Módulo 5.1: Diccionarios

Prof. Carlos Cedeño

Imagina una agenda telefónica o un diccionario real: cada nombre tiene un número, cada palabra tiene una definición. Los diccionarios en Python funcionan de manera similar.

1. ¿Qué es un Diccionario?

- Son colecciones de elementos desordenadas.
- Almacenan datos en pares **clave-valor**.
- Cada clave es **única** y se utiliza para acceder a su valor correspondiente.
- Son **mutables**, lo que significa que puedes añadir, modificar o eliminar elementos después de su creación.

1.1. Sintaxis Básica

Los diccionarios se definen usando llaves `{}` y pares `clave: valor` separados por comas.

```
mi_diccionario = {
    "clave1": "valor1",
    "clave2": "valor2",
    "clave3": "valor3"
}
```

- **Clave:** Debe ser un tipo de dato inmutable (strings, números).
- **Valor:** Puede ser cualquier tipo de dato (strings, números, listas, otros diccionarios, etc.).

2. Creación de Diccionarios

2.1. Diccionarios Vacíos

Puedes crear un diccionario vacío y luego añadir elementos.

```
diccionario_vacio = {}
# O también
diccionario_vacio = dict()
```

2.2. Diccionarios con Datos Iniciales

```
persona = {
    "nombre": "Ana",
    "edad": 30,
    "ciudad": "Madrid"
}

productos = {
    101: "Laptop",
    102: "Mouse",
    103: "Teclado"
}
```

3. Acceso a Elementos

Se accede a los valores usando sus claves entre corchetes `[]`.

```
persona = {"nombre": "Ana", "edad": 30}
print(persona["nombre"])  # Salida: Ana
print(persona["edad"])   # Salida: 30
```

- **¡Importante!** Si intentas acceder a una clave que no existe, Python lanzará un `KeyError`.

3.1. Usando `.get()`

Para evitar `KeyError`, puedes usar el método `.get()`. Si la clave no existe, `.get()` devuelve `None` o un valor predeterminado que especifiques.

```
persona = {"nombre": "Ana", "edad": 30}
print(persona.get("nombre"))   # Salida: Ana
print(persona.get("telefono")) # Salida: None
print(persona.get("pais", "Desconocido")) # Salida: Desconocido
```

4. Modificar y Añadir Elementos

Los diccionarios son mutables.

4.1. Añadir un Nuevo Par Clave-Valor

Simplemente asigna un valor a una nueva clave.

```
persona = {"nombre": "Ana"}
persona["edad"] = 30
print(persona) # Salida: {'nombre': 'Ana', 'edad': 30}
```

4.2. Modificar un Valor Existente

Asigna un nuevo valor a una clave existente.

```
persona = {"nombre": "Ana", "edad": 30}
persona["edad"] = 31
print(persona) # Salida: {'nombre': 'Ana', 'edad': 31}
```

5. Eliminar Elementos

5.1. Usando `del`

La palabra clave `del` elimina un par clave-valor.

```
persona = {"nombre": "Ana", "edad": 30}
del persona["edad"]
print(persona) # Salida: {'nombre': 'Ana'}
```

5.2. Usando `.pop()`

El método `.pop()` elimina un elemento por su clave y devuelve su valor.

```
persona = {"nombre": "Ana", "edad": 30}
edad_eliminada = persona.pop("edad")
print(edad_eliminada) # Salida: 30
print(persona)        # Salida: {'nombre': 'Ana'}
```

5.3. Usando `.clear()`

Elimina todos los elementos del diccionario, dejándolo vacío.

```
diccionario = {"a": 1, "b": 2}
diccionario.clear()
print(diccionario) # Salida: {}
```

6. Métodos Útiles de Diccionarios

- `.keys()` : Devuelve una vista de todas las claves del diccionario.
- `.values()` : Devuelve una vista de todos los valores del diccionario.
- `.items()` : Devuelve una vista de todos los pares clave-valor (como tuplas).

```
estudiante = {"nombre": "Luis", "carrera": "Sistemas", "id": 123}

print(estudiante.keys())   # Salida: dict_keys(['nombre', 'carrera', 'id'])
print(estudiante.values()) # Salida: dict_values(['Luis', 'Sistemas', 123])
print(estudiante.items())  # Salida: dict_items([('nombre', 'Luis'), ('carrera', 'Sistemas'), ('id', 123)])
```