# S. Y. B. Tech (ECE)

**Trimester: VI Subject: Linux Based Python Laboratory (CET2005A)**

# Name: Class:

**Roll No.: Batch:**

# Experiment – 04 Title: Introduction to Basic Data Structures of Python Performed on:

**Marks**

**Teacher’s Signature with date**

**Submitted on:**

**Aim**: Introduction to Basic Data Structures of Python.

# Objective:

1. To know the Basic Data Structures of Python.
2. To perform different operations on List and Set data structure.

# Theory:

The Data Structures in the Python Programming Language and how they are related to some specific Python Data Types. We will discuss all the in-built data structures like list, set, tuples, dictionaries, etc.

# Lists:

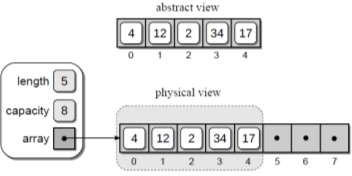
Python Lists are just like the arrays, declared in other languages which is an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

The implementation of Python List is similar to Vectors in C++ or ArrayList in JAVA. The costly operation is inserting or deleting the element from the beginning of the List as all the elements are needed to be shifted. Insertion and deletion at the end of the list can also become costly in the case where the preallocated memory becomes full.

Python's list structure is a mutable sequence container that can change size as items are added or removed. It is an abstract data type that is implemented using an array structure to store the items contained in the list.

# Creating a Python List:

Suppose we create a list containing several values: pyList = [ 4, 12, 2, 34, 17 ]

Figure 2.2 illustrates the abstract and physical views of our sample list. In the physical view, the elements of the array structure used to store the actual contents of the list are enclosed inside the dashed gray box. The elements with null references shown outside the dashed gray box are the remaining elements of the underlying array structure that are still available for use. This notation will be used throughout the section to illustrate the contents of the list and the underlying array used to implement it.

The length of the list, obtained using len(), is the number of items currently in the subarray and not the size of the underlying array.

# The list data type has some methods. Here are some of the methods of list objects:

**list.append(x)**

Add an item to the end of the list. Equivalent to a[len(a):] = [x].

# list.extend(iterable)

Extend the list by appending all the items from the iterable. Equivalent to a[len(a):] = iterable.

# list.insert(i, x)

Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0, x) inserts at the front of the list, and a.insert(len(a), x) is equivalent to **a.append(x).**

# list.remove(x)

Remove the first item from the list whose value is equal to x. It raises a ValueError if there is no such item.

# list.pop([i])

Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list. (The square brackets around the i in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

# Sets in Python:

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. The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set. This is based on a data structure known as a hash table. Since sets are unordered, we cannot access items using indexes like we do in lists.

# Creation of Sets:

There are two main methods of creating a set. One method is to use the set function that is available in Python. The other method is to use the curly braces ‘{}’ and type the list of various elements.

The sets that are created cannot be indexed because they are unordered elements. If any element is repeated in the set, it is not accounted for and will be disregarded. The elements in a set are always arranged in ascending order.

# # Python program to demonstrate sets

myset = set(["a", "b", "c"]) print(myset)

# Adding element to the set myset.add("d") print(myset)

# Methods for Sets:

**Adding elements- set.add()**

Insertion in set is done through **set.add()** function, where an appropriate record value is created to store in the hash table.

# Union

Two sets can be merged using union() function or | operator. Both Hash Table values are accessed and traversed with merge operation perform on them to combine the elements, at the same time duplicates are removed.

# Intersection

This can be done through intersection() or & operator. Common Elements are selected. They are similar to iteration over the Hash lists and combining the same values on both the Table.

# Difference

To find difference in between sets. Similar to find difference in linked list. This is done through difference() or – operator.

**Input**: Data in List and Set

**Output:** Different operations on List and Set

# Conclusion:

**Post Lab Questions:**

1. Describe basic data structures in Python
2. Write a python program to check even numbers from a set of numbers from 1 to 50.
3. Write a Python Program to Concatenate Two Lists
4. Write a Python program to perform any four operations on set.

# Additional Reference Links:

1. [https://www.python.org](https://www.python.org/)
2. [https://www.tutorialspoint.com](https://www.tutorialspoint.com/)
3. <https://www.programiz.com/python-programming>