List:

A list is a collection that contains elements of different/same data type

→ in list multiple values are assigned to single variable name

→ we create a list by using "[]" square brackets notation (nothing but symbol)

syntax:

List-Name = [V_1, V_2 . . . . V_n]

ex: a = [10, 20, 30, 40]

list concepts:

→ create

→ ordered

→ mutable

→ Allow dups

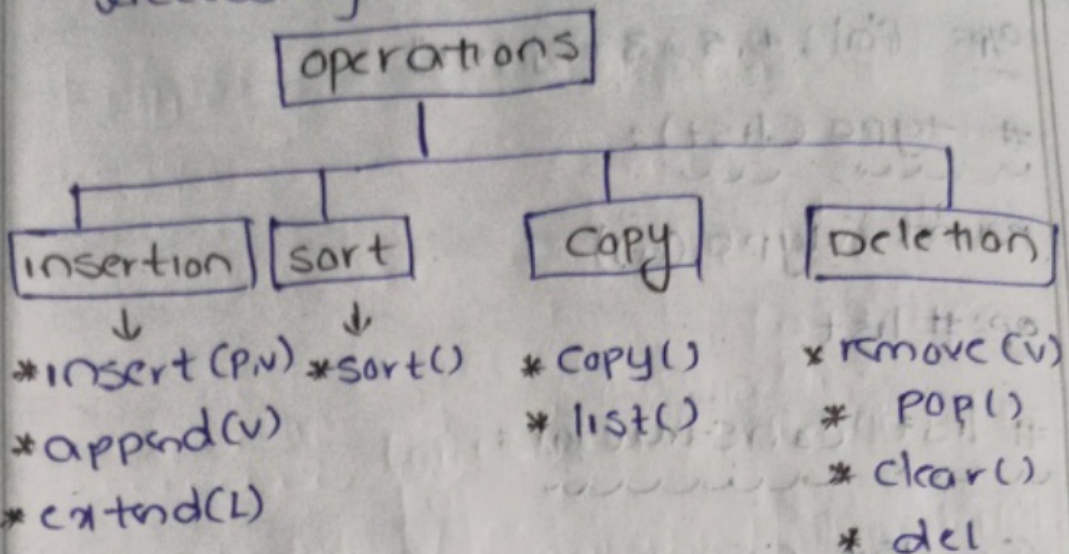
→ length

→ list contained different data type

→ type()

→ list constructor

Accessing list elements:



creation of list:

```
a=[10,20,30,40,50]
print(a)
```

o/p: [10,20,30,40,50] order property

changing element:

```
a[2] = 50
print(a)
```

o/p: [10,20,50,40,50]

length:

```
print(len(a))
```

o/p: 5

contains different data types:

```
b=['a',4.5,3+8j,90]
```

print(b)

op: ['a', 4.5, 3+8j, 90]

type(list):

print(type(a))

op: # list

list constructor:

c = list((1000, 2000, 3000))

print(c)

op: # (1000, 2000, 3000)

accessing list elements:

print(a[2])

op: #50

operations:

[10, 20, 50, 40, 50]
  0 1 2 3 4

insert(3, 60)

a.insert[3, 60]

print(a)

op: # [10, 20, 50, 60, 40, 50]

adding at position without replacing

append (70)

```
a.append(70)
  print(a)
```

op: # [10,20,50,60,40,50,70]

adding end of the list

Extend → $L_1 + L_2$ lombing:

```
a.extend(b)
  print(a)
```

[10,20,50,60,40,50,70,'a',4.5,(3+4j),90]

Sorting Ascending order:

```
a.sort()
  print(a)
```

op: # [10,20,40,50,50,60,70]

```
a.sort(reverse=True)
  print(a)
```

op: [70,60,50,50,40,20,10]

```
x=[10,3.4,7+8j]
  x.sort()
  print(x)
```

} type error

Copy: x = c.copy()

```
print(x)
print(c)
```

op: # [1000, 2000, 3000]

[1000, 2000, 3000]

deletion:

a.remove(50)

print(a)

op: [70, 60, 50, 40, 20, 10]

Pop:

z = [10, 20, 30, 40, 50, 30, 40]

z.pop

print(z)

op: [10, 20, 30, 50, 30, 40]

clear:

clear()

print(z)

op: #[]

del z

print(z)

name '2' is not defined.

Tuple: it is a collection that contains same / different data type

we create tuple by using () paranthesis

properties:

→ ordered

→ immutable

→ Allowed duplicates

```
a [10,20,30,40,50]
print(a)
```

op: # [10,20,30,40,50]

```
print (a[3])
```

op: # 50

```
a[3] = 60
print(a)
```

op: type error:

→ tuple object does not support item assignment

```
b = list[a]
print (b)
```

op: [10,20,30,50,40,50]

```
b[3] = 60
print (b)
```

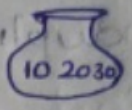
op: [10,20,30,60,40,50]

```
a = tuple(b)
print (a)
```

op: # (10,20,30,60,40,50)

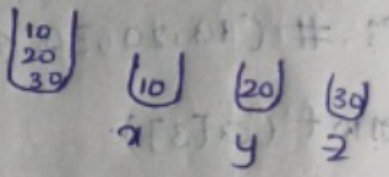
Packed tuple:

a = [10,20,30]



unpacked tuple:

(x,y,z)



x = (10,20,30)

(p,q,r) = x

print(x)

op: # (10,20,30)

print(q) # 20

q=80

print(q) # 80

print(x) # (10,20,30)

12/08

Set: A set is a collection of elements of different type

→ it is unordered

→ It is immutable

→ it doesn't allow dups

operations:

→ Creation

→ Constructor

→ insertion operation

* add()

* update()

→ Deletion operations.

* remove()

* discard()

* pop()

* clear()

* delete()

→ union()

Creation:

Syntax:

Set Name = {$V_1, V_2, V_3 \ldots V_n$}

Ex: a = {10, 20, 80, 40, 50}

Print (a) # {50, 20, 40, 10, 30}

a = {10, 20, 30, 40, 50, 30, 20}

Print (a) # {50, 20, 40, 10, 30}

Print (len (a)) # 5

Print (a[2]) # set object is not subscriptable

```
a[2] :60   # No access

a[a) # set object does not support
item assignment

b = set((10,20,30))

Print (b) # {10,20,30}
```

insertion operation:

i) add()

```
b = set ((10, 20,30))

Print(b)

b.add (40)

Print(b) # {40, 10, 20,30}

b.add (50)

Print (b) # {40,10,50, 20,30}
```

ii) update()

```
a.update (b)

Print (a) # {40,50, 10,20,30}
```

union()

```
a.union(b)

Print(a)
```

Op: # {40150, 10, 20, 30}

Deletion operation:

i, remove()

b.remove(40)

print(b)

op: # {10, 50, 20, 30}

ii, discard()

b.discard(30)

print(b)

op: # {10, 50, 20}

iii, pop()

b.pop()

print(b)

op: # {50, 20}

iv, clear()

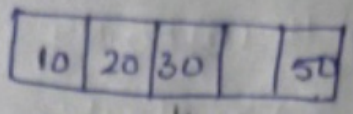
b.clear()

print(b)

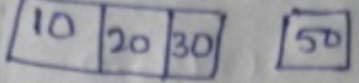
op: # set()

By using remove operation, the data stored in place is removed.

By using discard operation the data stored in place and the place is removed



remove (40)



discard (40)

Dictionary:

A dictionary is a collection of Elements in form of KEY:VALUE Pair

{10, 20, 30}
{10, 20, 30}
{10, 20, 30}

} Single dimentions

Sid	Name
101	aaa
102	bbb
103	ccc
104	aaa

keys

values

Syntax:

Dictionary-Name : { $k_1:v_1, k_2:v_2 \ldots k_n:v_n$ }

Ex: a = {1:10, 2:20, 3:30}

1	10
2	20
3	30

→ It is ordered
→ It doesn't allow dups
→ It allow dups in values but not in keys
→ In dictionary the index start from number 1
→ It is mutable

```
a = {1:10,
     2:20,
     3:30,
    }
print(a) # it is ordered
# {1:10, 2:20, 3:30}
a = {1:10,
     2:20,
     3:30
    }
print(a) # it doesnot allow dups.
# {1:10, 2:20, 3.30}
print(a.key())
# {1,2,3}
print(a.values())
# dict_values(10,20,30)
a[1] = 60
print(a)
# (1: 60, 2:30, 3:30)
print(a[0]) # key error
```

```
a.pop(2) # key
print(a) # {1:60, 3:30, 4:40}
a.popitem()
print(a) # {1:60, 3:30}
a.clear()
print(a) # {}
a.del()
print(a) # name is not defined.
```

Tuple It is a Collection that Contains Same/Different Data Type.

We Create Tuple By Using ( )

Properties

Oredred

Immutable

Allow Duplicates

```
a=(10,20,30,50,40,50)
print(a)                                '''TypeError:
#(10, 20, 30, 40, 50)                      'tuple' object
#(10, 20, 30, 50, 40, 50)                  does not support
print(a[3])#50                             item assignment'''
#a[3]=60
print(a)
```

```
#Extend -> L1+L2 Combine
b.extend(a)
print(b)
#['a', 4.5, (3+8j), 90, 10, 20,
50, 60, 40, 50, 70]
#[10, 20, 50, 60, 40, 50, 70]
#Sorting
#Ascending Order
a.sort()
```

```
#Descending Order
a.sort(reverse=True)
print(a)
#[70, 60, 50, 50, 40, 20,
10]
x=[10,3.4]
x.sort()
print(x)#[3.4, 10]
#copy()
```