**ASSIGNMENT7**

**Topic name:** Working with Dictionaries

**Iterating over a dictionary using loops.**

**Iterating Over a Dictionary in Python: Theory**

In Python, dictionaries are collections of key-value pairs. When working with dictionaries, it is common to iterate over their contents. This can be done in several ways, depending on whether you want to access keys, values, or both.

**1. Iterating Over Keys**

* When you iterate directly over a dictionary (for key in my\_dict), it iterates over the keys.
* You can explicitly use the keys() method to iterate over just the keys.
* Useful when you only need the keys.

**Example**:

for key in my\_dict.keys():

print(key)

**2. Iterating Over Values**

* Use the values() method to iterate over the values of the dictionary.
* Useful when you only need the values without concern for the keys.

**Example**:

for value in my\_dict.values():

print(value)

**3. Iterating Over Key-Value Pairs**

* **Method**: Use the items() method to iterate over both keys and values. It returns key-value pairs as tuples.
* **Use Case**: Useful when you need both keys and values for processing.

**Example**:

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

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

**MERGING TWO LISTS**

Merging two lists into a dictionary can be done in multiple ways. Here’s a breakdown using **loops** and the **zip() function**.

**Method 1: Using a Loop**

You can manually iterate through two lists to create a dictionary.

# Two lists to merge

keys = ['name', 'age', 'city']

values = ['Alice', 25, 'New York']

# Merging using a loop

merged\_dict = {}

for i in range(len(keys)):

merged\_dict[keys[i]] = values[i]

# Output the resulting dictionary

print("Merged dictionary using a loop:", merged\_dict)

**Output**:

Merged dictionary using a loop: {'name': 'Alice', 'age': 25, 'city': 'New York'}

**Method 2: Using zip()**

The zip() function pairs elements from two lists together, making it straightforward to create a dictionary.

# Two lists to merge

keys = ['name', 'age', 'city']

values = ['Alice', 25, 'New York']

# Merging using zip()

merged\_dict = dict(zip(keys, values))

# Output the resulting dictionary

print("Merged dictionary using zip():", merged\_dict)

**Output**:

Merged dictionary using zip(): {'name': 'Alice', 'age': 25, 'city': 'New York'}

**COUNT THE OCCURRENCES**

You can use a dictionary to count the occurrences of characters in a string by iterating through each character and updating the dictionary accordingly. Here's how you can do it:

### ****Program to Count Occurrences of Characters in a String Using a Dictionary****

# Input string

input\_string = "hello world"

# Create an empty dictionary to store character counts

char\_count = {}

# Iterate over each character in the string

for char in input\_string:

# If the character is already in the dictionary, increment its count

if char in char\_count:

char\_count[char] += 1

# Otherwise, add the character to the dictionary with a count of 1

else:

char\_count[char] = 1

# Output the character counts

print("Character counts:", char\_count)

### ****Output****

Character counts: {'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1, 'r': 1, 'd': 1}