6) Dictionaries

- **Introduction to dictionaries: key-value pairs.**

  What Are Key-Value Pairs?

  A key-value pair is a set where:

  - A dictionary (*dict*) is a mutable, ordered (since Python 3.7), and indexed collection that maps unique, immutable keys to values of any data type
  - Keys must be immutable (e.g., strings, numbers, tuples) and unique—duplicates overwrite existing entries .
  - The value is the data or information associated with that key.

Example :

```
my_dict = {

   "name": "UG",

   "age": 20,

   "city": "Gir"

}
```

"name", "age", and "city" are **keys**

"UG", 20, and "Gir" are **values**

Key Characteristics

1. **Uniqueness of Keys:**

   --Keys must be unique within a dictionary. Duplicate keys are not allowed.

2. **Key Types:**

   -- Keys must be of an immutable type (e.g., strings, numbers, tuples). Mutable types like lists are not allowed as keys.

3. **Value Types:**

   -- Values can be of any type—numbers, strings, lists, other dictionaries, etc.

4. **Unordered (prior to Python 3.7):**

   -- In older versions, dictionaries did not maintain order. From Python 3.7+, the insertion order is preserved.

- **Accessing, adding, updating, and deleting dictionary elements.**

    - **Accessing Dictionary Elements**

  You can access the value associated with a key using square brackets [] or the .get() method

```
person = {"name": "UG", "age": 20}

print(person["name"]) # Output: UG
```

    - **Adding Elements**

  You can add a new key-value pair simply by assigning a value to a new key.

```
d = {}

d["email"] = "ug@example.com"

print(d)
```

    - **Updating Elements**

-- If the key already exists, assigning a new value to it will update the old value.

```
d.update({"email": "ug@example.com"})
```

- **Deleting Element**

  -- There are several ways to delete elements from a dictionary

```
# Dictionary with data
d = {"email": "ug@example.com", "name": "Umar"}

# Deleting the 'email' key
del d["email"]

# Printing the updated dictionary
print(d)
```

- **Dictionary methods like keys(), values(), and items().**

  - **keys() Method**

    --The keys() method returns a view object that contains all the keys in the dictionary.

```
person = {"name": "ug", "age": 30, "city": "gir"}

print(person.keys())

# Output: dict_keys(['name', 'age', 'city'])
```

  - **values() Method**

    --The values() method returns a view object containing all the values in the dictionary.

```
person = {"name": "ug", "age": 30, "city": "gir"}

print(person.values())

# Output: dict_values(['ug', 30, 'gir'])
```

- **items() Method**

  -- The items() method returns a view object with all key-value pairs as tuples.

```
person = {"name": "ug", "age": 30, "city": "gir"}

print(person.items())

# Output: dict_items([('name', 'ug'), ('age', 30), ('city', 'gir')])
```