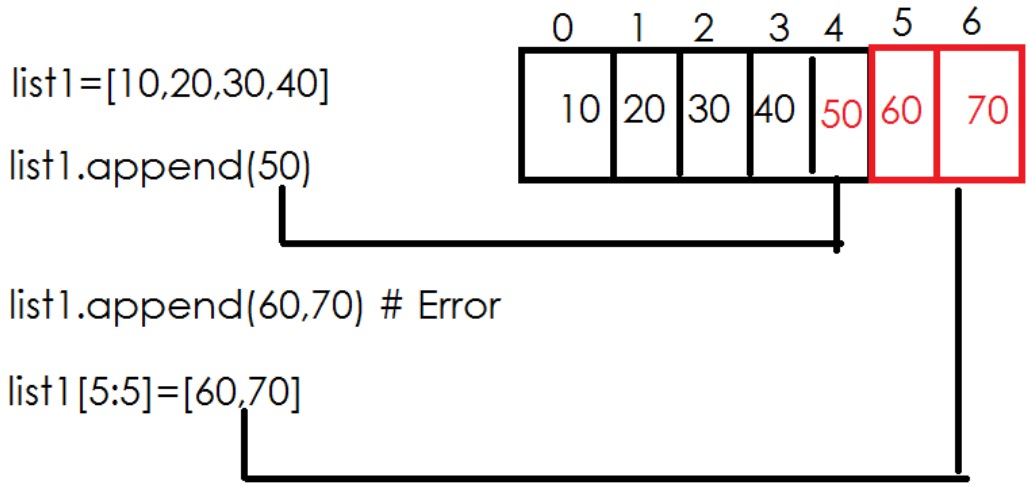
How to append more than one value?
Append more than one value is done using slicing.

Syntax: list-name[lengthlist:lengthlist]




```
salesList1=[1000,2000,3000]

salesList2=[4000,5000,6000,7000]

salesList1[len(salesList1):len(salesList1)]=salesList2

        print(salesList1)    [1000,200,3000,4000,5000,6000,7000]
        print(salesList2)    [4000,5000,6000,7000]
```

Example:

```
list1=[10,20,30,40,50]
list2=[60,70,80]
print(f'Before Append List1 {list1}')
print(f'List2 content is {list2}')
list1[len(list1):len(list1)]=list2
print(f'After Append List {list1}')
```

Output:

```
Before Append List1 [10, 20, 30, 40, 50]
List2 content is [60, 70, 80]
After Append List [10, 20, 30, 40, 50, 60, 70, 80]
```

Inserting value at given position

Insert(index,object)
This method inserts one object before index.

```
>>> playerName=["Rahul","Rohit","Virat","Surya"]
>>> print(playerName)
['Rahul', 'Rohit', 'Virat', 'Surya']
>>> playerName.insert(1,"Shubham")
>>> print(playerName)
['Rahul', 'Shubham', 'Rohit', 'Virat', 'Surya']
>>> list1=[10,20,30]
>>> list1.insert(1,100,200)
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    list1.insert(1,100,200)
TypeError: insert expected 2 arguments, got 3
```

Inserting more than one value/object?
Inserting more than one value is done using slicing.

List-name[startindex:endindex]=iterable

startindex:endindex must be same.

```
>>> list1=[10,20,30,40,50]
>>> print(list1)
[10, 20, 30, 40, 50]
>>> list1[2:2]=[88,99]
>>> print(list1)
[10, 20, 88, 99, 30, 40, 50]
```

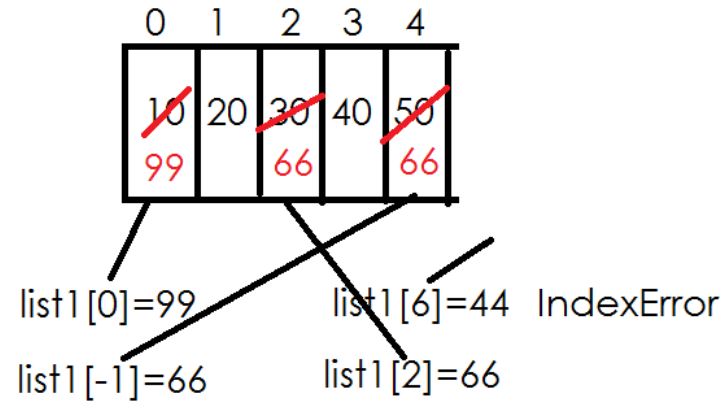
Replacing or updating values/elements
Replacing values are done in two ways.

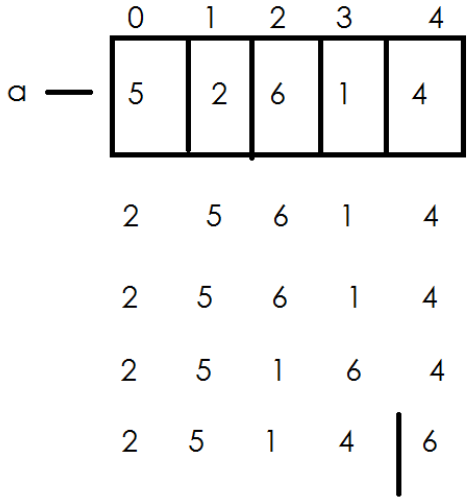
1. Using index
2. Using slicing

Using index one value is replaced.
Using slicing more than one value is replaced.

Syntax of replacing one value
list-name[index]=value

list1=[10,20,30,40,50]





bubble Sort

```
for i in range(5):
    for j in range(0,4):
        if a[j]>a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]
```

```
>>> list1=[10,20,30,40,50]
>>> print(list1)
[10, 20, 30, 40, 50]
>>> list1[0]=99
>>> print(list1)
```

```
[99, 20, 30, 40, 50]
>>> list1[-1]=88
>>> print(list1)
[99, 20, 30, 40, 88]
>>> list1[2]=66
>>> print(list1)
[99, 20, 66, 40, 88]
```

Example:

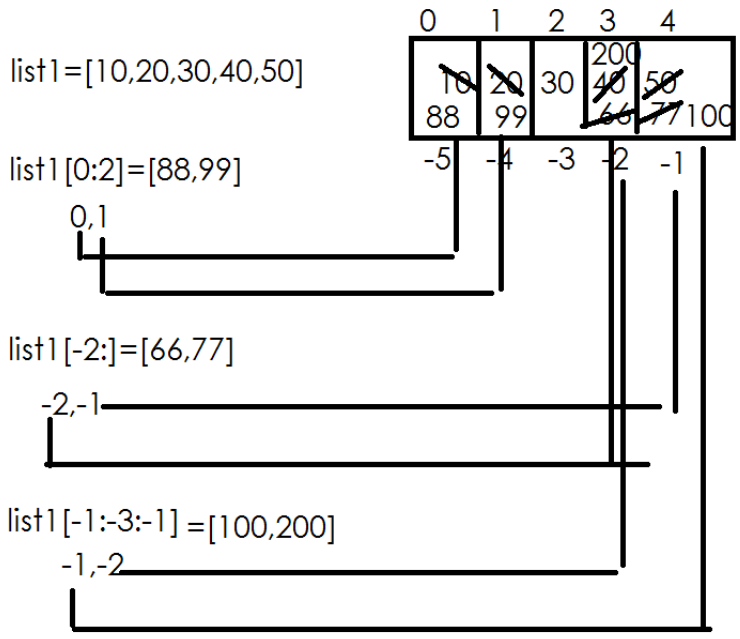
```
# Write a program to sort elements of the list
# in ascending order using bubble sorting/exchange sorting

n=int(input("Enter How Many Elements?"))
a=[]
for i in range(n):
    value=int(input("Enter Value "))
    a.append(value)

print(f'Before Sorting {a}')

for i in range(n):
    for j in range(n-1):
        if a[j]>a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]


print(f'After Sorting {a}')
```

Output:

```
Enter How Many Elements?5
Enter Value 4
Enter Value 2
Enter Value 1
Enter Value 5
Enter Value 3
Before Sorting [4, 2, 1, 5, 3]
After Sorting [1, 2, 3, 4, 5]
```

Syntax of replacing more than one value

list-name[startindex:endindex]=iterable



Example:

```
>>> a=list(range(10,110,10))
>>> print(a)
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> a[:2]=[88,99]
>>> print(a)
[88, 99, 30, 40, 50, 60, 70, 80, 90, 100]
>>> a[-2:]=[99,111]
>>> print(a)
[88, 99, 30, 40, 50, 60, 70, 80, 99, 111]
>>> a[::2]=[1,2,3,4,5]
>>> print(a)
[1, 99, 2, 40, 3, 60, 4, 80, 5, 111]
```

Delete elements or values from list

Python allows removing elements from list in different ways.

1. Using del keyword
2. Using remove method
3. Using pop method
4. Using clear method

del keyword
This keyword is used to delete one or more than one element from list.
This delete keyword required.

1. Index
2. Slicing

List → Mutable
Tuple
String
Range
Bytes
Bytearray → Mutable