In [ ]:
```
'''
Date : 27 dec 2025
list is a data type
it is used to make the homogeneous and the hetrogeneus list
list can be represented by []
Declaration is
obj = [element1, element2 , .....]

elements internally represented by index number
we are not making an array , but is like an array
with index beginning from 0 , 1 and so on.


'''
```

In [1]:
```
l = [1,2,3]
print(l)
```

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

In [2]:
```
#List is mutable datatype. Mutable means can be changed.
#You can also create a hetrogeneous list like :

l = [1,"Ravi",True]
print(l)
```

```
[1, 'Ravi', True]
```

In [3]:
```
#we can also know the type by
type(l)
```

Out[3]: list

In [4]:
```
'''Suppose that there is a list l
by index access

'''
print(l[0])
print(l[1])
print(l[2])
```

```
1
Ravi
True
```

In [ ]:
```
# Since it is mutable we do some change and see the result
l[1]='Soni'
print(l)
```

```
[1, 'Soni', True]
```

In [6]:
```
# a is a list and b is an another list , we can do the concatenation

a=[1,2,3]
b=[4,5,6]
c=a+b
print(c) # is used for the concatenation of lists
```

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

In [7]:
```
#if you want to repeat the list , want to print two times
p=[1,2,3]
print("Repeating it 2 times",(p*2))
```

```
Repeating it 2 times [1, 2, 3, 1, 2, 3]
```

In [8]:
```
# Suppose l is a list
# len is a function used to give the length
# python is procedural as well as the object oriented , so
# will support both function (procedural concept) and method(oops concept)
# methods use the . (dot operator)
#functions do not depend on the object and is independently applicable anywhere
l=[1,2,3,4,5]
print("length",len(l))
print("sum :",sum(l))
print("min :",min(l))
print("max :",max(l))
print("sorted :",sorted(l))
```

```
length 5
sum : 15
min : 1
max : 5
sorted : [1, 2, 3, 4, 5]
```

In [10]:
```
'''There can be also be use of nesting

'''
l=[1,2,[3,4],5]
print(l[2])

# to retrieve the 2nd index 0 and 1 index
print(l[2][0], l[2][1])
```

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

In [14]:
```
#Slicing means to retrive the important/desired data of the list.

# There is some process of slicing the list in the python
# Syntax: object[start_index : end_index+1 : step(default is 1 / incr./ dec(-ve number))]
# end index is always 1 less than
l = [1,2,3,4,5,6,7,8,9,10]
print("from index 0 to 3",l[0:4])

print("from index 0 to 6 , step value 2",l[0:7:2])

print("This is the complete list ",l[::])

print("other way of complete list",l[0::])

print("from index 2 till complete list",l[2::])

print("from index 0 (default) to complete list",l[:6:])
```

```
from index 0 to 3 [1, 2, 3, 4]
from index 0 to 6 , step value 2 [1, 3, 5, 7]
This is the complete list  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
other way of complete list [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
from index 2 till complete list [3, 4, 5, 6, 7, 8, 9, 10]
from index 0 (default) to complete list [1, 2, 3, 4, 5, 6]
```

In [18]:
```
#C/C++ / Java does not support the -ve index

l=[1,2,3,4,5]
print("last value is :",l[-1])

print("-ve index -1 to -3 , step value : -1 :",l[-1:-4:-1])
```

```
last value is : 5
-ve index -1 to -3 , step value : -1 : [5, 4, 3]
```

In [19]:
```
# Some methods
l=[] # this is an empty list

# To insert the values in the list (use l.append(value))

l.append(10)
l.append(20)
l.append(30)

print(l)
```

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

In [20]:
```
# Count in list
l=[1,2,3,2,3]
print("3 occurred how many times ? :",l.count(3))
```

```
3 occurred how many times ? : 2
```

In [21]:
```
# to know the 1st occurance of the element
print("3 occured 1st at ",l.index(3))
```

```
3 occured 1st at  2
```

In [22]:
```
#append always works at the last index
# insert will add value at an index eg l.insert(index_position , value)
l.insert(2,50)
print("Now the list becomes :",l)
```

```
Now the list becomes : [1, 2, 50, 3, 2, 3]
```

In [24]:
```
#extends works the same way like the concatenate work
# like a = a+b
# there is a differecnce : buffer a+b then remove value of a and then move the value of (a+b)

a=[1,2]
b=[3,4]
a.extend(b)
print("new list a becomes :",a)
```

```
new list a becomes : [1, 2, 3, 4]
```

In [29]:
```
print("list is ",l)

l.sort()
print("Sorted array is ",l )

l.sort(reverse=True)
print(" Descending sorted array is :",l)

l=[1,2,3]
l.reverse()
print("The reversed array is ",l)
```

```
list is  [50, 3, 3, 2, 2, 1]
Sorted array is  [1, 2, 2, 3, 3, 50]
 Descending sorted array is : [50, 3, 3, 2, 2, 1]
The reversed array is  [3, 2, 1]
```

In [ ]:
```
# Removing the value from the list is l.remove(value / element)

l.remove(3)
print(l)
```

```
[2, 1]
```

In [32]:
```
# pop , give index to delete the element else the last element will be deleted

l=[1,2,3,4,5]
l.pop(2)
print("Popping the index 2",l)

l.pop()
print("popping the last element",l)

l.clear()
print("removing all the elements",l)
```

```
Popping the index 2 [1, 2, 4, 5]
popping the last element [1, 2, 4]
removing all the elements []
```

Exported with runcell — convert notebooks to HTML or PDF anytime at runcell.dev.