19.07-25

01/List #

List in Py (ARRAY)

— While array does not have a built-in 'array' data type Python provides the alternatives — List.

In Python, 'List' is a built-in dynamic sized array. (automatically grows & shrinks).
Create by —([ ]) Square brackets.
List is a collection of items, can store all types of items in a list (including another list).

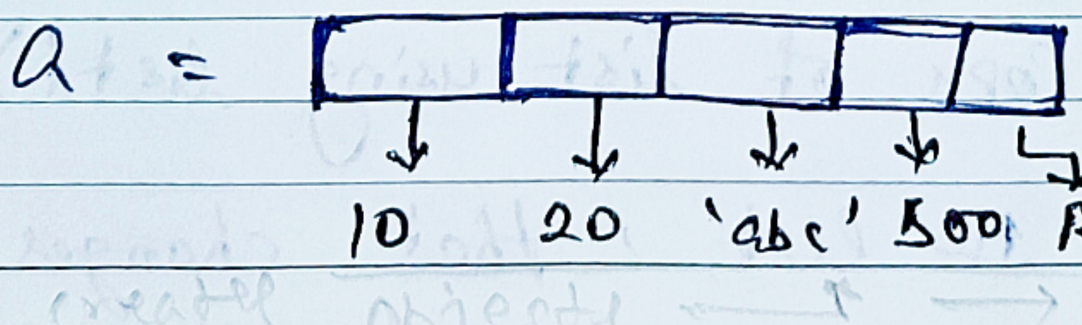
- List contain Duplicate items
- List are Mutable. - means — we can modify, replace, delete items.
- List are Ordered.
- Accessing items in List, by position (index). start from 0.

Memory reprentation —

In, Python, list stores references to objects, means each elements in the list points to a memory location where the object is stored.

List Store references, Not values.

Py. internally creates separate objects for values, then stores their memory addresses inside List. — is itself a container with (addres) references of the actual values.



⟹ list = [ ] Using Square Brackets

- List can store integer, float, string (charcter) type mixed elements at once.

list = ['a', 20, '40.7, True]

→ Using list() Constructor

In Python, 'list()' constructor is a built-in function — which construct list object.

Use list() constructor to create an empty list OR

Convert an iterable (like, dict, tuple, string in to list.

like — a = list()
b = list((1, 2, 3))
c = list('Good')

Also create nested list —

a = list([list(range(3)) for _ in range (3)])
Print(a)

O/P — [[0, 1, 2], [0, 1, 2], [0, 1, 2]]

Shallow Copy of list using list()

In, IP list, that changes doesn't reflac

But in Nested list, it's reflact the change.

Shallow copy → a = [1,2,3]
a1 = list(a)
a1[0] = 5
→ Not change same will print

But, a = [[1,2,3], [4,5,6]]
a1 = list[a] ← S.C.
a1[0][0] = 500
→ Change & return [[500,2,3], [4,5,6]]

Shallow copy → Copy = list(original)

Why? → Because →

a & a[1] - are different Outer lists.

a = [ ]

b & b[1] - are (same) different outer list. But both points the inner loop.

a = [[ ], [ ]]

Types

- On the basics of SIZE
  - Fixed (Static) size
  - Dynamic size
- on the basics of Dimensions
  - One-dimensional
  - Multi-dim
    - Two-dim
    - three-dim
    - N-dim

Accessing List Elements Using - Indexing —

```
a = [10, 20, 30]
Print (a[10])   ← Print 10 (first element)
print (a[-1])   ← 30 (last element)
```

Start from [0, 1, 2, 3, 4]
0 → [-5, -4, -3, -2, -1] ← start from -1

— Using 'title()' method →

```
b = ['trak', 'ari', 'muk', 'success']
Print (b[0])  → O/P - trak
```

'title()' method →

```
Print (b[0].title())  → O/P - Trak
```

'title()' → return a version of string where each word is titlecased.
Word start with uppercase & remain are same as lowercase.

" INDEXING POSITION START FROM 0 NOT 1 "

Modifying element in LIST

```
a = ['ari', 'bycicle', 'arix']
a[0] = 'aribra'   ← change 'ari'
Print(a)
```

Adding Element to List -

- append () : Adds AN ELEMENT at the END of list.
- extand () : Adds MULTIPLE ELEMENT to the END of list.
- insert () : Adds an element at a SPECIFIC Position.

" a.append ('Anitra Mukherjee') "

o/p - [arita, bycicle, arix, Anitra Mukherjee]

" a.extand (['AI', 'ML', 'DL'])

o/p → [anitra, bycicle, arix, Anitra Mukherjee, AI, ML, DL]

" a.insert (0, "Data Scientist ")

o/p - [Data Scientist, anitra, bycycle, ..., DL]

Removing Element from a list -

- Remove () : Remove the first occurence of the specified element from the list.

- PoP () : Removes element at a specific index if assigned OR if not assigned by default it remove the last element.

- del - statement () : Delete an element at a specific index.

a = [100, 40, 90, 20, 1000]

a.remove (40)

a.PoP (2) ← o/p → remove the 2 index value

a.PoP () ← o/p → last element

del a[0] ← o/p :) Delete first element

Organizing a List

- sort() - Sorting a List Permanently.
- sorted() - Sorting a list Temporarily.

```
a = ['xyz', '20', 'Ari', 'Muk']
a.sort()
# Sort the list Permanent
sorted(a)
# sort temporary
```

- Printing a List in Reverse

```
a.reverse()
# reverse the
```

- Finding the length of List → Find quickly using len() function

```
len(a)
```

- Looping through an entire list

Go through the list using 'for loop' or other methods. Looping through lists is helpful when we need to perform an operation on each item or access specific items based on certain conditions.

```
for i in a:
    Print(i)
```

25.07.25

→ Making Numerical list

Using 'range()' function —

To print a series of numbers.

```
for value in range(1,5):
    print(value)
```

n, (n-1)
range(1,5)
n = 5
n-1 = 4
Print O/P ⇒ 1,2,3,4

Using range() make a list of numbers —

```
num = list(range(1,6))
print(num)
```

n = 6 , n-1 = 5
(1,2,3,4,5)

Can convert the result of 'range()' directly into list using 'list()' function.

Find even, odd =

```
even_num = list(range(2,15,2))
odd_num = list(range(3,15,3))
```

Slicing a List

```
a = ['ari', 'mun', 'book', 'AI']
a[2:]
```

Looping through a Slice

```
for i in a[:2]:
    print(i)
```

Copying a list

```
ab = a[:]
```