Session - 4

List in Python.

→ What are Lists?
→ Lists vs arrays
→ Characteristics of a list
→ How to create a list
→ Access items from a list
→ Editing items in a list
→ Deleting items from a list
→ Operations on Lists
→ functions on Lists.
→ List Comprehension
→ zip
→ Disadvantages of python list

```
L = [20, 'Amit', 35.75, [30, 30, 60]]
```

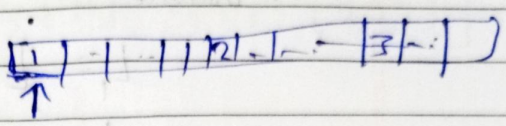
⇒ Array vs list

fixed vs Dynamic Size : List is flexible
Homogenious vs Hetrogenious (∵ Same data type — Multiple data type)
Fast - Slow
low Space — More Space
Convenience

id() ⇒ Print Memory address

→ Storage of Array in List — Continuous

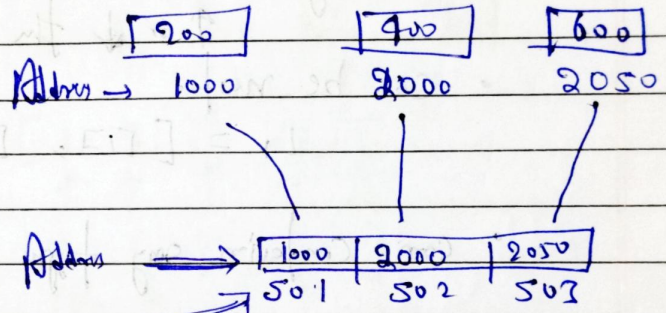
→ int arr[50]



→ List — Non Continuous

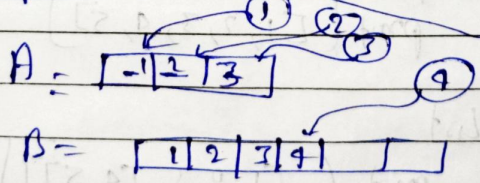
Referential array

List = [200, 400, 600] ← values



Address on 501, Add. 1000 is stored
on 502, Add. 2000 is stored.

→ Dynamic Arrays: • Slow & take more time
• They can store multiple type of data



* Characteristics of a list
  - ordered
    $L_1 = [1, 2, 3]$
    $L_2 = [3, 2, 1]$
    $\therefore L_1 \neq L_2$
  - changeable / mutable
    $L_1[0] = 5$
  - Heterogeneous (multiple data types)
  - Can have duplicates
    $L_1 = [1, 1, 1]$
  - are dynamic
    ↑ or ↓ the size.
  - Can be nested
    $L_1 = [[1], [2]]$
  - Can contain any types of objects in python

* creating a list
  → Empty
    print([])
  → 1D & Homogeneous
    print([1, 2, 3, 4, 5])
  → 2D List
    print([1, 2, 3], [4, 5])
    Heterogeneous

→ 3D list

[[[1,2],[3,4]],[[5,6],[7,8]]]

It's Homogenous

→ Heterogenous List

[1, True, 5, 5.6, 5+6j, 'Hello']

→ list('hello')

⇒ ['h', 'e', 'l', 'l', 'o']

→ fetch items / Accessing items from a list

L = [1, 2, 3, 4, 5]

positive Indexing ⇒ 0 1 2 3 4

Negative Indexing ⇒ -5 -4 -3 -2 -1

→ Slicing

L = [1, 2, 3, 4, 5, 6]

L[0:3] ⇒ 0, 1, 2

L[-1:0] ⇒ [ ]

L[-3:0] ⇒ [4, 5, 0]

L[0::2] ⇒ [1, 3, 5]

L[::-1] ⇒ [6, 5, 4, 3, 2, 1]

Adding items to a list

① Append ② Extend ③ Insert

L = [1, 2, 3, 4,]

Single item → L.append(6), [1, 2, 3, 4, 5, 6]

Multiple items → L.extend([6, 7, 8]), [1, 2, 3, 4, 5, 6, 6, 7, 8]

L = [1, 2, 3, 4, 5]

at a position → L.insert(1, 100)

index, elem

∴ L = [1, 100, 3, 4, 5]

Editing items in List

L = [1, 2, 3, 4, 5]

with index → L[4] = 500 { [1, 2, 3, 4, 500] }

with slicing → L[1:4] = [200, 300, 400]

{ [1, 200, 300, 400, 500] }

Deleting items from a list

• Del • remove • pop • clear

Del → L = [1, 2, 3, 4, 5]

del L { L = ~~[ ]~~ nothing }

L = [1, 2, 3, 4, 5]

del L[-1] (L = [1, 2, 3, 4])

del [1:3] (L = [1, 3])

Remove
delete with value

L = [1, 2, 3, 4, 5]

L.remove(5)

(L = [1, 2, 3, 4])

Pop
at index

L = [1, 2, 3, 4, 5]

L.pop(0) { [2, 3, 4, 5]

L.pop() ([2, 3, 4])

clear
make empty

L = [1, 2, 3, 4, 5]

L.clear() { L = [] }

Operations on List

- arithmetic
- membership
- loop

L1 = [1, 2, 3, 4]

L2 = [5, 6, 7, 8]

L1 + L2

⇒ [1, 2, 3, 4, 5, 6, 7, 8]

L1 = [1, 2, 3, 4, 5]

L2 = [1, 2, 3, 4, [5, 6]]

L1 * 3

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

5 in L1 ⇒ True

5 in L2 ⇒ False

[5, 6] in L2 ⇒ True

#1 List functions.

→ Common functions.

- len / min / max / sorted

L = [3, 1, 5, 7, 0]

len(L) = 5

only works on numbers
Not works on string

max(L) = 7

min(L) = 0

sorted(L, reverse) ⇒ [7, 5, 3, 1, 0]

- count

frequency

L = [1, 2, 1, 3, 4, 1, 5]

L.count(1) = 3

- index

L.index(2) = 1

L.index(1) = 0

- reverse

L.reverse() ⇒ original list changed

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

- sort vs sorted

sort ⇒ permanent change

sorted ⇒ original will not change

L.sort()

sorted(L)

- copy() ⇒ do shallow copy

```
L = [2,1,5,7,0]
print(L)                    [2,1,5,7,0]
print(id(L))                xyz101
L1 = L.copy()
print(L1)                   [2,1,5,7,0]
print(id(L1))               pyo234
```

#) List Comprehension :-

Short cut way to declear list

```
new list = [expression for item in iterable if
                                condition == True]
```

Adv
- Mem Time & Space efficient than loops
- Requires fewer lines of code
- Transforms iterative statement into a formula

2 ways to traverse a list

- Itemwise
- Indexwise

Zip :-

```
L1 = [1,2,3,4]
L2 = [-1,-2,-3,-4]
list(zip(L1,L2)) ⇒ [(1,-1),(2,-2),(3,-3),(4,-4)]
```

```
[i+j for i,j in zip(L1,L2)] ⇒ [0,0,0,0]
```

* In python Everything is object.

* Python lists are so flexible i.e. it can store any type of object.

```
L = [1, 2, print, type, input]
print(L)
```

[1, 2, <built in fn 'print'>, <class 'type'>, ---]

* Disadvantage of List
  - Slow
  - Eats up more memory
  - Risky Usage
    - Eg. a = [1, 2, 3]

Now b will point to same memory location as a

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

∴ use copy() function