Replacing values or updating values
List is a mutable collection; we can replace or update values of list
This replacing or updating values of list is done in 2 ways

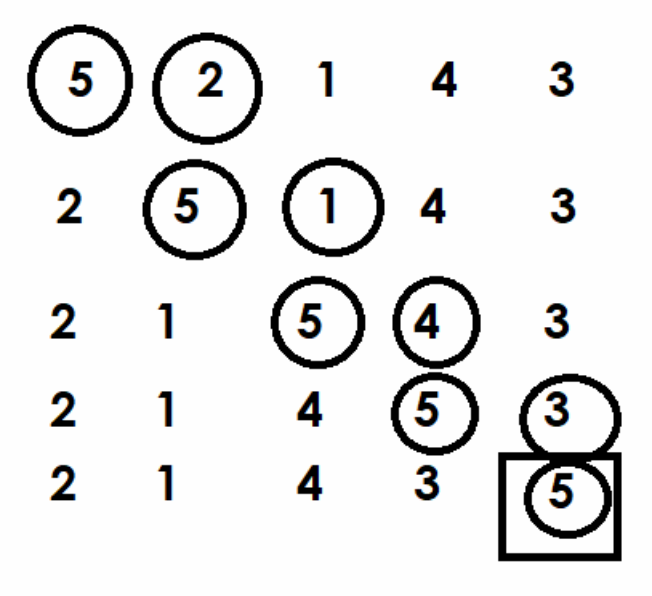
1. using index
2. using slicing

Replacing a single value using index

Using index we can replace single value, if the index is not within range it raises IndexError. If index is within range it replaces the value

Syntax: list-name[index]=value

```
>>> A=[10,20,30,40,50]
>>> print(A)
[10, 20, 30, 40, 50]
>>> A[1]=99
>>> print(A)
[10, 99, 30, 40, 50]
>>> A[-2]=88
>>> print(A)
[10, 99, 30, 88, 50]
>>> A[0]=77
>>> print(A)
[77, 99, 30, 88, 50]
>>> A[2]=66
>>> print(A)
[77, 99, 66, 88, 50]
>>> A[5]=100
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    A[5]=100
IndexError: list assignment index out of range
>>> A[-7]=90
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    A[-7]=90
IndexError: list assignment index out of range
```



Example:

```
# Write a program to input n integer values into list
# and sort elements/values in ascending order using
# bubble sorting
n=int(input("How many integer values?"))
A=[]

for i in range(n):
    value=int(input("Enter any 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}')

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

```
How many integer values?5
Enter any value 3
Enter any value 1
Enter any value 5
Enter any value 3
Enter any value 2
Before Sorting [3, 1, 5, 3, 2]
After Sorting [1, 2, 3, 3, 5]
After Sorting [5, 3, 3, 2, 1]
```

Example:

```
# Python program to interchange first and last elements
# in a list

A=[10,20,30,40,50]
print(f'Before Swapping {A}')
A[0],A[-1]=A[-1],A[0]
print(f'After Swaping {A}')
```

Output

```
Before Swapping [10, 20, 30, 40, 50]
After Swaping [50, 20, 30, 40, 10]
```

Replacing multiple values using slicing operator

Index can be used to replace single value.
Using slicing operator we can replace multiple values.

Syntax:

```
list-name[start:stop:step]=collection/iterable
```

```
>>> A=[10,20,30,40,50,60,70,80,90,100]
>>> print(A)
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> A[0:3]=[11,22,33]
>>> print(A)
```

```
[11, 22, 33, 40, 50, 60, 70, 80, 90, 100]
>>> A[-3:]=[88,99,111]
>>> print(A)
[11, 22, 33, 40, 50, 60, 70, 88, 99, 111]
>>> A[::2]=[1,2,3,4,5]
>>> print(A)
[1, 22, 2, 40, 3, 60, 4, 88, 5, 111]
>>> A[-1:-4:-1]=10,20,30
>>> print(A)
[1, 22, 2, 40, 3, 60, 4, 30, 20, 10]
>>> A[-3:]=[1,2,3,4,5,6]
>>> print(A)
[1, 22, 2, 40, 3, 60, 4, 1, 2, 3, 4, 5, 6]
```

Example:

```
# Python Program to Swap Two Elements in a List
# First two elements with last two elements

A=[10,20,30,40,50,60,70,80,90,100]
print(f'Before Swaping {A}')
A[0],A[1],A[-1],A[-2]=A[-1],A[-2],A[0],A[1]
print(f'After Swaping {A}')
A[:2],A[-1:-3:-1]=A[-1:-3:-1],A[:2]
print(f'After Swaping {A}')
```

Output

```
Before Swaping [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
After Swaping [100, 90, 30, 40, 50, 60, 70, 80, 20, 10]
After Swaping [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

Deleting elements/values from list

Deleting elements or values from list is done in different ways

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

using del keyword

“del” keyword is used to delete one or more than one element from list.

Del keyword uses index and slicing for deleting one or more than one value.

1. Using index
2. Using slicing

Using index, we can delete single element or value from list
Using slicing, we can delete multiple elements or values from list

Using index

Syntax: del list-name[index]

If index is within range, it delete element from list by shifting elements left side
If index is not within range or invalid index, it generates IndexError

```
>>> A=[10,20,30,40,50,60,70]
>>> print(A)
[10, 20, 30, 40, 50, 60, 70]
>>> del A[0]
>>> print(A)
[20, 30, 40, 50, 60, 70]
>>> del A[-1]
>>> print(A)
[20, 30, 40, 50, 60]
>>> del A[-3]
>>> print(A)
[20, 30, 50, 60]
>>> del A[4]
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    del A[4]
IndexError: list assignment index out of range
```

Deleting multiple elements using slicing

Syntax: del list-name[start:stop:step]

Example:

```
>>> A=list(range(10,110,10))
>>> print(A)
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> del A[:2]
>>> print(A)
[30, 40, 50, 60, 70, 80, 90, 100]
>>> del A[3:6]
>>> print(A)
[30, 40, 50, 90, 100]
>>> del A[-3:]
>>> print(A)
[30, 40]
>>> del A[:]
>>> print(A)
[]
```

using remove method

remove() is a method of list, this method is used to remove a value from list using value. It will remove first occurrence value. If value not exists within list, it raises ValueError

Syntax: list-name.remove(value)

```
>>> names=["naresh","suresh","ramesh","rajesh"]
>>> print(names)
['naresh', 'suresh', 'ramesh', 'rajesh']
>>> names.remove("ramesh")
>>> print(names)
['naresh', 'suresh', 'rajesh']
>>> names.remove("ramesh")
Traceback (most recent call last):
  File "<pyshell#47>", line 1, in <module>
    names.remove("ramesh")
ValueError: list.remove(x): x not in list
>>> A=[10,20,30,30,30,40,30,30,10]
>>> print(A)
[10, 20, 30, 30, 30, 40, 30, 30, 10]
>>> A.remove(30)
>>> print(A)
```

[10, 20, 30, 30, 40, 30, 30, 10]