

Motivation behind Aggregate types:

$m = 15$

$n = 25$

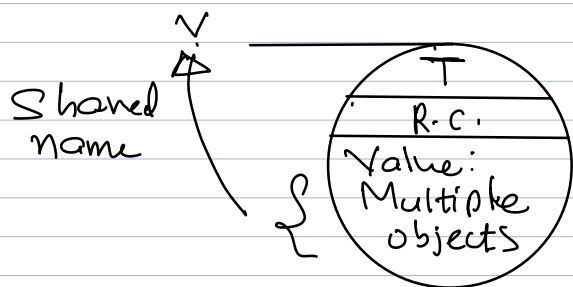
$k = 35$

$p = 45$

$q = 55$

'Set helps us to conceive plurality as a Singularity'

At programming level, we should be able to define **such data types**, whose one object can contain multiple objects of other types. We can dedicate a variable name to that single object (=which is in effect a shared name between all contained objects). And then we should figure out some way to reach to each of the contained object using a single variable name.



Aggregate type:

{ ① One single object should be able to contain multiple objects. } Compulsory

② Single name given to an aggregate object is sufficient to access multiple objects stored in it. 99%
optional 99% follow

List

```
>>> L = [15, 25, 35, 45, 55]
```

```
>>> print(L)
```

```
[15, 25, 35, 45, 55]
```

```
>>> type(L)
```

```
<class 'list'>
```

```
>>> id(L)
```

large positive integer

```
>>> L[0]
```

15

```
>>> L[1]
```

25

```
>>> L[2]
```

35

```
>>> L[3]
```

45

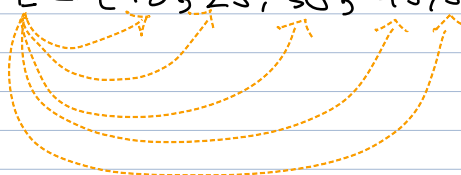
```
>>> L[4]
```

55

```
L = [15, 25, 35, 45, 55]
```

```
>>> L
```

```
>>> print(L)
```



Every object stored inside list object gets a unique roll number assigned to it.

Roll number assignment : starts from zero and proceeds serially.

$L = [15, 25, 35, 45, 55]$
0 1 2 3 4 \longrightarrow index of element

→ Every object in the list object gets a unique index assigned to it.

And name given to the list object and index of element can be used together to access the element individually.

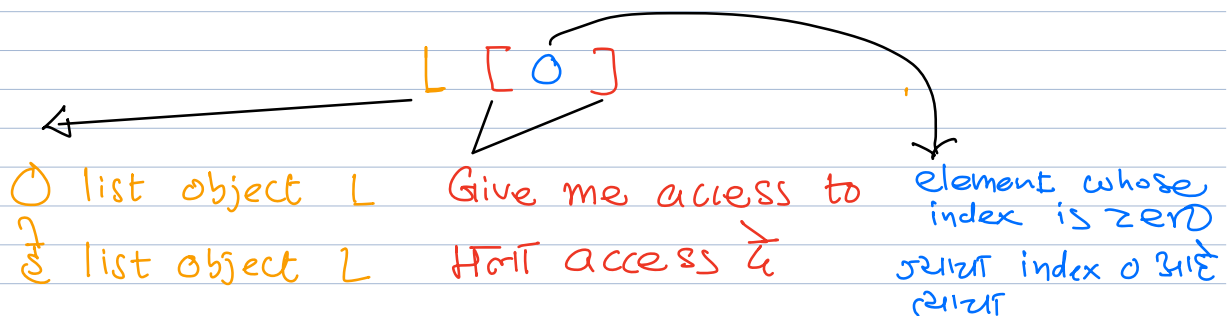
$L[0]$

$L[1]$

$L[2]$

$L[3]$

$L[4]$



$L[0]$ L subscript 0 $L[]$

$L[1]$ L subscript 1

$L[i]$ L subscript i

Let $\{L_n\}$ be a sequence.

$L_0 \ L_1 \ L_2 \ L_3 \ L_4 \longrightarrow$ terms of sequence

$L_n \ L_0 \ L_1 \ L_2 \ L_3 \ L_4$

$\overline{\overline{L}} \text{ script}$
 $0 \longrightarrow$ subscript

$2 \longrightarrow$ super script
 $a \longrightarrow$ script

$a \ b \ c \ d \longrightarrow$ script

$L = [15, 25, 35, 45, 55]$

`print(L0)`

`print(L1)`

`print(L2)`

`print(L3)`

`print(L4)`

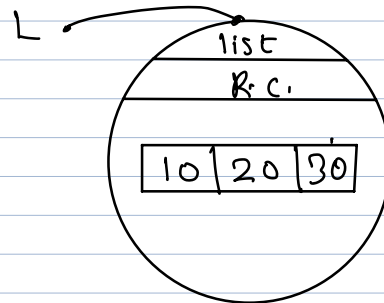
$\forall i (0 \leq i < N)$ `print(i2)`
==

for `i in range(N):`
 `print(i**2)`

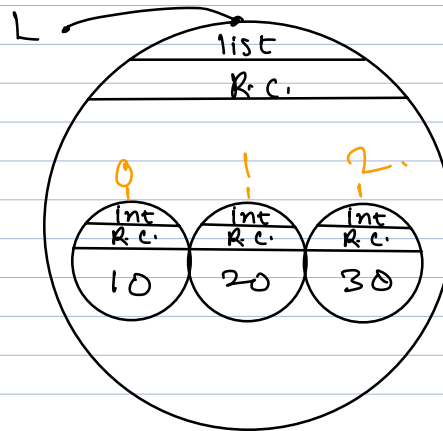
- ① A Haskell Road to logic, math & programming,
- ② Discrete Mathematics with computer.

Scheme - I

$L = [10, 20, 30]$



Scheme - II



Scheme - III

