# MODULE III

**List, Tuples, Set and Dictionaries .**

Python List: Introduction, accessing List, List operations, Working with Lists, List functions and methods

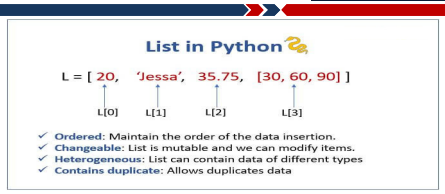
Python Tuple:- Introduction, accessing Tuple, operations on Tuple, Working with Tuple ,Functions and Methods,

Python Set - Introduction, accessing Set, Set operations, working with Set, Functions and Methods,

Python Dictionaries – Introduction, working with dictionaries, Properties, Functions. Dictionaries Operations, List Comprehension.



## PYTHON LIST




**Mutable:** The elements of the list can be modified. We can add or remove items to the list after it has been created.

**Ordered:** The items in the lists are ordered. Each item has a unique index value. The new items will be added to the end of the list.

**Heterogeneous:** The list can contain different kinds of elements i.e; they can contain elements of string, integer, boolean, or any type.

**Duplicates:** The list can contain duplicates i.e., lists can have two items with the same values.



## Why Use a List

•The **list data structure is very flexible** It has many unique inbuilt functionalities like **pop(), append(),** etc which makes it easier, where the data keeps changing.

•The list can **contain duplicate elements** i.e two or more items can have the same values.

•Lists are **Heterogeneous i.e**, different kinds of objects/elements can be added.

•Lists are **mutable** it is used in applications where the values of the items change frequently.



## List creation

- The list can be created using either the **list constructor list()** or using **square brackets [].**

```
# Using list constructor
my_list1 = list((1, 2, 3))
print(my_list1)
# Output [1, 2, 3]

# Using square brackets[]
my_list2 = [1, 2, 3]
print(my_list2)
# Output [1, 2, 3]

# with heterogeneous items
my_list3 = [1.0, 'Jessa', 3]
print(my_list3)
# Output [1.0, 'Jessa', 3]

# empty list using list()
my_list4 = list()
print(my_list4)
# Output []

# empty list using []
my_list5 = []
print(my_list4)
# Output []
```

**Length of a List :** list1 = [1, 2, 3]

**Print ( len** (list1) )

**Accessing items of a List :** The items in a list can be **accessed through indexing** and **slicing**

| P | Y | T | H | O | N |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |
| -6 | -5 | -4 | -3 | -2 | -1 |

Positive Indexing

Negative Indexing

**indexing**

```
my_list = [10, 20, 'Jessa', 12.50, 'Emma']
# accessing 2nd element of the list
print(my_list[1])  # 20
# accessing 5th element of the list
print(my_list[4])  # 'Emma'
```

**Slicing**

```
my_list = [10, 20, 'Jessa', 12.50, 'Emma', 25, 50]
# Extracting a portion of the list from 2nd till 5th element
print(my_list[2:5])
# Output ['Jessa', 12.5, 'Emma']
```

RA20 APTT

```
my_list = [5, 8, 'Tom', 7.50, 'Emma']

# slice first four items
print(my_list[:4])
# Output [5, 8, 'Tom', 7.5]

# print every second element
# with a skip count 2
print(my_list[::2])
# Output [5, 'Tom', 'Emma']

# reversing the list
print(my_list[::-1])
# Output ['Emma', 7.5, 'Tom', 8, 5]

# Without end_value
# Stating from 3nd item to last item
print(my_list[3:])
# Output [7.5, 'Emma']
```

Examples of slicing a list:

- Extract a portion of the list
- Slicing with a step
- Reverse a list
- Slice without specifying start or end position

RA20 APTT

## Iterating a List without and with index number

```
my_list = [5, 8, 'Tom', 7.50, 'Emma']

# iterate a list
for item in my_list:
    print(item)
```

```
my_list = [5, 8, 'Tom', 7.50, 'Emma']

# iterate a list
for i in range(0, len(my_list)):
    # print each item using index number
    print(my_list[i])
```

RA20 APTT

## Adding elements to the list

We can add a new **element / list** of elements to the list using the list methods such as **append() , insert() , and extend().**

**Append item at the end of the list:** The **append()** method will accept **only one** parameter and add it at the **end of** the list.

```
my_list = list([5, 8, 'Tom', 7.50])

# Using append()
my_list.append('Emma')
print(my_list)
# Output [5, 8, 'Tom', 7.5, 'Emma']

# append the nested list at the end
my_list.append([25, 50, 75])
print(my_list)
# Output [5, 8, 'Tom', 7.5, 'Emma', [25, 50, 75]]
```

RA20 APTT

## Adding elements to the list

**Add item at the specified position in the list:** the **insert()** method to add the object / item at the **specified position** in the list. The insert method **accepts two parameters position and object.**

```
my_list = list([5, 8, 'Tom', 7.50])

# Using insert()
# insert 25 at position 2
my_list.insert(2, 25)
print(my_list)
# Output [5, 8, 25, 'Tom', 7.5]

# insert nested list at at position 3
my_list.insert(3, [25, 50, 75])
print(my_list)
# Output [5, 8, 25, [25, 50, 75], 'Tom', 7.5]
```

RA20 APTT

## Adding elements to the list

**Using extend():** The **extend method** will accept the list of elements and add them at the **end of the list**. We can even add another list by using this method.

```
my_list = list([5, 8, 'Tom', 7.50])

# Using extend()
my_list.extend([25, 75, 100])
print(my_list)
# Output [5, 8, 'Tom', 7.5, 25, 75, 100]
```

RA20 APTT

## Modify the items of a List

- The list is a mutable sequence of **iterable objects**.
- It means we can modify the items of a list.
- Use the index number and assignment operator (=) to assign a new value to an item.
- **Modify the individual item** and **Modify the range of items**

```
my_list = list([2, 4, 6, 8, 10, 12])

# modify single item
my_list[0] = 20
print(my_list)
# Output [20, 4, 6, 8, 10, 12]

# modify range of items
# modify from 1st index to 4th
my_list[1:4] = [40, 60, 80]
print(my_list)
# Output [20, 40, 60, 80, 10, 12]

# modify from 3rd index to end
my_list[3:] = [80, 100, 120]
print(my_list)
# Output [20, 40, 60, 80, 100, 120]
```

RA20 APTT

## Removing the items from the List

| method | Description |
|---|---|
| `remove(item)` | To remove the first occurrence of the item from the list. |
| `pop(index)` | Removes and returns the item at the given index from the list. |
| `clear()` | To remove all items from the list. The output will be an empty list. |
| `del list_name` | Delete the entire list. |

RA20 APTT

## Removing Specific items from the List

SR UNIVERSITY

•Use the **remove() method** to remove **the first occurrence** of the item from the list.

• A **keyerror** is thrown if an item not present in the original list.

```
my_list = list([2, 4, 6, 8, 10, 12])

# remove item 6
my_list.remove(6)
# remove item 8
my_list.remove(8)

print(my_list)
# Output [2, 4, 10, 12]
```

### Remove all occurrence of a specific item

```
list1 = [1, 2, 3, 1, 5, 1, 7, 1]

while 1 in list1:
    list1.remove(1)
print(list1)
```

```
list1 = list([1, 2, 3, 1, 5, 1, 7, 1])
for item in list1:
    list1.remove(1)
print(list1)
```

```
Unit2
C:\Users\home\PycharmProjects\SRU\venv\Scripts\python
[2, 3, 5, 7]

Process finished with exit code 0
```

```
list1 = list([1, 2, 3, 1, 5, 1, 7, 1])
list1 = [i for i in list1 if i != 1]
print(list1)
```

```
Unit2
C:\Users\home\PycharmProjects\SRU\venv\Scripts
[2, 3, 5, 7]

Process finished with exit code 0
```

```
# This is called comprehension
```

RA20 APTT

## Removing items at a present index

SR UNIVERSITY

•Use the **pop() method** to remove the item at the given index.

•The **pop() method** removes and returns the **item present at the given index**.

•Remove the **last item** from the list if the **index number is not passed.**

```
my_list = list([2, 4, 6, 8, 10, 12])

# remove item present at index 2
my_list.pop(2)
print(my_list)
# Output [2, 4, 8, 10, 12]

# remove item without passing index number
my_list.pop()
print(my_list)
# Output [2, 4, 8, 10]
```

RA20 APTT

## Removing Range of items

SR UNIVERSITY

•Use **del keyword** along with **list slicing** to remove the range of items

```
my_list = list([2, 4, 6, 8, 10, 12])

# remove range of items
# remove item from index 2 to 5
del my_list[2:5]
print(my_list)
# Output [2, 4, 12]

# remove all items starting from index 3
my_list = list([2, 4, 6, 8, 10, 12])
del my_list[3:]
print(my_list)
# Output [2, 4, 6]
```

RA20 APTT

## Removing All Items

SR UNIVERSITY

•Use **clear() method** to remove all items from the list.

```
my_list = list([2, 4, 6, 8, 10, 12])

# clear list
my_list.clear()
print(my_list)
# Output []

# Delete entire list
del my_list
```

RA20 APTT

## Finding Items in a list

Use the `index()` function to find an item in a list.

The `index()` function will accept the value of the element as a parameter and returns the first occurrence of the element or returns `ValueError` if the element does not exist.

```
my_list = list([2, 4, 6, 8, 10, 12])

print(my_list.index(8))
# Output 3

# returns error since the element does not exist in the list.
# my_list.index(100)
```

RA20 APTT

## Concatenation of two lists

The concatenation of two lists means merging of two lists. There are two ways to do that.

- Using the `+` operator.
- Using the `extend()` method. The `extend()` method appends the new list's items at the end of the calling list.

```
my_list1 = [1, 2, 3]
my_list2 = [4, 5, 6]

# Using + operator
my_list3 = my_list1 + my_list2
print(my_list3)
# Output [1, 2, 3, 4, 5, 6]

# Using extend() method
my_list1.extend(my_list2)
print(my_list1)
# Output [1, 2, 3, 4, 5, 6]
```

RA20 APTT

## Copying a list

•There are two ways to copy of a list can be created . One is using **assignment operator (=)**

•This is a straightforward way of creating a copy and its called **deep copying.**

•The **changes made to the original list are reflected in the copied list as well.**

•When you set list1 = list2, you are making **them refer to the same list object**.

```
my_list1 = [1, 2, 3]

# Using = operator
new_list = my_list1
# printing the new list
print(new_list)
# Output [1, 2, 3]

# making changes in the original list
my_list1.append(4)

# print both copies
print(my_list1)
# result [1, 2, 3, 4]
print(new_list)
# result [1, 2, 3, 4]
```

RA20 APTT

## Use Copy Method

•The copy method can be used to create a copy of a list.

•This will create a new list and any **changes made in the original list** will **not reflect in the new list**. And This is **shallow copying.**

```
my_list1 = [1, 2, 3]

# Using copy() method
new_list = my_list1.copy()
# printing the new list
print(new_list)
# Output [1, 2, 3]

# making changes in the original list
my_list1.append(4)

# print both copies
print(my_list1)
# result [1, 2, 3, 4]
print(new_list)
# result [1, 2, 3]
```

RA20 APTT

## List Operations

SR UNIVERSITY

**Sort List using sort()**

```
mylist = [3,2,1]
mylist.sort()
print(mylist)

[1, 2, 3]
```

**max() & min()**

```
mylist = [3, 4, 5, 6, 1]
print(max(mylist)) #returns the maximum number in the list.
print(min(mylist)) #returns the minimum number in the list.

6
1
```

**Reverse a List using reverse()**

```
mylist = [3, 4, 5, 6, 1]
mylist.reverse()
print(mylist)

[1, 6, 5, 4, 3]
```

**Using sum()**

```
mylist = [3, 4, 5, 6, 1]
print(sum(mylist))

19
```

RA20 APTT

## Nested List

SR UNIVERSITY

```
nestedlist = [[2,4,6,8,10],[1,3,5,7,9]]

print("Accessing the third element of the second list",nestedlist[1][2])
for i in nestedlist:
  print("list",i,"elements")
  for j in i:
    print(j)

Accessing the third element of the second list 5
list [2, 4, 6, 8, 10] elements
2
4
6
8
10
list [1, 3, 5, 7, 9] elements
1
3
5
7
9
```

RA20 APTT

## Summary of List Operations

SR UNIVERSITY

l1 and l2 are lists, x, i, j, k, n are integers.
`l1 = [10, 20, 30, 40, 50]` and `l2 = [60, 70, 80, 60]`

| Operation | Description |
|---|---|
| **X in l1** | Check if the list l1 contains item x. |
| **X not in l2** | Check if list l1 does not contain item x. |
| **l1 + l2** | Concatenate the lists l1 and l2. Creates a new list containing the items from l1 and l2. |
| **l1 * 5** | Repeat the list l1 5 times. |
| **l1[i]** | Get the item at index i. Example l1[2] is 30. |
| **l1[i:j]** | List slicing. Get the items from index i up to index j (excluding j) as a List. An example l1[0:2] is [10, 20] |
| **l1[i:j:k]** | List slicing with step. Returns a List with the items from index i up to index j taking every k-th item. An example l1[0:4:2] is [10, 30]. |
| **len(l1)** | Returns a count of total items in a list. |

RA20 APTT

## Summary of List Operations

SR UNIVERSITY

| | |
|---|---|
| **l2.count(60)** | Returns the number of times a particular item (60) appears in a list. The answer is 2. |
| **l1.index(30)** | Returns the index number of a particular item (30) in a list. The answer is 2. |
| **l1.index(30, 2, 5)** | Returns the index number of a particular item (30) in a list. But search Returns the item with maximum value from a list. The answer is 60 only from index number 2 to 5. |
| **min(l1)** | Returns the item with a minimum value from a list. The answer is 10. |
| **max(l1)** | Returns the item with maximum value from a list. The answer is 60. |
| **l1.append(100)** | Add item at the end of the list |
| **l1.append([2, 5, 7])** | Append the nested list at the end |
| **l1[2] = 40** | Modify the item present at index 2 |
| **l1.remove(40)** | Removes the first occurrence of item 40 from the list. |
| **pop(2)** | Removes and returns the item at index 2 from the list. |
| **l1.clear()** | Make list empty |
| **l3= l1.copy()** | Copy l1 into l2 |

RA20 APTT