Day 3: Dictionaries

DICTIONARIES

Definition: mutable collection of key-value pairs

Syntax:

Curly Braces **{ }**

Colon **:** to separate keys and values

Example: `my_dict = {'name': 'John', 'age': 25, 'city': 'New York'}`

Key Operations:

REMEMBER: Keys must be unique

ACCESSING VALUES:

Get the value associated with a key in my_dict:

```
my_dict = {'name': 'John', 'age': 25, 'city': 'New York'}
my_dict['name']
```

Result: 'John'

ADDING OR UPDATING VALUES:

Add a new key-value pair or update an existing key's value:

```
my_dict = {'name': 'John', 'age': 25}
my_dict['city'] = 'New York'  # Adds new key-value pair
my_dict['age'] = 26  # Updates the value of 'age'
```

Result: {'name': 'John', 'age': 26, 'city': 'New York'}

REMOVING VALUES:

Remove a key-value pair using .pop():

```
my_dict = {'name': 'John', 'age': 25, 'city': 'New York'}
my_dict.pop('age')
```

Result: {'name': 'John', 'city': 'New York'}

USEFUL METHODS:

Get all keys: `my_dict.keys()`
Result: `dict_keys(['name', 'age', 'city'])`

Get all values: `my_dict.values()`
Result: `dict_values(['John', 25, 'New York'])`

Get all key-value pairs: `my_dict.items()`
Result: `dict_items([('name', 'John'), ('age', 25), ('city', 'New York')])`

NOTE:

In Python, `.keys()`, `.values()`, and `.items()` return **view objects** (e.g., dict_keys([...]), dict_values([...])), which provide a real-time, read-only view of the dictionary. They update automatically if the dictionary changes. To work with them like lists, you can convert them using `list()`.

Example:

```
my_dict = {'name': 'John', 'age': 25, 'city': 'New York'}
values_as_list = list(my_dict.values())
```

Result: ['John', 25, 'New York']

Use Case

DICTIONARIES are useful for storing data that can be accessed by a specific label or identifier, like a contact list where each person's name is a key and their details are values.

PRACTICE EXERCISES

CREATING AND ACCESSING VALUES:

Create a dictionary with details about a book, including keys for **title**, **author**, and **year**. Then, access and print the value for "**author**".

Example Dictionary:

```
book = {'title': '1984', 'author': 'George Orwell', 'year': 1949}
```

Expected Output: **'George Orwell'**

ADDING AND UPDATING ENTRIES:

Start with an empty dictionary for a person's profile, `profile = {}`.

Add keys and values for **'name'**, **'age'**, and **'city'**, then update the **'city'** value.

Expected Result: `{'name': 'Alice', 'age': 30, 'city': 'Paris'}`

After update: `{'name': 'Alice', 'age': 30, 'city': 'London'}`

REMOVING AND RETRIEVING KEYS AND VALUES:

Given the dictionary:

```
student = {'name': 'Emma', 'grade': 'A', 'subject': 'Math'}
```

Remove the **'subject'** key using `.pop()`, and then retrieve the remaining keys and values separately.

Expected Output: `{'name': 'Emma', 'grade': 'A'}`

Keys: `dict_keys(['name', 'grade'])`

Values: `dict_values(['Emma', 'A'])`