EXPERIMENT NO. - 11

AIM: Write a function called how many, which returns the sum of the number of values associated with a dictionary.

THEORY:

Python dictionary is an ordered collection (starting from **Python 3.7**) of items. It stores elements in **key/value** pairs. Here, **keys** are unique identifiers that are associated with each **value**.

Let's see an example,

If we want to store information about countries and their capitals, we can create a dictionary with country names as **keys** and capitals as **values**.

Keys	Values
Nepal	Kathmandu
Italy	Rome
England	London

Create a dictionary in Python

Here's how we can create a dictionary in Python.

```
capital_city = {"Nepal": "Kathmandu", "Italy": "Rome", "England": "London"}
print(capital_city)
Run Code
```

Output

```
{'Nepal': 'Kathmandu', 'Italy': 'Rome', 'England': 'London'}
```

In the above example, we have created a dictionary named capital_city. Here,

1. **Keys** are "Nepal", "Italy", "England"

2. **Values** are "Kathmandu", "Rome", "London"

Note: Here, **keys** and **values** both are of string type. We can also have **keys** and **values** of different data types.

Example 1: Python Dictionary

```
# dictionary with keys and values of different data types
numbers = {1: "One", 2: "Two", 3: "Three"}
print(numbers)
Run Code
```

Output

```
[3: "Three", 1: "One", 2: "Two"]
```

In the above example, we have created a dictionary named numbers. Here, **keys** are of integer type and **values** are of string type.

Add Elements to a Python Dictionary

We can add elements to a dictionary using the name of the dictionary with []. For example,

```
capital_city = {"Nepal": "Kathmandu", "England": "London"}
print("Initial Dictionary: ",capital_city)

capital_city["Japan"] = "Tokyo"

print("Updated Dictionary: ",capital_city)
Run Code
```

Output

```
Initial Dictionary:  {'Nepal': 'Kathmandu', 'England': 'London'}
Updated Dictionary:  {'Nepal': 'Kathmandu', 'England': 'London', 'Japan': 'Tokyo'}
```

In the above example, we have created a dictionary named capital_city. Notice the line,

```
capital_city["Japan"] = "Tokyo"
```

Here, we have added a new element to capital_city with **key**: Japan and **value**: Tokyo.

Change Value of Dictionary

We can also use [] to change the value associated with a particular key. For example,

```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}
print("Initial Dictionary: ", student_id)

student_id[112] = "Stan"

print("Updated Dictionary: ", student_id)
```

Run Code

Output

```
Initial Dictionary:  {111: 'Eric', 112: 'Kyle', 113: 'Butters'}
Updated Dictionary:  {111: 'Eric', 112: 'Stan', 113: 'Butters'}
```

In the above example, we have created a dictionary named student_id. Initially, the value associated with the key 112 is "Kyle". Now, notice the line,

```
student_id[112] = "Stan"
```

Here, we have changed the value associated with the key 112 to "Stan".

Accessing Elements from Dictionary

In Python, we use the keys to access their corresponding values. For example,

```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print(student_id[111])  # prints Eric
print(student_id[113])  # prints Butters
```

Run Code

Here, we have used the keys to access their corresponding values.

If we try to access the value of a key that doesn't exist, we'll get an error. For example,

```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}
print(student_id[211])

# Output: KeyError: 211
```

Run Code

Removing elements from Dictionary

We use the del statement to remove an element from the dictionary. For example,

```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print("Initial Dictionary: ", student_id)

del student_id[111]

print("Updated Dictionary ", student_id)
```
Run Code

Output

```
Initial Dictionary:  {111: 'Eric', 112: 'Kyle', 113: 'Butters'}
Updated Dictionary  {112: 'Kyle', 113: 'Butters'}
```

Here, we have created a dictionary named student_id. Notice the code,

```
del student_id[111]
```

The del statement removes the element associated with the key 111.

We can also delete the whole dictionary using the del statement,

```
student_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

# delete student_id dictionary
del student_id

print(student_id)

# Output: NameError: name 'student_id' is not defined
```
Run Code

We are getting an error message because we have deleted the student_id dictionary and student_id doesn't exist anymore.

Iterating Through a Dictionary

We can iterate through each key in a dictionary using a for loop.

```
# Iterating through a Dictionary
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
for i in squares:
    print(squares[i])
Run Code
```

Output

```
1
9
25
49
81
```

Here, we have iterated through each **key** in the `squares` dictionary using the `for`

<u>EXERCISE:</u>

Write a function called how many, which returns the sum of the number of values associated with a dictionary.

T= animals = {‘L’:[‘Lion’],’D’:[‘Donkey’],’E’:[‘Elephant’]}

```
>>>print(how_many(animals))
```

Conclusion: Hence, we have successfully studied about program for writing a function called how many, which returns the sum of the number of values associated with a dictionary.