List - Python Datastructure

- List represents sequence of elements indexed by their integer position
- Lists in python are mutable, that is elements can be edited later.
- Python list elements can be of any type, each element can any object of python
- It allows repeated elements to be present in a list.

1. Create a list

- Using [ ] or list( )
- By converting from other data types

1.1 Using [ ] or ( )

- A list is made of zero or more elements
- Seperated by commas (,) and surrounded by square brackets [ ]
- `fruits=[]` defining a empty list variable fruits
- List can also be initialized during definition `fruits=["mango","apple"]`
- `list( )` is also used to create an empty list e.g: `fruits=list()`

1.2 By Converting other from data types to list

- Other data types can be converted to python using `list()` method.

```
# Converting a string to list of characters
characters = list("cat")        #OUTPUT: ['c','a','t']

actions_tuple = ("ready","aim","fire")

# Converting tuple to list
action=list(actions_tuple)       #OUTPUT: ["ready","aim","fire"]
```

- `split()` method is also used for splitting strings based on some characters

```
birthday="16-10-1992"
bday=birthday.split('-')         #OUTPUT: ['16','10','1992']
```

2. Accessing elements from list

- Each single value can be extracted from the list using index or offset of the elemet
- Example: `fruits=["mango","apple"]` then `fruits[1] = "apple"`
- Negative offset can also be used where -1 means first item from last.
- slice `[start:end:step]` can also be used to extract all elements within given range as used with strings
- `index()` method is used to find the index of an element eg: `fruits.index('apple')`

3. Inserting Element

- Inserting at end: `.append(element)`
- Inserting at postion: `.insert(position,element)`
- Merging two list: `.extend(secondList)` ,
  - Alternatively `+=secondList` can also be used for same purpose

```
names=["ram","shyam"]

# Append
names.append("Mohan")

# Insert
names.insert(1,"Krishna")

# Extend
list2=["Sita","Radha"]

names.extend(list2)
        #or
names+=list2
```

4. Deleting Element

- Using `del[]` :
    - It is used to delete an element with the given `index` value
    - Eg: `del fruits[1]` removes element at index 1

- Using `pop()`
    - It is used to remove the value and return the removed value
    - It can also be used with an index value
    - By default index value is `-1` , that removes the last element in the list
    - Eg: `fruits.pop()` or `fruits.pop(1)`

`

- Using `remove()`
    - It is used to remove the element based on the value
    - Eg: `fruits.remove("Apple")`

5. Copy Element

- Using `copy()`
- Using `list()`
- Using slice `[start:end:step]`

```
a=[1,2,3,4,"hello",False]

#Using copy
c = a.copy()

#Using list()
d = list(a)

#Using slice
e = a[:]
```

6. Sort Element

- Using `sort()`
  - It sorts the elements in-place, changes original array
  - Eg: `a.sort()`

- Using `sorted()`
  - It reutrn a sorted list
  - It does not make change in original array
  - Eg: `b = sorted(a)`

7. Check presence of an element

- `in` operator is used to check ig an element is in list
- returns `True` or `Flase`
- Eg:

```
a=[1,2,3,4,"hello",False]

'hello' in a #Returns True

10 in a      #Returns False
```

8. Count number of Occurance & Number of elements

- `count()` method is used to numbe rof occuraance of a single element
- Eg: `a.count(1)` will return 1.

- `len()` is used to count total number of elements in list
- Eg: `len(a)` will return 6.

9. List Comprehension

> Comprehension are compact way of writing looping and conditional instructions together.

- Output range in a list
  - `a = list(range(10))`
  - `a=[0,1,2,3,4,5,6,7,8,9]`
- $[expression\ for\ items\ in\ iterable]$
  - `numbers= [x for x in range(1,6)]`
  - `numbers=[1,2,3,4,5]`
- $[expression\ for\ items\ in\ iterable if condition]$
  - `numbers= [x for x in range(1,20) if x%2== 0]`
  - `numbers=[2, 4, 6, 8, 10, 12, 14, 16, 18]`
- Nesting loops

```
rows= list(range(2))
cols= list(range(3))

cells = [(x,y) for x in rows for y in cols]

#output
cells=[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2)]
```