Experiment No. 3
To explore basic data types of python like strings, list, dictionaries and tuples
Date of Performance: 6-02-2024
Date of Submission: 27-02-2024

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	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

row and multiple rows and columns			
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:

```
# String
string_variable = "Hello, World!"
print("String variable:", string_variable)

# List
list_variable = [1, 2, 3, 4, 5]
print("List variable:", list_variable)

# Dictionary
dictionary_variable = {'name': 'John', 'age': 30, 'city': 'New York'}
print("Dictionary variable:", dictionary_variable)

# Tuple
tuple_variable = (1, 2, 3, 4, 5)
print("Tuple variable:", tuple_variable)

# Accessing elements
print("\nAccessing elements:")
print("First character of string:", string_variable[0])
print("Second element of list:", list_variable[1])
print("Value associated with 'name' key in dictionary:", dictionary_variable['name'])
print("Third element of tuple:", tuple_variable[2])
```

```
# Modifying elements

list_variable[0] = 10

dictionary_variable['age'] = 35

print("\nModified list variable:", list_variable)

print("Modified dictionary variable:", dictionary_variable)


# Length of sequences

print("\nLength of sequences:")

print("Length of string:", len(string_variable))

print("Length of list:", len(list_variable))

print("Length of dictionary:", len(dictionary_variable))

print("Length of tuple:", len(tuple_variable))
```

RESULT:

```
C:\Users\anand\PycharmProjects\pythonProject1\.venv\Scripts\python.exe C:\Users\anand\PycharmProjects\pythonProject1\main.py
String variable: Hello, World!
List variable: [1, 2, 3, 4, 5]
Dictionary variable: {'name': 'John', 'age': 30, 'city': 'New York'}
Tuple variable: (1, 2, 3, 4, 5)

Accessing elements:
First character of string: H
Second element of list: 2
Value associated with 'name' key in dictionary: John
Third element of tuple: 3

Modified list variable: [10, 2, 3, 4, 5]
Modified dictionary variable: {'name': 'John', 'age': 35, 'city': 'New York'}

Length of sequences:
Length of string: 13
Length of list: 5
Length of dictionary: 3
Length of tuple: 5
```

Conclusion: The study and implementation of basic data types in Python, including strings, lists, dictionaries, and tuples, offer fundamental insights into data manipulation and storage within the language. By exploring these data structures, programmers gain essential skills for organizing and processing information efficiently in Python programs. Understanding how to create, access, and modify elements within each data type enables developers to build versatile and robust applications tailored to diverse use cases. Python's rich set of built-in data types fosters code clarity and flexibility, facilitating rapid development and maintenance of software projects. Mastery of these foundational concepts lays a solid groundwork for tackling more complex data-related challenges and leveraging Python's extensive ecosystem for advanced data processing and analysis tasks.