|  |
| --- |
| Experiment No. 3 |
| To explore basic data types of python like strings, list, dictionaries and tuples |
| Date of Performance: 05/02/2024 |
| Date of Submission: 12/02/2024 |

**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 Tuple is also a non-

structure which homogeneous data Set data structure Dictionary is also a

stores the structure which stores is also non- non-homogeneous

elements in single single row and homogeneous data data structure which

row and multiple multiple rows and structure but stores stores key value

rows and columns columns in single row pairs

Tuple can be

represented by

List can be ( ) Set can be Dictionary can be

represented by [ ] represented by { } represented by { }

Set will not allow

duplicate elements

List allows Tuple allows Set will not allow but keys are not

duplicate elements duplicate elements duplicate elements duplicated

List can use Tuple can use nested Set can use nested Dictionary can use

nested among all among all among all nested among all

Example: [1, 2, 3, Example: (1, 2, 3, 4, Example: {1, 2, 3, Example: {1, 2, 3, 4,

4, 5] 5) 4, 5} 5}

Dictionary can be

List can be created Tuple can be created Set can be created created

using **list()** function using **tuple()** function. using **set()** function using **dict()** function.

Set is mutable i.e

we can make any

List is mutable i.e Tuple is immutable changes in set. But Dictionary is

we can make any i.e we can not make elements are not mutable. But Keys

changes in list. any changes in tuple duplicated. are not duplicated.

List is ordered Tuple is ordered Set is unordered Dictionary is ordered

Creating a set

a=set()

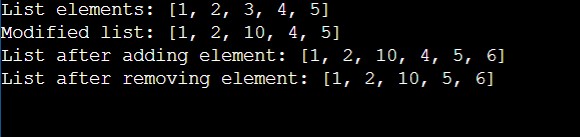
Creating an empty Creating an empty

list Tuple

l=[] t=() b=set(a)

**Code: List** my\_list = [1, 2, 3, 4, 5] print("List elements:", my\_list) my\_list[2] = 10 print("Modified list:", my\_list) my\_list.append(6) print("List after adding element:", my\_list) my\_list.remove(4) print("List after removing element:", my\_list)

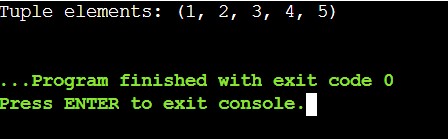
**Output:**



**Code: Tuple**

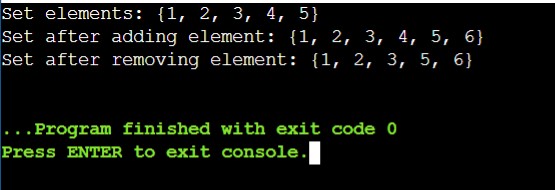
my\_tuple = (1, 2, 3, 4, 5) print("Tuple elements:", my\_tuple)

**Output:**

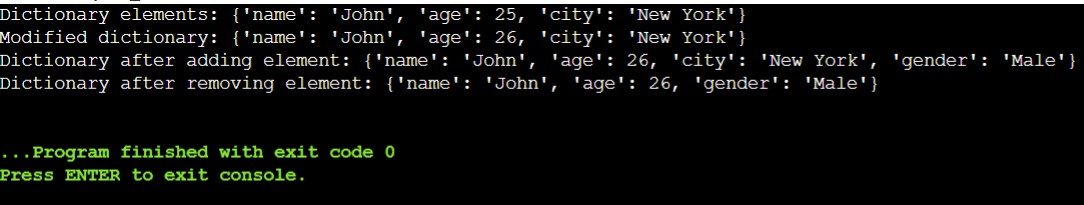


**Code: Set** my\_set = {1, 2, 3, 4, 5} print("Set elements:", my\_set) my\_set.add(6) print("Set after adding element:", my\_set) my\_set.remove(4) print("Set after removing element:", my\_set)

**Output:**



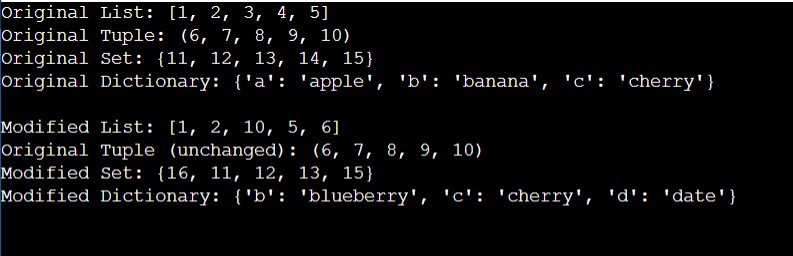
**Code: Dictionary** my\_dict = {'name': 'John', 'age': 25, 'city': 'New York'} print("Dictionary elements:", my\_dict) my\_dict['age'] = 26 print("Modified dictionary:", my\_dict) my\_dict['gender'] = 'Male' print("Dictionary after adding element:", my\_dict) my\_dict.pop('city') print("Dictionary after removing element:", my\_dict) **Output:**



**Code: Python program for List, Tuple, Set and Dictionary** my\_list = [1, 2, 3, 4, 5] my\_tuple = (6, 7, 8, 9, 10) my\_set = {11, 12, 13, 14, 15} my\_dict = {'a': 'apple', 'b': 'banana', 'c': 'cherry'} print("Original List:", my\_list) print("Original Tuple:", my\_tuple) print("Original Set:", my\_set) print("Original Dictionary:", my\_dict) my\_list.append(6) my\_list[2] = 10 my\_list.remove(4) my\_set.add(16) my\_set.remove(14)

my\_dict['d'] = 'date' my\_dict['b'] = 'blueberry' my\_dict.pop('a') print("\nModified List:", my\_list) print("Original Tuple (unchanged):", my\_tuple) print("Modified Set:", my\_set) print("Modified Dictionary:", my\_dict)

**Output:**



**Conclusion:**

These basic data types are fundamental to working effectively with Python. Each data type has its own characteristics, use cases, and operations. Strings are used for representing text data, lists for ordered collections, dictionaries for key-value mappings, and tuples for immutable sequences. Mastery of these data types is crucial for writing clear, concise, and efficient Python code.