

MCT-242 : COMPUTER PROGRAMMING-I

using Python 3.9



Prepared By:
Mr. Muhammad Ahsan Naeem



YouTube Playlist

https://www.youtube.com/playlist?list=PLWF9TXck7O_wMDB-VriREZ6EvwkWLNb7q

LAB 16: LIST: CLO-4

Recall the task we did in one lab for entering marks of five subjects and calculating the mean of those. Firstly, we did it by declaring 5 variables and later we did it using a `for` loop and summing up all values. What if you want to store the marks of the whole class comprising of 50 students? Declaring 50 variables for this would not be a good solution. Secondly, using a `for` loop even for the case of 5 subject marks, the individual marks were never saved, we just kept adding entered marks and could not access the individual entries.

The solution for such cases is the use of `list` which is defined as single variable but can hold multiple values in it. The entries of `list` are generally known as elements or items.

Defining and printing a list:

A list can be defined by specifying its elements inside `[]` as shown here:

```
a=[5, 300, -20]
```

Here `a` is the name of the list having three numbers. Another example with string elements is shown here:

```
b=['Python', 'Computer', 'Programming', 'Course']
```

Interestingly, we can have elements of a list with different data types as shown below:

```
c=[10, 20, 'Programming', 100]
```

Printing a list is as simple as printing any variable. Use `print(a)` for the first list and respective list names for other two to see the displayed output.

Accessing and Updating list elements:

We can always access the individual elements of a list using the index of the element inside `[]`. The index of the list elements starts from `0`. So, if we want to access second element of a list it can be done as:

```
a=[5, 300, -20]
print(a[1])
```

Of course, other than printing `a[1]` we can always use it as any other variable for example; `x=a[1]+10` or storing the user input.

We can also update the value of any element as shown here:

```
a=[5, 300, -20]
a[2]=20
print(a)
```

You will see in the output that -20 is changed to 20.

If you will try to access some element beyond the index range of that list, you will get this error:

IndexError: list index out of range

Can the index be negative?

In most of the programming languages it cannot be. But Python has the feature of negative indices in reverse direction. Each element can be accessed through two indices. One, the positive index in forward direction as seen in above examples and the other is a negative index in reverse direction. The last element of the list has index -1, second last has -2 and so on till the first element. So, a list shown below has two set of indices:

```
myList=[30,45,"Hello","Python",950]
```

List Elements	30	45	Hello	Python	950
Positive Indices	0	1	2	3	4
Negative Indices	-5	-4	-3	-2	-1

List Slicing

Other than one element from of the list, we can access multiple elements at a time, known as **List Slicing**. Recall how we used `range()` function earlier and we can provide start, stop and step parameter. Likewise, as List indices, we can specify those by providing start, stop and step values.

Consider these examples:

```
a=[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9] Positive Indices
# [-10,-9,-8,-7,-6,-5,-4,-3,-2,-1] Negative Indices

print(a[0:4])      # Will print [1, 2, 3, 4]
print(a[2:4])      # Will print [3, 4]
print(a[0:9:2])    # Will print [1, 3, 5, 7, 9]
# We can also specify the last index as -1
# So that we do not need to know total number of List elements.
print(a[0:-1:2])   # Will print [1, 3, 5, 7, 9]
print(a[2:-2])     # Will print [3, 4, 5, 6, 7, 8]
```

Traversing/Scanning a List:

If we need to traverse a complete list, say we want the product of each element of list, we can use a **for** loop for that. For example, for a list named `myList` with five elements, we can do it like:

```
myList=[30,45.2,33,42,15.8]
p=1
for i in range(5):
    p*=myList[i]
print(f"The product of list numbers is: {p}")
```

The above method is good to scan a list of five elements. For list of some other size we will use that number in range parameter. But a much better way is to use a Python built-in function `len()` that gives the length i.e. size of the list that is passed as the input parameter. For example, `len(myList)` will return 5 in above case. Therefore, to scan a complete list, the better approach will be:

```
myList=[30,45.2,33,42,33,42,15.8]
p=1
for i in range(len(myList)):
    p*=myList[i]
print(f"The product of list numbers is: {p}")
```

List Membership Test using “in” Operator:

Like we have used the `in` operator with `range()` function, we can use it on list elements as well. It can be used to check if an element is inside a list or not. See this example:

```
myList=[30,45.2,33,42,33,42,15.8]
print(30 in myList) # Will print True
print(32 in myList) # Will print False
```

See the example below to find if the list contains number 3 or not:

```
a = [7,3,-6,10,6]
if 3 in a:
    print("3 is in the list")
```

We can also use `not in` as check for non-existence of some element in the list as:

```
a = [7,3,-6,10,6]
if 3 not in a:
    print("3 is not in list")
else:
    print("3 is in the list")
```

Even a better way to use `in` operator is directly using it on list elements. For example, like:

```
for i in myList:
```

In such case, the loop variable `i` is not the index but it is the list element starting from first element in the first iteration to the last element in last iteration. Hence the task of finding the product of all list numbers can be done as:

```
myList=[30,45.2,33,42,33,42,15.8]
p=1
for i in myList:
    p*=i
print(f"The product of list numbers is: {p}")
```

You can see this way is simple and we do not need to access each element through indexing, the loop variable itself is the element of the list. We will be using this format as much possible but at times we may need to access the element by index.

Similarly, if we have a list with string elements and we want to display each element twice on the screen we can do as:

```
fruits = ['apple', 'mango', 'banana']
for i in fruits:
    print(i, "\t", i)
```

This will have following output:

```
apple    apple
mango    mango
banana   banana
```

Tasks:

- [1] Write a program that will find the average of the numbers inside a list. The list may contain any number of values. In your program, define a list with a few entries as shown in above example, then use a **for** loop to find the sum of numbers in that list. After the loop, use that sum to find and display the average.

Note: There is a built-in Python function named as **sum()** which can be applied on a list to find the sum of elements but at present do not use it and do this by finding the sum by scanning the whole list.

- [2] Write a program that will replace all list entries which are greater than 10 with 10 and negative entries with 0. You have to do it as:
- Start the program by defining a list with few entries similar to above task.
 - Display the complete list.
 - Use a **for** loop to scan the complete list. Within **for** loop you will check if list element is greater than **10** then update that element to **10**. Likewise, the entries less than **0** should be replaced with **0**.
 - Display the complete list after the entries have been updated.

Sample Output is:

```
Initial List: [5, 6, -2.3, 12, 4, 13.8, 100, 5, -2]
Final List: [5, 6, 0, 10, 4, 10, 10, 5, 0]
```

- [3] Write a program that will display number of primes in a list.

Sample Output is:

```
List Elements: [5, 6, 8, 12, 4, 13, 100, 5, 21]
There are 3 prime numbers in above list
```

You will do it as:

- a. Start the program by defining a list with few positive integers similar to the above task.
 - b. Scan the list using a `for` loop.
 - c. Within `for` loop, use another `for` loop to find whether that list element is prime or not. If it is prime, increment the count variable.
 - d. Do not define function to determine if the number is prime or not.
 - e. Display the count.
- [4] Repeat above task but now to find whether a number is prime or not, define and use a function named as `isPrime` as we did earlier. Use this function inside the list scanning `for` loop.

[5] Finding the maximum value in a list

In this program define a list with a few entries and then program must find and display the maximum value from list entries.

How to do that:

Suppose here is our initial list:

```
myList=[30,45,33,82,33,42,15]
```

The logic to find maximum out of above list goes something like:

- Start with declaring first value as the maximum value, like: `highest=myList[0]`
- Start scanning the array from second entry till the last one; if the current index value is greater than highest declared, update the highest value as:
`if(myList[i])>highest):`
`highest=myList[i]`

- [6] Update above task so that both the maximum and the minimum value is inside the list is found and displayed. You have to find both values within the same `for` loop.