Full stack web development using python

List

Saurabh Shukla (MySirG)

Agenda

1. What is a list?
2. How to create list object?
3. How to access list elements?
4. Concept of negative indexing
5. Accessing list elements via for loop
6. How to delete an element from a list?
7. How to edit an element of the list?
8. How to add more elements in the list?
9. Packing and Unpacking
10. built-in methods

11. list() method
12. comparison operator on list
13. concatenation operator
14. Repetition operator
15. list of lists
16. list object methods
17. list comprehension
18. user input
19. Mutability and hashability

list

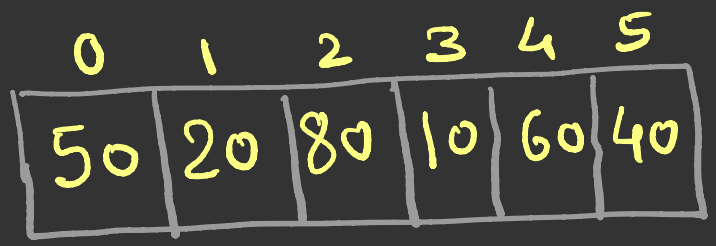
- list is a class
- list is an iterable sequence
- list is mutable
- list is growable
- list can store heterogeneous data
- list elements are indexed

How to create list object?

```
l1 = [10, 20, 30]        # Square brackets are used to denote a list

l2 = []                  # empty list object

l3 = [50, 3.5, "abc"]    # Heterogeneous elements
```

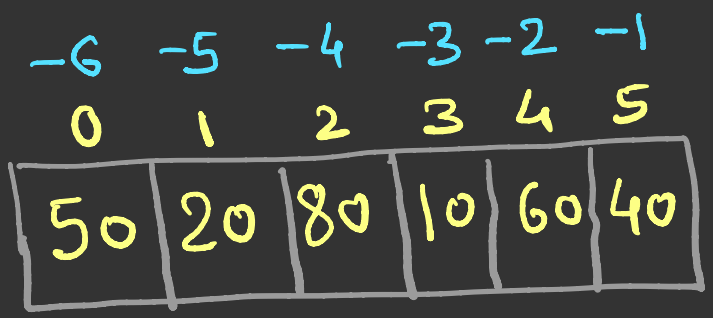
How to access list elements?

```
l1 = [50, 20, 80, 10, 60, 40]
print(l1)              # [50, 20, 80, 10, 60, 40]

print (l1[0])          # 50
print (l1[1], l1[2])   # 20 80
```



Concept of negative Indexing

l1 = [50, 20, 80, 10, 60, 40]

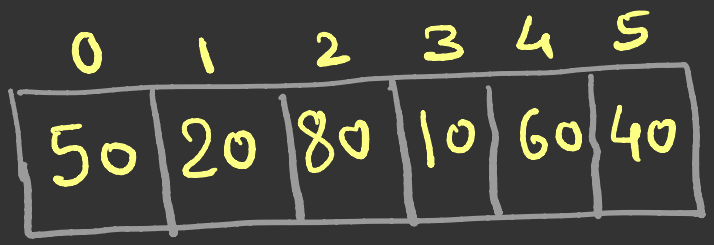


print (l1[-1]) # 40

print (l1[-2]) # 60

Accessing list elements via for loop

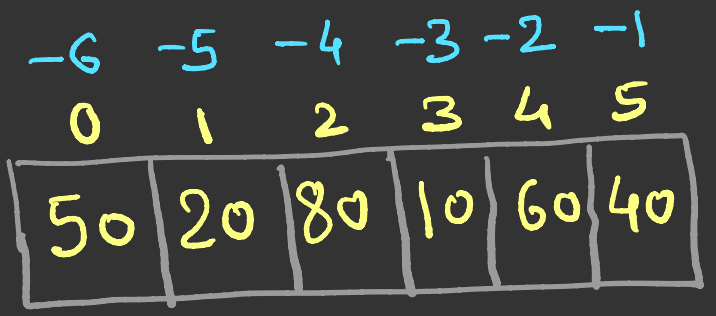
```
l1 = [50, 20, 80, 10, 60, 40]

for x in l1:
    print(x, end=' ')
```



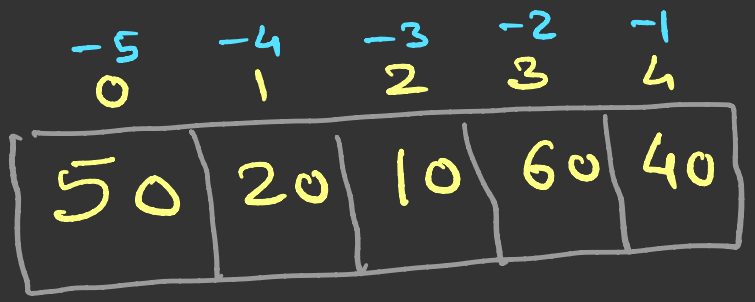

```
i=0
while i<=5:
    print(l1[i],end=' ')
    i+=1
```

How to delete an element from the list?

```
l1 = [50, 20, 80, 10, 60, 40]
```

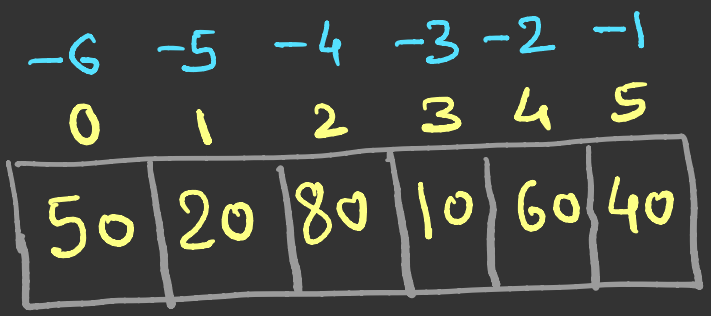



```
del l1[2]
```

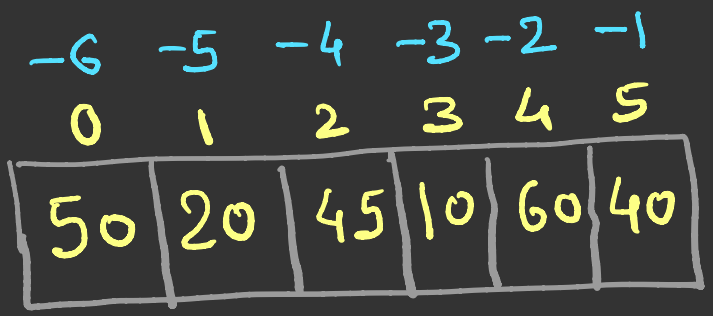


How to edit an element of the list?

```
l1 = [50, 20, 80, 10, 60, 40]
```

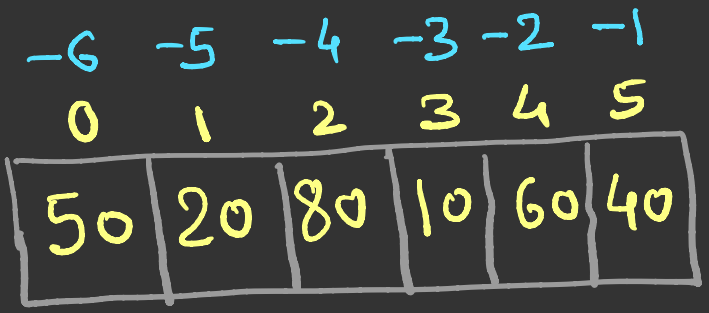



```
l1[2] = 45
```



How to add more elements in the list?

```
l1 = [50, 20, 80, 10, 60, 40]
```




```
l1[6] = 70   # index error
```

listObject [invalidIndex] is an error

```
l1[5] = 70
l1[4] = 90
```

Not adding more values in the list but only updating values at index 5 and 4

There are two standard ways to add elements in the list:

① append()
② insert()

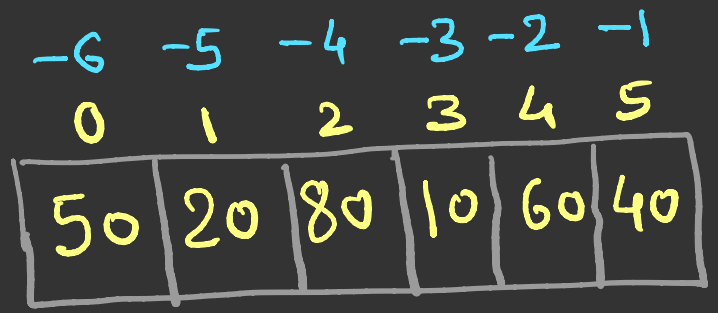
} attributes of list class

- Class is a group of variables and functions
- These variables and function are called attributes

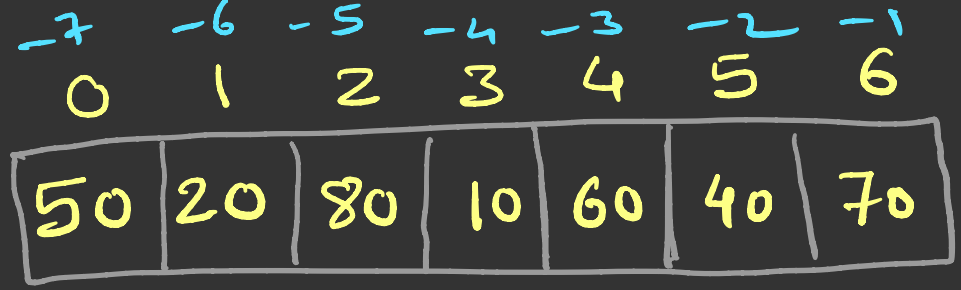
listObject.append(value)
listObject.insert(index, value)

append method adds value at the end of the list, that is new element will become the last element of the list

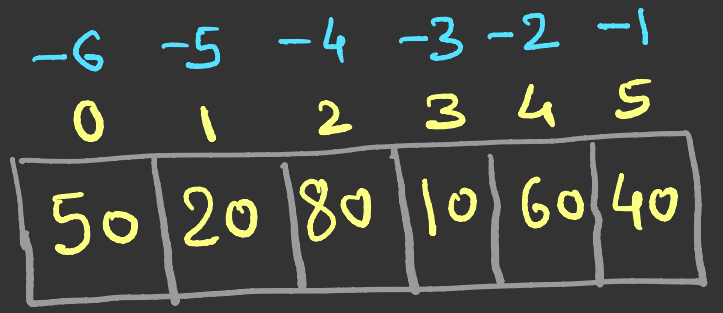
l1 = [50, 20, 80, 10, 60, 40]



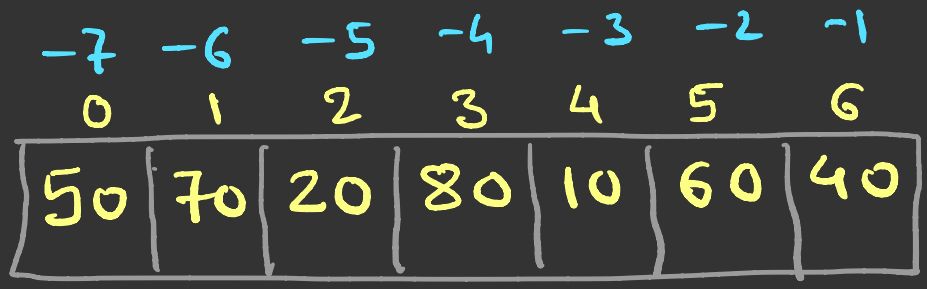
l1.append(70)



l1 = [50, 20, 80, 10, 60, 40]



l1.insert(-5, 70)



l1.insert(index, value)

if index > lastindex then value will store at lastindex+1 only(append).

Packing and Unpacking

```
l1 = [20, 50, 30]

a, b, c = l1          # unpacking
```

number of variables in the left hand side must be same as the number of elements in the list.

```
a = 5
b = 6
c = 10

l2 = [a, b, c]        # packing
```

Built in methods

len() → returns length of specified iterable

min() → returns min value element

max() → returns max value element

sum() → returns sum of elements

sorted() → returns a sorted list of elements.

list() method

```
l1 = list()          # l1 = []
l1 = list(10)        # l1 = [10]  Error
l1 = list(10,20,30)
l1 = list("MySirG")
l1 = list(range(5))
l1 = list([10,20])
```

list() method can take at most one argument.

one argument ⟶ iterable

Comparison Operator on list

```
l1 = [1, 2, 3]
l2 = [2, 3, 1]
l3 = [1, 2, 3, 4, 5]
l4 = [1, 2, 3]
```

`l1 == l2` False

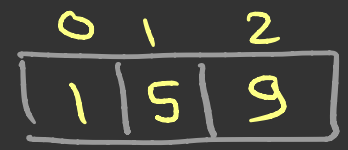
`l1 == l3` False

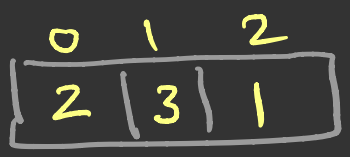
`l1 == l4` True

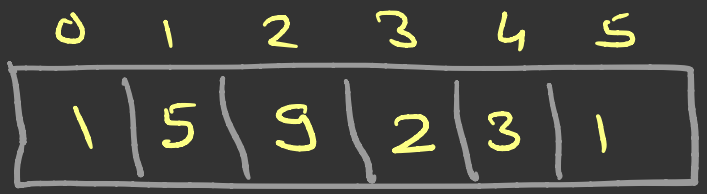
`l1 > l2` False

Concatenation Operator

```
l1 = [1, 5, 9]
l2 = [2, 3, 1]
l3 = l1 + l2
```



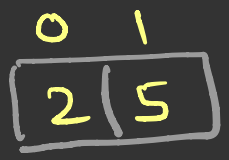




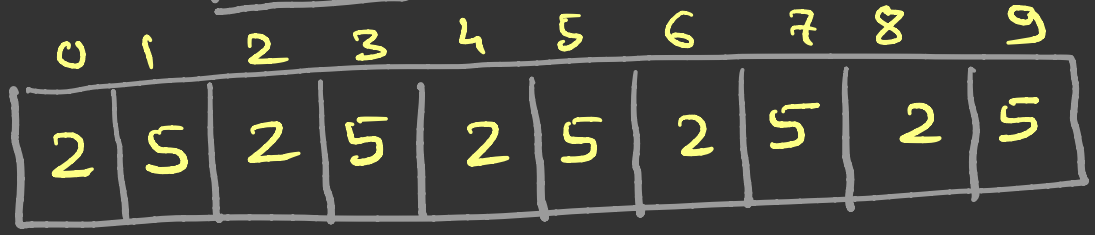

```
l1 += l2  ⟶  l1 = l1 + l2
```

Repetition Operator

l1 = [2,5]

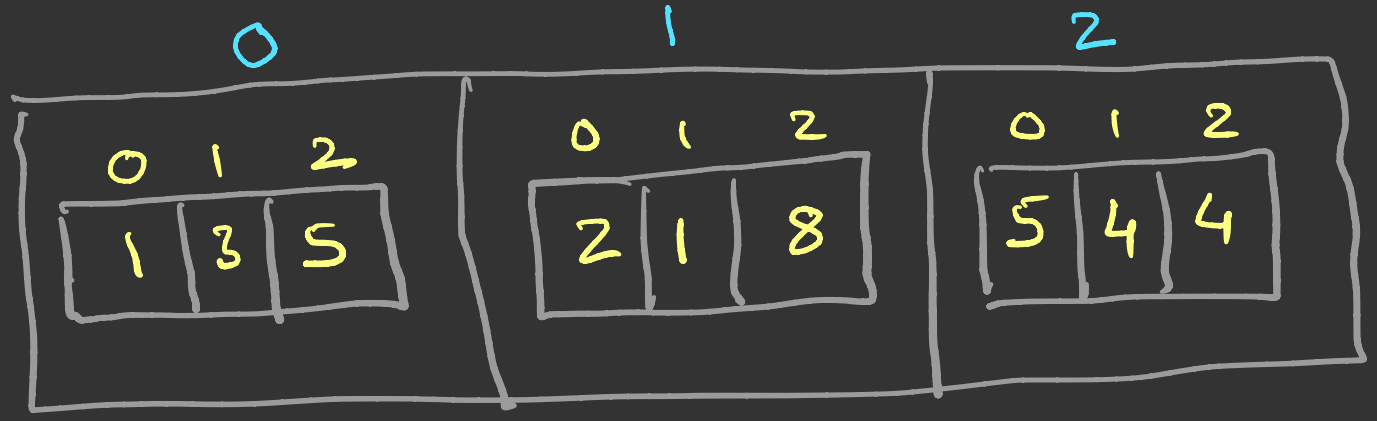


l1 * 5



list of lists

```
l1 = [ [1,3,5], [2,1,8], [5,4,4] ]
```



`l1[0]` → `[1, 3, 5]`

`l1[0][2]` → `5`

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 1 & 8 \\ 5 & 4 & 4 \end{bmatrix}$$

list object methods

append()
insert()
remove() → Remove first occurrence of specified element. It gives error if the element is not present in the list

pop() → Remove and return last element optionally you can pass index number to pop specific element.

clear() → Remove all the elements

reverse() → Reverse the list

sort() → Sort list elements

index() → Returns index of first occurrence of specified element

count() → Count number of occurrence of specified element

list Comprehension

[expression for variable in iterable]

User input

Mutability and hashability

- Mutable objects are changable
- immutable objects are not changable
- All immutable objects are hashable, but not all hashable objects are immutable.
- Hashable is a feature of Python objects that tells if the object has a hash value or not. If it has a hash value that does not change during its entire lifetime.

	Hashable	Mutable
int	Yes	NO
float	Yes	NO
Complex	Yes	NO
bool	Yes	NO
str	Yes	NO
range	Yes	NO
list	NO	Yes
tuple	Yes	NO
set	NO	Yes
dict	NO	Yes