Practical No. 5

Date:

Aim: Implementing the concept of sets and the Dictionaries in python.

Practical No. 5

Aim: Implementing the concept of sets & the Dictionaries in python.

Theory:

Set :

A set is an unordered collection of unique elements in python. Unlike list or tuples, sets do not allow duplicate values. Sets are also mutable, meaning user can add or remove items after the set's creation.

Key Characteristics of Set :

i) Unordered : The items in a set are not stored in a specific order, so you cannot access elements by index.

ii) Unique items : A set does not allow duplicates.

iii) Mutable : User can add or remove items, but the items themselves must be immutable (eg: numbers, strings or tuples).

iv) Optimized : For membership testing making them ideal for checking if an item exists in a collection.

Common Set Operations :

i) Creating a set.

ii) Adding elements.

iii) Removing elements.

iv) Set operations like :

*) Union, Intersection, Difference & Symmetric Difference.

Codes:

1) Creating a set:

```
my_set = {1,2,3,4}
empty_set = set()  #creating an empty set.
```

2) Adding elements:

```
my_set = {1,2,3,4}
my_set.add(6)  # Add "6" to the set.
```

3) Removing elements:

```
my_set.remove(2) # Removes "2", raises error if
                 the element is not present
my_set.discard(7) # Removes "7", but No error if
                  element is not present.
```

4) Set Operations:

i) Union (combine elements from both sets):

```
set_1 = {1,2,3}
set_2 = {3,4,5}
union_set = set_1.union(set_2)
```

ii) Intersection (common elements from both sets):

```
set_1 = {1,2,3}
set_2 = {3,4,5}
intersection_set = set1.intersection(set_2)
```

iii) Difference : (Elements in set1 but not in set2):

```
set_1 = {1, 2, 3}
set_2 = {3, 4, 5}
difference_set = set_1.difference(set_2)
```

iv) Symmetric Difference (elements in either set_1 or set_2 but not both):

```
set_1 = {1, 2, 3}
set_2 = {3, 4, 5}
sym_diff_set = set1.symmetric_difference(set_2)
```

Theory: Dictionaries :

A dictionaries in python is an unordered collection of key-value pairs. Each key is associated with a value & both keys & values can be of any data type. However each key must be unique & immutable (eg: strings, numbers or tuples).

Key characteristics of Dictionary:

i) Key value Pairs : Each item in a dictionary is stored as a pair of key and it's corr. value.

ii) Unordered : Like sets, the dictionaries do not maintain any specific order.

iii) Mutable : User can add, update or delete key-value pair after creating the dictionary.

Common Dictionary Operations :

i) Creating a Dictionary
ii) Accessing Values
iii) Adding / Updating Values
iv) Removing Elements
v) Iterating through a Dictionary

Codes :

1) Creating a Dictionary :

```
my_dictionary = {'name': 'John', 'age': 25,
                 'city': 'New York'}
empty_dict = {} #Creating an empty dictionary.
```

2) Accessing Values :

```
print(my_dictionary['name'])
print(my_dict.get('age', 'key not found'))
 # .get() to avoid errors.
```

3) Adding / Updating Values :

```
my_dic['age'] = 26 #Updates the key value
my_dic['country'] = 'USA' #Add a new key-value
                          pair.
```

4) Removing Elements :

```
my_dictionary.pop('city') #Removes the key 'city',
                          raises error if not found
del my_dictionary['name'] #Removes 'name' key.
```

Conclusion: These examples / codes showcase how sets and the dictionaries work in python, along with their common operations.

5) Iterating through a Dictionary :

```
for key, value in my_dictionary.items():
    print(f'{key} : {value}')
```

Conclusion: These examples / codes showcase how sets & the dictionaries work in python, along with their common operations.