**ASSIGNMENT6**

**Topic name:** Dictionaries

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

**Introduction to Dictionaries: Key-Value Pairs**

A **dictionary** in programming (commonly found in Python and other similar languages) is a data structure that stores data in the form of **key-value pairs**. This structure allows you to efficiently map unique keys to corresponding values, making it ideal for scenarios where fast lookups are needed.

**Key-Value Pair Concept**

* **Key**: A unique identifier used to access the corresponding value.
* **Value**: The data or information associated with a specific key.

In Python, dictionaries are enclosed in curly braces {} and keys are separated from values using a colon :.

**Example**

# Example of a dictionary

person = { "name": "Alice","age": 25., "city": "New York"}

# Accessing values using keys

print(person["name"]) # Output: Alice

print(person["age"]) # Output: 25

**Key Properties of Dictionaries**

1. **Unordered**: Before Python 3.7, dictionaries were unordered, but from Python 3.7 onwards, they preserve the insertion order of keys.
2. **Unique Keys**: Keys in a dictionary must be unique. If a duplicate key is added, the latest value will overwrite the previous one.
3. **Mutable**: You can change, add, or remove key-value pairs from a dictionary.

**Common Dictionary Operations**

1. **Create a Dictionary**

my\_dict = {"key1": "value1", "key2": "value2"}

1. **Access a Value**

**Example:**

print(my\_dict["key1"]) # Output: value1

1. **Add or Update a Key-Value Pair**

**Example:**

my\_dict["key3"] = "value3"

print(my\_dict) # Output: {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}

1. **Remove a Key-Value Pair**

**Example:**

del my\_dict["key2"]

print(my\_dict) # Output: {'key1': 'value1', 'key3': 'value3'}

1. **Iterate Through a Dictionary**

**Example:**

for key, value in my\_dict.items():

print(f"{key}: {value}")

1. **Check if a Key Exists**

**Example:**

if "key1" in my\_dict:

print("Key exists")

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

**1. Accessing Elements**

To access the value associated with a key in a dictionary, use the key inside square brackets or the get() method.

**Example:**

my\_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}

# Access using square brackets

print(my\_dict['name']) # Output: Alice

# Access using the get() method

print(my\_dict.get('age')) # Output: 25

# Handling non-existent keys with get()

print(my\_dict.get('country', 'Not Found')) # Output: Not Found

**2. Adding Elements**

To add a new key-value pair, use the assignment operator with the new key.

**Example:**

my\_dict = {'name': 'Alice', 'age': 25}

# Add a new key-value pair

my\_dict['city'] = 'New York'

print(my\_dict) # Output: {'name': 'Alice', 'age': 25, 'city': 'New York'}

**3. Updating Elements**

To update an existing key-value pair, use the assignment operator with the key.

**Example:**

my\_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}

# Update an existing value

my\_dict['age'] = 30

print(my\_dict) # Output: {'name': 'Alice', 'age': 30, 'city': 'New York'}

**4. Deleting Elements**

You can delete elements using the del statement or the pop() method.

**Example:**

my\_dict = {'name': 'Alice', 'age': 30, 'city': 'New York'}

# Delete using del

del my\_dict['city']

print(my\_dict) # Output: {'name': 'Alice', 'age': 30}

# Delete using pop

age = my\_dict.pop('age')

print(my\_dict) # Output: {'name': 'Alice'}

print(age) # Output: 30

**Example:**

my\_dict = {'name': 'Alice', 'age': 30}

# Clear the dictionary

my\_dict.clear()

print(my\_dict) # Output: {}

# Pop the last item

my\_dict = {'name': 'Alice', 'age': 30}

last\_item = my\_dict.popitem()

print(last\_item) # Output: ('age', 30)

print(my\_dict) # Output: {'name': 'Alice'}

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

1**. keys() Method**

* **Description:** Returns a view object that displays a list of all the keys in the dictionary.
* **Syntax:** dictionary.keys()
* **Example:**

my\_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}

print(my\_dict.keys()) # Output: dict\_keys(['name', 'age', 'city'])

* Use Case: Iterate over the keys of a dictionary.

for key in my\_dict.keys():

print(key)

2. values() Method

* Description: Returns a view object that displays a list of all the values in the dictionary.
* Syntax: dictionary.values()
* Example:

my\_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}

print(my\_dict.values()) # Output: dict\_values(['Alice', 25, 'New York'])

* Use Case: Iterate over the values of a dictionary.

for value in my\_dict.values():

print(value)

3. items() Method

* Description: Returns a view object that displays a list of tuples, where each tuple contains a key-value pair.
* Syntax: dictionary.items()
* Example:

my\_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}

print(my\_dict.items()) # Output: dict\_items([('name', 'Alice'), ('age', 25), ('city', 'New York')])

* Use Case: Iterate over both keys and values in a dictionary.

Example:

for key, value in my\_dict.items():

print(f"{key}: {value}")