|  |
| --- |
| 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 an empty list  l=[] | Creating an empty Tuple  t=() | Creating a set  a=set()    b=set(a) |  |

**Code :**

my\_list = [1, 2, 3, 4, 5]

print("Original List:", my\_list)

my\_list.append(6)

print("List after appending 6:", my\_list)

my\_list.insert(2, 10)

print("List after inserting 10 at index 2:", my\_list)

my\_list.remove(4)

print("List after removing 4:", my\_list)

popped\_element = my\_list.pop(3)

print(f"Popped element at index 3: {popped\_element}, List after popping:", my\_list)

list\_length = len(my\_list)

print("Length of the list:", list\_length)

element\_to\_check = 5

is\_in\_list = element\_to\_check in my\_list

print(f"Is {element\_to\_check} in the list? {is\_in\_list}")

element\_to\_find = 2

index\_of\_element = my\_list.index(element\_to\_find)

print(f"Index of {element\_to\_find}: {index\_of\_element}")

my\_list.reverse()

print("Reversed List:", my\_list)

my\_list.sort()

print("Sorted List in ascending order:", my\_list)

my\_list.clear()

print("List after clearing all elements:", my\_list)

my\_tuple = (1, 2, 3, 'four', 5.0)

for index, value in enumerate(my\_tuple):

print(f"Index {index}: {value}")

tuple\_length = len(my\_tuple)

for count in range(tuple\_length):

print(f"Count {count}: {my\_tuple[count]}")

**Output :**

Original List: [1, 2, 3, 4, 5]

List after appending 6: [1, 2, 3, 4, 5, 6]

List after inserting 10 at index 2: [1, 2, 10, 3, 4, 5, 6]

List after removing 4: [1, 2, 10, 3, 5, 6]

Popped element at index 3: 3, List after popping: [1, 2, 10, 5, 6]

Length of the list: 5

Is 5 in the list? True

Index of 2: 1

Reversed List: [6, 5, 10, 2, 1]

Sorted List in ascending order: [1, 2, 5, 6, 10]

List after clearing all elements: []

Index 0: 1

Index 1: 2

Index 2: 3

Index 3: four

Index 4: 5.0

Count 0: 1

Count 1: 2

Count 2: 3

Count 3: four

Count 4: 5.0

**Conclusion:**

Basic data types of python has been studied and implemented.