## 5. Estructuras de datos

### a) Diccionarios

Los diccionarios son una estructura de datos en Python que nos permiten almacenar pares de key-value. Es decir, para cada valor en un diccionario, tenemos una clave que lo identifica. Las claves son únicas e inmutables, mientras que los valores pueden ser de cualquier tipo. res clave-valor. Además, los diccionarios tienen varios métodos útiles, como keys(), values() y items().

### b) Creación e inicialización de diccionarios

Podemos crear un diccionario vacío usando las llaves {} o la función dict(). También podemos inicializar un diccionario con valores clave-valor.

#### Crear un diccionario vacío

```
my_dict = {}
print(my_dict)
```

```
> {}
```

#### Crear un diccionario con valores clave-valor

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
print(my_dict)
```

```
> {'a': 1, 'b': 2, 'c': 3}
```

#### Crear un diccionario con la función dict()

```
my_dict = dict(a=1, b=2, c=3)
print(my_dict)
```

```
> {'a': 1, 'b': 2, 'c': 3}
```

### c) Operaciones de diccionarios

#### Adición de pares key-value

Podemos agregar un nuevo par key-value a un diccionario usando la sintaxis

```
dict[key] = value
```

Por ejemplo:

```
my_dict = {'a': 1, 'b': 2}
my_dict['c'] = 3
print(my_dict)
```

```
> {'a': 1, 'b': 2, 'c': 3}
```

#### Actualización de pares key-value

Podemos actualizar el valor de un par key-value existente usando la sintaxis

```
dict[key] = new_value
```

Por ejemplo:

```
my_dict['b'] = 4
print(my_dict)
```

```
> {'a': 1, 'b': 4, 'c': 3}
```

#### Eliminación de pares key-value

Podemos eliminar un par key-value de un diccionario usando la sintaxis

```
del dict[key]
```

Por ejemplo

```
del my_dict['a']
print(my_dict)
```

```
> {'b': 4, 'c': 3}
```

### d) Métodos de diccionarios

#### keys()

Devuelve una lista de todas las claves en el diccionario.

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
print(my_dict.keys())
```

```
> dict_keys(['a', 'b', 'c'])
```

#### values()

Devuelve una lista de todos los valores en el diccionario.

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
print(my_dict.values())
```

```
> dict_values([1, 2, 3])
```

#### items()

Devuelve una lista de tuplas (key, value) de todos los pares key-value en el diccionario.

```
print(my_dict.items())
```

```
> dict_items([('a', 1), ('b', 2), ('c', 3)])
```

En conclusión, los diccionarios en Python son estructuras de datos muy útiles y flexibles que permiten almacenar datos en pares key-value. Se pueden crear e inicializar fácilmente utilizando llaves y los valores pueden ser de cualquier tipo de datos, incluso otros diccionarios. Los métodos de diccionarios como keys(), values() y items() son muy útiles para acceder a diferentes partes de los diccionarios. Además, se pueden realizar varias operaciones en los diccionarios, como agregar, actualizar o eliminar elementos. En general, los diccionarios son una herramienta esencial para cualquier programador de Python que necesite almacenar y manipular datos de manera eficiente y organizada.