Experiment No. 3
To explore basic data types of python like strings, list, dictionaries and tuples
Date of Performance:
Date of Submission:

Experiment No. 3

Title: To explore basic data types of python like strings, list, dictionaries and tuples.

Aim: To study and explore basic data types of python like strings, list, dictionaries and tuples.

Objective: To introduce basic data types of python

Theory:

Lists: are just like dynamic sized arrays, declared in other languages (vector in C++ and ArrayList in Java). Lists need not be homogeneous always which makes it a most powerful tool in Python.

Tuple: A Tuple is a collection of Python objects separated by commas. In someways a tuple is similar to a list in terms of indexing, nested objects and repetition but a tuple is immutable unlike lists that are mutable.

Set: A Set is an unordered collection data type that is iterable, mutable and has no duplicate elements. Python's set class represents the mathematical notion of a set.

Dictionary: in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key value is provided in the dictionary to make it more optimized.

List, Tuple, Set, and Dictionary are the data structures in python that are used to store and organize the data in an efficient manner.

List	**Tuple**	**Set**	**Dictionary**
List is a non-homogeneous data structure which stores the elements in single row and multiple rows and columns	Tuple is also a non-homogeneous data structure which stores single row and multiple rows and columns	Set data structure is also non-homogeneous data structure but stores in single row	Dictionary is also a non-homogeneous data structure which stores key value pairs
List can be represented by [ ]	Tuple can be represented by ( )	Set can be represented by { }	Dictionary can be represented by { }
List allows duplicate elements	Tuple allows duplicate elements	Set will not allow duplicate elements	Set will not allow duplicate elements but keys are not duplicated
List can use nested among all	Tuple can use nested among all	Set can use nested among all	Dictionary can use nested among all
Example: [1, 2, 3, 4, 5]	Example: (1, 2, 3, 4, 5)	Example: {1, 2, 3, 4, 5}	Example: {1, 2, 3, 4, 5}
List can be created using **list()** function	Tuple can be created using **tuple()** function.	Set can be created using **set()** function	Dictionary can be created using **dict()** function.

List is mutable i.e we can make any changes in list.	Tuple is immutable i.e we can not make any changes in tuple	Set is mutable i.e we can make any changes in set. But elements are not duplicated.	Dictionary is mutable. But Keys are not duplicated.
List is ordered	Tuple is ordered	Set is unordered	Dictionary is ordered
		Creating a set a=set()	
Creating an empty list l=[]	Creating an empty Tuple t=()	b=set(a)	

Code:

```
# Array

cars = ["Ford", "Volvo", "BMW","Rolls Royce","Thar"]

print("Before modification:",cars)

cars[1] = "Toyota"

print("After modification:",cars)

x = len(cars)

print("length of array:",x)


# Lists

fruits = ["Apple", "Orange", "Strawberry","Mango","Apple"]

print("List is:",fruits)
```

```
print("Length of list is:",len(fruits))

fruits.append("orange")

print(fruits)


#Tuple

actors = ("Rashmika", "Nayantara", "keerthy", "Samantha", "Trisha","Anushka")

print("Tuple is:",actors)

x = list(actors)

x[1] = "Alia"

actors = tuple(x)

print("Tuple is:",actors)

print("Length of Tuple is:",len(actors))

print(actors[-5:-2])


# Sets

thisset = {"apple", "Blue", "Rashmika", True, 1, 2}

print(thisset)

thisset.add("orange")

print(thisset)

thisset.remove("Blue")

print(thisset)
```

```
# Dictionary

thisdict = {"brand": "Ford","model": "Mustang","year": 1964}

print(thisdict)

print(thisdict["year"])

thisdict["color"] = "red"

print(thisdict)

thisdict.pop("model")

print(thisdict)
```

Output:

```
C:\Users\Student\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\Student\PycharmProjects\pythonProject\main.py
Before modification: ['Ford', 'Volvo', 'BMW', 'Rolls Royce', 'Thar']
After modification: ['Ford', 'Toyota', 'BMW', 'Rolls Royce', 'Thar']
length of array: 5
List is: ['Apple', 'Orange', 'Strawberry', 'Mango', 'Apple']
Length of list is: 5
['Apple', 'Orange', 'Strawberry', 'Mango', 'Apple', 'orange']
Tuple is: ('Rashmika', 'Nayantara', 'keerthy', 'Samantha', 'Trisha', 'Anushka')
Tuple is: ('Rashmika', 'Alia', 'keerthy', 'Samantha', 'Trisha', 'Anushka')
Length of Tuple is: 6
('Alia', 'keerthy', 'Samantha')
{'Blue', True, 2, 'apple', 'Rashmika'}
{'Blue', True, 2, 'apple', 'orange', 'Rashmika'}
{True, 2, 'apple', 'orange', 'Rashmika'}
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
1964
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
{'brand': 'Ford', 'year': 1964, 'color': 'red'}

Process finished with exit code 0
```

Conclusion:

Array: The array 'cars' is initially defined and modified by replacing the second element with "Toyota". The length of the array is printed. No specific conclusion can be drawn as the array operations are straightforward.

Lists: The list 'fruits' is defined with some elements, its length is printed, and an element "orange" is appended to it. The conclusion here is that lists in Python are mutable, allowing for modifications such as appending elements.

Tuple: The tuple 'actors' is defined and then converted to a list to modify its second element. After modification, the list is converted back to a tuple. This demonstrates that tuples are immutable, but you can convert them to a mutable data structure like a list to perform modifications.

Sets: The set 'thisset' is defined with a mix of data types, elements are added and removed. The conclusion here is that sets are unordered collections of unique elements where you can easily add or remove items.

Dictionary: The dictionary 'thisdict' is defined and its elements are accessed and modified using keys. The conclusion is that dictionaries are mutable and allow for dynamic addition, modification, and removal of key-value pairs.