Practical No. 5

Aim: Write a python program to use data structure concept(List,Tuple,Set,Dictionary)

Theory:Python data structures allow us to store and organize data in a form that is easily accessible and customization. Collections are among these data structures. Lists, sets, tuples, and dictionaries are some of the most often used built-in collections.
Python provides several built-in data structures for organizing and storing data. Four fundamental types are lists, tuples, sets, and dictionaries, each with distinct characteristics:

1. List:
Definition: An ordered, mutable collection of items.
Syntax: Defined using square brackets [].
Characteristics:
Allows duplicate members.
Elements can be of different data types (heterogeneous).
Supports indexing and slicing for accessing elements.

2.Tuple:
Definition: An ordered, immutable collection of items.
Syntax: Defined using parentheses ().
Characteristics:
Allows duplicate members.
Elements can be of different data types.
Supports indexing and slicing.
Once created, elements cannot be modified, added, or removed.

3. Set:
Definition: An unordered collection of unique, immutable items.
Syntax: Defined using curly braces {} or the set() constructor.
Characteristics:
Does not allow duplicate members; duplicates are automatically removed.
Elements must be immutable (e.g., numbers, strings, tuples).
Does not support indexing or slicing due to lack of order.
Useful for membership testing and mathematical set operations (union, intersection, etc.).

4. Dictionary:
Definition: An unordered (in versions prior to Python 3.7, ordered by insertion in 3.7+), mutable collection of key-value pairs.
Syntax: Defined using curly braces {} with key: value pairs.
Characteristics:
Keys must be unique and immutable (e.g., numbers, strings, tuples).
Values can be of any data type and can be duplicated.

Provides efficient lookup of values based on their keys.

Program:

```
print("----- LIST OPERATIONS -----")
# Creating a list
my_list = ["apple", "banana", "cherry", "apple"]
print("Original List:", my_list)

# Accessing elements
print("First element:", my_list[0])
print("Last element:", my_list[-1])

# Adding elements
my_list.append("orange")          # add at end
my_list.insert(1, "kiwi")        # add at specific index
print("After adding elements:", my_list)

# Removing elements
my_list.remove("banana")          # remove by value
popped_item = my_list.pop()      # remove last element
print("After removing elements:", my_list)
print("Popped item:", popped_item)

# Other operations
print("Length of list:", len(my_list))
print("List slice [1:3]:", my_list[1:3])
print("Is 'apple' in list?", "apple" in my_list)

print("\n----- TUPLE OPERATIONS -----")
# Creating a tuple
my_tuple = ("apple", "banana", "cherry", "apple")
print("Original Tuple:", my_tuple)

# Accessing elements
print("First element:", my_tuple[0])
print("Slice [1:3]:", my_tuple[1:3])

# Tuple is immutable, so cannot add/remove elements
# You can create a new tuple by concatenation
new_tuple = my_tuple + ("orange",)
print("New Tuple after adding element:", new_tuple)

print("\n----- SET OPERATIONS -----")
```

```
# Creating a set
my_set = {"apple", "banana", "cherry", "apple"}   # duplicates removed automatically
print("Original Set:", my_set)

# Adding elements
my_set.add("orange")
my_set.update(["kiwi", "mango"])   # add multiple elements
print("After adding elements:", my_set)

# Removing elements
my_set.remove("banana")     # error if element not present
my_set.discard("grapes")   # no error if element not present
popped_item = my_set.pop()   # removes a random element
print("After removing elements:", my_set)
print("Popped element:", popped_item)

# Other operations
print("Length of set:", len(my_set))
print("Is 'apple' in set?", "apple" in my_set)

print("\n----- DICTIONARY OPERATIONS -----")
# Creating a dictionary
my_dict = {"a": "apple", "b": "banana", "c": "cherry"}
print("Original Dictionary:", my_dict)

# Accessing elements
print("Value for key 'b':", my_dict["b"])
print("Keys:", my_dict.keys())
print("Values:", my_dict.values())

# Adding/updating elements
my_dict["d"] = "orange"        # add
my_dict["b"] = "blueberry"   # update
print("After adding/updating:", my_dict)

# Removing elements
del my_dict["a"]                    # remove by key
removed_value = my_dict.pop("c")   # remove by key and return value
print("After removing elements:", my_dict)
print("Removed value:", removed_value)

# Other operations
print("Length of dictionary:", len(my_dict))
print("Is key 'b' present?", "b" in my_dict)
```

Output:

```
----- LIST OPERATIONS -----
Original List: ['apple', 'banana', 'cherry', 'apple']
First element: apple
Last element: apple
After adding elements: ['apple', 'kiwi', 'banana', 'cherry', 'apple', 'orange']
After removing elements: ['apple', 'kiwi', 'cherry', 'apple']
Popped item: orange
Length of list: 4
List slice [1:3]: ['kiwi', 'cherry']
Is 'apple' in list? True

----- TUPLE OPERATIONS -----
Original Tuple: ('apple', 'banana', 'cherry', 'apple')
First element: apple
Slice [1:3]: ('banana', 'cherry')
New Tuple after adding element: ('apple', 'banana', 'cherry', 'apple', 'orange')

----- SET OPERATIONS -----
Original Set: {'cherry', 'apple', 'banana'}
After adding elements: {'mango', 'cherry', 'banana', 'apple', 'kiwi', 'orange'}
After removing elements: {'cherry', 'apple', 'kiwi', 'orange'}
Popped element: mango
Length of set: 4
Is 'apple' in set? True

----- DICTIONARY OPERATIONS -----
Original Dictionary: {'a': 'apple', 'b': 'banana', 'c': 'cherry'}
Value for key 'b': banana
Keys: dict_keys(['a', 'b', 'c'])
Values: dict_values(['apple', 'banana', 'cherry'])
After adding/updating: {'a': 'apple', 'b': 'blueberry', 'c': 'cherry', 'd': 'orange'}
After removing elements: {'b': 'blueberry', 'd': 'orange'}
Removed value: cherry
Length of dictionary: 2
Is key 'b' present? True
```

Result:Thus we have successfully performed the operations on data structure in python.