List

- It is one of the most used datatype in Python and is very flexible.

Empty List

- empty list can be created in 2 ways

In [1]:

```
empty_list_1 = []
empty_list_1
```

Out[1]:

```
[]
```

In [2]:

```
empty_list_2 = list()
empty_list_2
```

Out[2]:

```
[]
```

- List is an ordered sequence of items.
- Each element or value that is inside of a list is called an item.
- Declaring a list is , Items separated by commas are enclosed within brackets [ ].
- All the items in a list do not need to be of the same type (heterogenous).

In [3]:

```
li=[1,0.2,True,'new']              # List is a heterogenous.
li
```

Out[3]:

```
[1, 0.2, True, 'new']
```

Length of list

In [4]:

```
my_list = [10, 20.5, "Hello"]
len(my_list)
```

Out[4]:

```
3
```

Nested list

- within list, one of the item is list

In [5]:

```
nest = [1,[3, 4],3]
nest
```

Out[5]:

```
[1, [3, 4], 3]
```

list Indexing

In [6]:

```
nest[0]
```

Out[6]:

```
1
```

In [7]:

```
nest[1]
```

Out[7]:

```
[3, 4]
```

In [8]:

```
nest[2]
```

Out[8]:

```
3
```

list slicing

In [9]:

```
numbers = [10, 20, 30, 40, 50,60,70,80]
numbers[0:4]
```

Out[9]:

```
[10, 20, 30, 40]
```

list concatenation

In [10]:

```
l1 = ["d", 'b', 'c']
l2 = ['a',6,4.0]
l3 = l2+l2
print(l3)
```

```
['a', 6, 4.0, 'a', 6, 4.0]
```

Lists are Mutable

Mutable : value of elements of a list can be altered in the same object

In [11]:

```
a=[1,2,3]
print(a)

id(a)
```

```
[1, 2, 3]
2358596038592
```

In [12]:

```
a[1] = 20                # value will replaced in the index position
print(a)

id(a)
```

```
[1, 20, 3]
2358596038592
```


List Methods

list.append()

- used to **add only 1 item** at the end of list
- item can be either int, float, str, bool, list

In [13]:

```
lst=[1,2,3,4]
lst
```

Out[13]:

```
[1, 2, 3, 4]
```

In [14]:

```
lst.append(5)
lst
```

Out[14]:

```
[1, 2, 3, 4, 5]
```

In [15]:

```
lst.append([6,7])
lst
```

Out[15]:

```
[1, 2, 3, 4, 5, [6, 7]]
```

list.extend()

- used to add multiple items in a list
- it adds each item individually at the end

In [16]:

```
lst=[1,2,3,4]

lst.extend([5,6])
print(lst)
```

```
[1, 2, 3, 4, 5, 6]
```

difference between .append() & .extend()

In [17]:

```
lst=[1,2,3,4]
lst.append([5,6])
print("list with append :",lst)

lst=[1,2,3,4]
lst.extend([5,6])
print("list with extend :",lst)
```

```
list with append : [1, 2, 3, 4, [5, 6]]
list with extend : [1, 2, 3, 4, 5, 6]
```

list.insert()

- used to insert a item in a specific index position
- list.insert(x, y) will add element item "y" at location indexnumber "x"

In [18]:

```
lst = [11,12,13,14]

lst.insert(3, 10)
print(lst)
```

```
[11, 12, 13, 10, 14]
```

list.remove()

- used to remove an item based on value
- if value repeats, it removes the first one only

In [19]:

```
numbers=[10,20,30,40,10]
numbers.remove(10)
numbers
```

Out[19]:

```
[20, 30, 40, 10]
```

list.pop()

- used to remove an item based on index number
- by default, it selects the index_number = -1

In [20]:

```
numbers=[10,20,30,40]
numbers.pop(3)
numbers
```

Out[20]:

```
[10, 20, 30]
```

list.clear()

- used to clear all items in a list and returns empty list

In [21]:

```
a=[1,2,3,4]
a.clear()
print(a)
```

```
[]
```

del

- it will remove the entire object

In [22]:

```
a=[1,2,3,4]
del a
print(a)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[22], line 3
      1 a=[1,2,3,4]
      2 del a
----> 3 print(a)

NameError: name 'a' is not defined
```

list.count() --> frequency of value in a list

In [23]:

```
l=[1, 2, 3, 1, 1, 0, 3, 4, 2, 5]
l.count(1)
```

Out[23]:

```
3
```

list.reverse()

- used to reverse all items in list

In [24]:

```
l=[1,4.9,[4,6]]
l.reverse()
l
```

Out[24]:

```
[[4, 6], 4.9, 1]
```

list.sort()

- used to sort items in a list either in ascending order or descending order
- by default, its sort in ascending order
- **Note** Sort is applicable for either only alphabet or only numeric values

In [25]:

```
lst = [1, 20, 5, 5, 4.2]
lst.sort()                      # ascending order
lst
```

Out[25]:

```
[1, 4.2, 5, 5, 20]
```

In [26]:

```
lst = [1, 20, 5, 5, 4.2]
lst.sort(reverse=True)          # descending order
lst
```

Out[26]:

```
[20, 5, 5, 4.2, 1]
```

In [27]:

```
lst = [1, 20, 'b', 5, 'a']
print(lst.sort())
```

```
---------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[27], line 2
      1 lst = [1, 20, 'b', 5, 'a']
----> 2 print(lst.sort())

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [28]:

```
lst = [1, 20, 5, 4.2]
sorted(lst)
```

Out[28]:

```
[1, 4.2, 5, 20]
```

Difference between list.sort() & sorted(list)

In [29]:

```
l=[10,1,2,40,6]
l.sort()
print(l)
```

```
[1, 2, 6, 10, 40]
```

In [30]:

```
l=[10,1,2,40,6]
sorted(l)
```

Out[30]:

```
[1, 2, 6, 10, 40]
```

list.copy()

- used to create a new list object with copy of items

In [31]:

```
a=[1,2,3,4]
b=a.copy()
b.append(5)
print(a)
print(b)
```

```
[1, 2, 3, 4]
[1, 2, 3, 4, 5]
```

Difference between shallow copy vs deep copy

In [32]:

```
#shallow copy -- it indicates to different object
a=[1,2,3,4]
b=a.copy()
b.append(5)
print(b)
print(a)
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4]
```

In [33]:

```
#deep copy -- it indicates the same object
a=[1,2,3,4]
b=a
b.append(5)
print(b)
print(a)
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```