#  Lab 9

#  Lists

Python offers a range of compound data types often referred to as sequences. List is one of the most frequently used and very versatile data types used in Python.

A list is a data type that allows you to store various types of data in it.

List is a compound data type which means you can have different-2 data types under a list, for example we can have integer, float and string items in a same list.

**Create a list**

In Python programming, a list is created by placing all the items (elements) inside square brackets [], separated by commas.

It can have any number of items and they may be of different types (integer, float, string etc.).

*List-Name = [Element 1, Element 2…..]*

**Program 1**

*# empty list*

*my\_list = []*

*# list of integers*

*my\_list = [1, 2, 3]*

*# list with mixed data types*

*my\_list = [1, "Hello", 3.4]*

**Output:**

**Access list elements**



**List Index**

We can use the index operator [] to access an item in a list. In Python, indices start at 0. So, a list having 5 elements will have an index from 0 to 4.

Trying to access indexes other than these will raise an IndexError. The index must be an integer. We can't use float or other types, this will result in TypeError.

Nested lists are accessed using nested indexing.

**Program 2**

*# List indexing*

*my\_list = ['p', 'r', 'o', 'b', 'e']*

*# Output: p*

*print(my\_list[0])*

*# Output: o*

*print(my\_list[2])*

*# Output: e*

*print(my\_list[4])*

*# Nested List*

*n\_list = ["Happy", [2, 0, 1, 5]]*

*# Nested indexing*

*print(n\_list[0][1])*

*print(n\_list[1][3])*

*# Error! Only integer can be used for indexing*

*print(my\_list[4.0])*

**Output:**

**Negative indexing**

Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.

**Program 3**

*# Negative indexing in lists*

*my\_list = ['p','r','o','b','e']*

*print(my\_list[-1])*

*print(my\_list[-5])*

**Output**

**Slice list**

We can access a range of items in a list by using the slicing operator :(colon).

**Program 4**

*my\_list = ['p','r','o','g','r','a','m','i','z']*

*# elements 3rd to 5th*

*print(my\_list[2:5])*

*# elements beginning to 4th*

*print(my\_list[:-5])*

*# elements 6th to end*

*print(my\_list[5:])*

*# elements beginning to end*

*print(my\_list[:])*

**Output**

**change elements to a list**

We can use the assignment operator (=) to change an item or a range of items.

**Program 5**

*# Correcting mistake values in a list*

*odd = [2, 4, 6, 8]*

*# change the 1st item*

*odd[0] = 1*

*print(odd)*

*# change 2nd to 4th items*

*odd[1:4] = [3, 5, 7]*

*print(odd)*

**Output**

**Add Element to list**

We can add one item to a list using the append() method or add several items using extend() method.

Furthermore, we can insert one item at a desired location by using the method insert() or insert multiple items by squeezing it into an empty slice of a list.

**Program 6.a**

*# Appending and Extending lists in Python*

*odd = [1, 3, 5]*

*odd.append(7)*

*print(odd)*

*odd.extend([9, 11, 13])*

*print(odd)*

**Output**

**Program 6.b**

*# Demonstration of list insert() method*

*odd = [1, 9]*

*odd.insert(1,3)*

*print(odd)*

*odd[2:2] = [5, 7]*

*print(odd)*

**Output**

**combine two lists**

We can also use + operator to combine two lists. This is also called concatenation. The \* operator repeats a list for the given number of times.

**Program 6**

*odd = [1, 3, 5]*

*print(odd + [9, 7, 5])*

*print(["re"] \* 3)*

**Output**

**Delete elements from a list**

We can delete one or more items from a list using the keyword del. It can even delete the list entirely.

**Program 8**

*my\_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']*

*# delete one item*

*del my\_list[2]*

*print(my\_list)*

*# delete multiple items*

*del my\_list[1:5]*

*print(my\_list)*

*# delete entire list*

*del my\_list*

*# Error: List not defined*

*print(my\_list)*

**Output**

**Remove elements from a list**

We can use remove() method to remove the given item or pop() method to remove an item at the given index.

The pop() method removes and returns the last item if the index is not provided. This helps us implement lists as stacks (first in, last out data structure).

We can also use the clear() method to empty a list.

**Program 9**

*my\_list = ['p','r','o','b','l','e','m']*

*my\_list.remove('p')*

*# Output: ['r', 'o', 'b', 'l', 'e', 'm']*

*print(my\_list)*

*# Output: 'o'*

*print(my\_list.pop(1))*

*# Output: ['r', 'b', 'l', 'e', 'm']*

*print(my\_list)*

*# Output: 'm'*

*print(my\_list.pop())*

*# Output: ['r', 'b', 'l', 'e']*

*print(my\_list)*

*my\_list.clear()*

*# Output: []*

*print(my\_list)*

**Output**

**Lab Task**

Create your own list of 10 elements and reapply all the operations in today’s lab on it.

*Marks Obtained: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Instructor’s Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*