List_ List Methods

* Derived data types in Python are [8]
  - list ✓
  - tuple ✓
  - set
  - frozenset
  - dictionary
  - bytes
  - bytearray
  - range

[operator precedence youtube]

* List :- Anything kept inside the square bracket is a list

Ex:-
```
lst = [10, 20, 30, 40]
print(lst, type(lst))
```
If we know the element in advance.

output : [10, 20, 30, 40] <class 'list'>

* Taking user input

```
lst1 = input('Enter an input: ')
print(lst1, type(lst1))
```

output: Enter an input: [20, 30, 40]
<class 'str'>

Conversion from str to list (or) tupl

* Simply Add eval

Ex:-

```
lst1 = eval(input("Enter an iput :
print(lst1, type(lst1))
```

* Using list [Iterable]

Creating a list from string

* print(list('Python'))

output : ['P', 'y', 't', 'h', 'o', 'n

tuple => * print(list((10, 20, 30)))

output : [10, 20, 30]

set => * print(list({1:10, 2:20, 3:30}))

output : [1, 2, 3]

Properties of list :-

* list is a sequence
* It is an ordered collection of the elements
  [The elements are indexed]
* list also supports +ve & -ve indexing
  - ⇒ +ve indexing starts from 0 [LHS]
  - ⇒ -ve " " " " -1 [RHS]
* list supports indexing & slicing
* It is mutable - modify (Assignment is allowed, will keep the changes in the existing object)
* It allows the duplicate elements
* Basic operation - Indexing, Slicing, Concatenation, Repetation, Membership, Identity operator
* It can hold any data as it's elements

```
-4  -3  -2  -1   >> Negative indexing
[10, 20, 30, 40]
 0   1   2   3   >> Positive indexing
```

* Indexing - accessing one element at a time

lst = [10, 20, 30, 10+20j, 'Python', None, True,
[10,20,30], (100,200,300), {100,200,300},
{1:200, 2:200, 3:300}]

Accessing the 1st element

print (list[0])
print (lst[-len(lst)])

* print (list[5])

output: True

* Concatenation & repetation

=> Concatenation - both operands should be like list - "+"

print (lst + [111, 222, 333, True])

=> Repetation - one operand should be list and other should be integer - "*", Creates a new list object

print (lst * 3)

* print ('Python' in lst)

output: True

* lst1 = lst
lst2 = lst[::]
print (lst is lst1)
print (lst2 is lst)
print (id (lst))
print (id (lst1))
print (id (lst2))

output: True
False

* Nested list:-

lst1 = [10, 20, 30, [40, 50, 60], [70, 80, [90, [200, 300]]]]

* print (dir (lst))

append, clear, copy, count, extend, index, insert, pop, remove, reverse, sort

* Insert element inside a list - append, extend, insert

```
lst = [10,20,30]
lst.append(111)
print(lst)
```

output: [10,20,30,111]

* Adding a list at the end

```
lst.append([1,2,3])
print(lst)
```

output : [10, 20, 30, 111, [1,2,3]]

* extend - similar to concatenation but it don't create a new list object rather it updates into the existing one.

Ex:-

```
lst1 = [False, {1:100, 2:200}, None, 40, 200]
# Extending lst1 into lst
lst.extend(lst1)
print(lst)
print(lst1)
```

* Insert :- [index, value]

Ex:- lst.insert (5, False)
print (lst)

output : [10, 20, 30, 111, [1, 2, 3], False]

* Removing an element :- 'pop', 'remove'

* print (lst.pop( )) ⇒ It will remove by default last element

* Giving index & removing an element

print (lst.pop(5))

* Removing an element by value

lst.remove ('Python')
print (lst)

* Index, Count, Clear :-

* Count :- To count the number of times an element is repeating

Ex:- lst1 = [10, 20, 10, 10, 30, 40, 10]

print(lst1.count(10)) ↵

output:-4

* Index :- Similar to the index in string

* We do not have rindex, find & rfind

* returns ValueError if element is not found

Ex:- lst1 = [10, 20, 30, 100, 20, 30, 20, 40, 20, 40, 50, 60, 20, 200, 900, 20]

lst1.index(20, 7, 9) ↵

[finding value] range from [7-9]

output : 8

clear :- clear method empties the list & keeps the empty list

Ex:- lst.clear().
print(lst) ↵

output : [ ]

list is empty

1hr 20min

* Reverse the list :-

lst = [10, 20, 30, 40]
lst.reverse()
print(lst) ↵

output : 40, 30, 20, 10

* Sorting in ascending order is default

lst = [10, 20, 10, 30, 10, 40, 50]
lst.sort()
print(lst) ↵

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

* sorting in descending order :

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

* Aliasing , shallow copy, deep copy

Aliasing :- Giving a diff name to an existing object

```
lst = [10, 20, 30]
lst1 = lst
print(lst is lst1)
```

output : True

* 4 × 4

[[0, 0, 0, 0], [0, 1, 2, 3], [0, 2, 4, 6], [0, 3, 6, 9]]

```
[[i * j for j in a range (4)] for i in a range (4)]
```

Deep Copy :-

* from copy import deepcopy

```
lst3 = [10, 20, 30, [40, 50, 60], 70, 80]

lst3 = deepcopy(lst4)
print ( lst3 is lst4)
lst3 [3] [-1] = 6666
print(lst3)
```

output : False

[10, 20, 30, [40, 50, 60, 6666], 70, 80]

List comprehension :-

* To create a list of all even numbers from 20 to 50

```
out_lst = [ ]
for i in range (20, 51) :
    if (i % 2 == 0) :
        out_lst.append (i)
print (out_lst)
```

```
lst_out = [ ]
from i in
    range(20, 30):
  if i % 2 == 0 :
lst_out.append (i)
print (out_lst)
```

output : 20, 22, 24, 26, 28, 30, 32, 34 ...

* [i for i in range (20, 51) if i % 2 == 0]

output :- 20, 22, 24, 26, 28, ...

* [[i * j for j in range (4)] for i in range (4)]

output: [[0,0,0,0], [0,1,2,3], [0,2,4,6], [0,3,6,9]]

* [[[i * j for k in range (6)] for j in range (4)] for i in range (3)]

output :- [0,0,0,0,0,0], [0,0,0,0,0,6],
[0,0,0,0,0,0], [0,0,0,0,0,0]
---- etc

Tuple :-

* Is similar to the list
* All the element should be kept inside "small bracket" & separated by comma.

Ex:- tp1 = (10, 20, 30, 40)
print (tp1, type (tp1))

Eval function

tp1 = eval(input("Enter the value :"))
print (tp1, type (tp1))

output: 10, 20, 30
<class 'tuple'>

Properties of Tuple :-

* Tuple is a seq
* It is an ordered collection of the elements ⇒ The elements are indexed
* Tuple also supports +ve & -ve indexing
  +ve indexing starts from 0 from LHS &
  -ve indexing starts from -1 from RHS

* Tuple support indexing & slicing both
* It is immutable
* It allows the duplicate elements
* Basic operations - Indexing, slicing, Concatenation, Repetation, Membership, Identity Operator
* It can hold any data as it's elements

* Methods in tuple :-
  * count
  * index

* Tuple packing & unpacking :-

Tuple packing :- packing (or) assigning multiple values to a single variable

```
a = 100
b = 200
c = 300
d = 400
tp1 = a, b, c, d
print(tp1)
```

Unpacking :- fetching the individual value from a tuple

tpl = (111, 222, 333, 444, 555, 666)
a b c d e f

a, b, c, d, e, f = tpl

print(a)
print(b)
print(c)
print(d)
print(e)
print(f)

* In unpacking no. of variables on LHS & no. of elements should match