicc`:

```
valores = dicc.values()
print(valores)
```

- **Explicación:**

- `dicc.values()` devuelve un objeto `dict_values` que muestra todos los valores en el diccionario `dicc`.
- El objeto `dict_values` es una vista dinámica de los valores del diccionario y se actualiza automáticamente si se modifican los valores del diccionario.

## 2. Convertir a Lista (Opcional):

- Si se necesita una lista de los valores en lugar de una vista, se puede convertir el objeto `dict_values` a una lista utilizando `list()`:

```
lista_valores = list(dicc.values())
print(lista_valores)
```

## Obtener todos los valores

`dicc = {"nombre": "José Ortega", "edad": 29, "estatura": 1.74}` `dicc.values()`

### 1. Utilizar el Método `items()` para Obtener Todos los Elementos Clave-Valor:

- El método `items()` de un diccionario devuelve una vista de todos los pares clave-valor en el diccionario.
- La sintaxis básica del método `items()` es:

```
dicc.items()
```

- Este método no toma ningún argumento y devuelve un objeto de vista de pares clave-valor (`dict_items`), que se puede convertir a una lista de tuplas si se necesita:

```
elementos = list(dicc.items())
```

- En el siguiente código, se utiliza `items()` para obtener todos los pares clave-valor del diccionario `dicc`:

```
elementos = dicc.items()
print(elementos)
```

- **Explicación:**

- `dicc.items()` devuelve un objeto `dict_items` que muestra todos los pares clave-valor en el diccionario `dicc`.
- El objeto `dict_items` es una vista dinámica de los pares clave-valor del diccionario y se actualiza automáticamente si se modifican los pares clave-valor del diccionario.

## 2. Convertir a Lista de Tuplas (Opcional):

- Si se necesita una lista de pares clave-valor en lugar de una vista, se puede convertir el objeto `dict_items` a una lista de tuplas utilizando `list()`:

```
lista_elementos = list(dicc.items())
print(lista_elementos)
```

```
# Obtener todos los elementos clave-valor
dicc = {"nombre": "José Ortega",
        "edad": 29,
        "estatura": 1.74}
dicc.items()
```

```
dict_items([('nombre', 'José Ortega'), ('edad', 29), ('estatura', 1.74)])
```

## Añadir o modificar un elemento de un diccionario

### Agregar un Nuevo Elemento:

- Puedes añadir un nuevo par clave-valor al diccionario simplemente asignando un valor a una nueva clave.
- La sintaxis para agregar un nuevo elemento es:

```
dicc[nueva_clave] = nuevo_valor
```

- En este caso, se desea agregar la clave `"carrera"` con el valor `"Ing. Biomédica"` al diccionario `dicc`:



```
dicc["carrera"] = "Ing. Biomédica"
```

```
dicc = {"nombre": "José Ortega",  
        "edad": 29,  
        "estatura": 1.74}  
dicc["carrera"] = "Ing. Biomédica"  
print(dicc)
```

```
{'nombre': 'José Ortega', 'edad': 29, 'estatura': 1.74, 'carrera': 'Ing. Biomédica'}
```

Y para modificar un elemento, simplemente se accede al valor de una clave existente y se asigna un nuevo valor.

```
dicc["estatura"] = 1.75  
print(dicc)
```

```
{'nombre': 'José Ortega', 'edad': 29, 'estatura': 1.75, 'carrera': 'Ing. Biomédica'}
```

## Crear un diccionario a partir de un diccionario vacío

### 1. Listas de Claves y Valores:

- Define dos listas: una para las claves y otra para los valores que deseas asociar en el diccionario.

```
claves = ["nombre", "edad", "estatura"]  
valores = ["José", 29, 1.74]
```

### 2. Llenar el Diccionario Usando un Bucle:

- Utiliza la función `zip` para combinar las listas de claves y valores, y luego itera sobre estas combinaciones para agregar los pares clave-valor al diccionario.
- La sintaxis del bucle `for` es:

```
for clave, valor in zip(claves, valores):  
    dicc[clave] = valor
```

- Esto asigna cada `valor` a su correspondiente `clave` en el diccionario `dicc`.

```

dicc = {}

# Lista de claves y valores
claves = ["nombre", "edad", "estatura"]
valores = ["José", 29, 1.74]

# Llenar el diccionario
for clave, valor in zip(claves, valores):
    dicc[clave] = valor
print(dicc)

```

```
{'nombre': 'José', 'edad': 29, 'estatura': 1.74}
```

## Combinar diccionarios

### Combinar los Diccionarios en un nuevo diccionario, sin modificar los originales:

- Para combinar los dos diccionarios en uno nuevo sin modificar los diccionarios originales, utiliza la sintaxis de descompactación (**\*\***) de diccionarios.
- La sintaxis para combinar los diccionarios es:

```
nuevo_dicc = {**dicc1, **dicc2}
```

- Aquí, **\*\*dicc1** y **\*\*dicc2** descomponen los diccionarios en pares clave-valor, que luego se combinan en un nuevo diccionario **nuevo\_dicc**.

```

# Combinar sin modificar los diccionario originales
dicc1 = {"nombre": "José Ortega",
        "edad": 29,
        "estatura": 1.74}
dicc2 = {"peso": 93,
        "carrera": "Ing. biomédica"}
nuevo_dicc = {**dicc1, **dicc2}
print(nuevo_dicc)

```

```
{'nombre': 'José Ortega', 'edad': 29, 'estatura': 1.74, 'peso': 93, 'carrera': 'Ing. biomédica'}
```

### Combinar Diccionarios Modificando un Original:

- Para combinar los dos diccionarios y modificar uno de ellos, usa el método **update()**.
- El método **update()** agrega los pares clave-valor del segundo diccionario al primero.

```
dicc1.update(dicc2)
```

- Este método modifica `dicc1` al agregar las claves y valores de `dicc2`. Si hay claves duplicadas, los valores del segundo diccionario (`dicc2`) sobrescriben los valores del primero (`dicc1`).

```
# Combinar modificando los diccionario originales
dicc1 = {"nombre": "José Ortega",
        "edad": 29,
        "estatura": 1.74}
dicc2 = {"peso": 93,
        "carrera": "Ing. biomédica"}
dicc1.update(dicc2)
print(dicc1)
```

```
{'nombre': 'José Ortega', 'edad': 29, 'estatura': 1.74, 'peso': 93, 'carrera': 'Ing. biomédica'}
```

## Borrar elementos de un diccionario

### Eliminar un Elemento por su Clave:

- Para eliminar un elemento de un diccionario por su clave, usa la palabra clave `del` seguida de la clave del elemento que deseas eliminar entre corchetes.

```
del(dicc["edad"])
```

- Esto eliminará la clave "edad" y su valor asociado del diccionario.

```
# Borrar elementos por su clave
dicc = {"nombre": "José",
        "edad": 29,
        "estatura": 1.74}
del(dicc["edad"])
print(dicc)
```

```
{'nombre': 'José', 'estatura': 1.74}
```

### Eliminar un Elemento Usando `pop()`:

- El método `pop()` se utiliza para eliminar un elemento de un diccionario, especificando su clave. Este método no solo elimina el elemento, sino que también devuelve el valor asociado a la clave eliminada.

```
elemento_eliminado = dicc.pop("edad")
```

- Aquí, el elemento con la clave `"edad"` es eliminado del diccionario y su valor (29) se almacena en la variable `elemento_eliminado`.

```
# Borrar elementos por su clave, usando pop()
dicc = {"nombre": "José",
        "edad": 29,
        "estatura": 1.74}
elemento_eliminado = dicc.pop("edad")
print(f"Diccionario: {dicc}")
print(f"Elemento eliminado: {elemento_eliminado}")
```

```
Diccionario: {'nombre': 'José', 'estatura': 1.74}
Elemento eliminado: 29
```

### Borrar Todos los Elementos Usando `clear()`:

- El método `clear()` se utiliza para eliminar todos los elementos de un diccionario, dejándolo vacío.

```
dicc.clear()
```

```
# Borrado completo del diccionario
dicc = {"nombre": "José",
        "edad": 29,
        "estatura": 1.74}
dicc.clear()
print(dicc)
```

```
{}
```

### Reasignar el Diccionario a un Diccionario Vacío:

- Para borrar todos los elementos del diccionario, puedes simplemente reasignarlo a un diccionario vacío `{}`.

```
dicc = {}
```

```
# Borrado completo del diccionario
dicc = {"nombre": "José",
        "edad": 29,
        "estatura": 1.74}
dicc = {}
print(dicc)
```

```
{}
```

### Pertenencia de un elemento en un diccionario

```
dicc = {"nombre": "José",
        "edad": 29,
        "estatura": 1.74}
"nombre" in dicc
```

True

```
dicc = {"nombre": "José",
        "edad": 29,
        "estatura": 1.74}
"nombre" not in dicc
```

False

### Longitud de un diccionario

```
dicc = {"nombre": "José",
        "edad": 29,
        "estatura": 1.74}
longitud = len(dicc)
print(longitud)
```

3

## Iterar sobre un diccionario

### Iterar sobre las Claves:

- Usa un bucle for para iterar sobre las claves del diccionario.
- El método `.keys()` devuelve una vista de las claves del diccionario.

```
for clave in dicc.keys():  
    print(clave)
```

```
# Iterar sobre las claves  
dicc = {"nombre": "José Ortega",  
        "edad": 29,  
        "estatura": 1.74}  
for clave in dicc.keys():  
    print(clave)
```

```
nombre  
edad  
estatura
```

### Iterar sobre los Valores:

- Usa un bucle for para iterar sobre los valores del diccionario.
- El método `.values()` devuelve una vista de los valores del diccionario.

```
for valor in dicc.values():  
    print(valor)
```

```
# Iterar sobre los valores  
dicc = {"nombre": "José Ortega",  
        "edad": 29,  
        "estatura": 1.74}  
for valor in dicc.values():  
    print(valor)
```

```
José Ortega  
29  
1.74
```

### Iterar sobre Claves y Valores:

- Usa un bucle `for` para iterar simultáneamente sobre las claves y los valores del diccionario.
- El método `.items()` devuelve una vista de pares clave-valor.

```
for clave, valor in dicc.items():
    print(f"{clave}: {valor}")

::: {.cell execution_count=170}
``` {.python .cell-code}
# Iterar sobre clave-valor
dicc = {"nombre": "José Ortega",
        "edad": 29,
        "estatura": 1.74}
for clave, valor in dicc.items():
    print(f"{clave}:{valor}")
```

```
nombre:José Ortega
edad:29
estatura:1.74
```

```
:::
```

## Copia de un diccionario

- Se crea una copia del diccionario original utilizando el método `.copy()`.
- La copia se almacena en `nuevo_dicc`.

```
nuevo_dicc = dicc.copy()
```

```
dicc = {"nombre": "José Ortega",
        "edad": 29,
        "estatura": 1.74}
nuevo_dicc = dicc.copy()
dicc["estatura"] = 1.75
print(dicc)
print(nuevo_dicc)
```

```
{'nombre': 'José Ortega', 'edad': 29, 'estatura': 1.75}
{'nombre': 'José Ortega', 'edad': 29, 'estatura': 1.74}
```

```
dicc is nuevo_dicc
```

False