Example:
Write a program remove/delete multiple occurences
of given value (10)

```
A=[10,20,10,10,30,40,20,20,10,10,10,10]
print(f'Before deleting or removing {A}')
value=10
while True:
    if value in A:
        A.remove(value)
    else:
        break

print(f'After deleting or removing {A}')
```

Output
Before deleting or removing [10, 20, 10, 10, 30, 40, 20, 20, 10, 10, 10, 10]
After deleting or removing [20, 30, 40, 20, 20]

clear() method
clear() method of list remove all the values or elements from list.
This method empty the list

```
>>> A=[10,20,30,40,50]
>>> print(A)
[10, 20, 30, 40, 50]
>>> A.clear()
>>> print(A)
[]
>>> B=[10,20,30,40,50]
>>> print(B)
[10, 20, 30, 40, 50]
>>> del B[:]
>>> print(B)
[]
```

pop() method
pop() method is used for implementing STACK data structure
Stack allows two operations
   1. Push → Adding element/value

2. Pop → removing value/element

Pop() method always remove last value.

STACK follows a method called LIFO (Last In First Out). The value added last is removed first.

Pop() method perform two operations
1. Reading
2. Removing

Before removing value, it first read value and then removes

Syntax: <variable-name>=list-name.pop(index=-1)

```
>>> A=[10,20,30,40,50]
>>> print(A)
[10, 20, 30, 40, 50]
>>> x=A.pop()
>>> print(x)
50
>>> print(A)
[10, 20, 30, 40]
>>> y=A.pop()
>>> print(y)
40
>>> print(A)
[10, 20, 30]
>>> z=A.pop(0)
>>> print(A)
[20, 30]
>>> print(z)
10
```

Example:
```
# Write a program to implement STACK data structure
S=[]
while True:
    print("1.Push")
    print("2.Pop")
    print("3.View")
    print("4.Exit")
    opt=int(input("Enter Your Option "))
```

```python
    if opt==1:
        value=int(input("Enter Any Value "))
        S.append(value)
        print(f'{value} is pushed in stack')
    elif opt==2:
        if len(S)==0:
            print("Stack is empty")
        else:
            value=S.pop()
            print(f'{value} is poped from stack')
    elif opt==3:
        print(f'Stack is {S}')
    elif opt==4:
        break
    else:
        print("invalid option")
```

Output
1.Push
2.Pop
3.View
4.Exit
Enter Your Option 1
Enter Any Value 10
10 is pushed in stack
1.Push
2.Pop
3.View
4.Exit
Enter Your Option 1
Enter Any Value 20
20 is pushed in stack
1.Push
2.Pop
3.View
4.Exit
Enter Your Option 1

Inserting values into list
Inserting values inside list is done in 2 ways

1. Using insert method
2. Using slicing operator

insert() method

insert method of list insert a given value at given index (position). It allows inserting only single value or element.

Syntax: <list-name>.insert(index,value)

If index is within range, it inserts value a given index
If index is not within range, it add at beginning or ending of list

Example:
```
>>> A=[10,20,30,40,50]
>>> print(A)
[10, 20, 30, 40, 50]
>>> A.insert(0,99)
>>> print(A)
[99, 10, 20, 30, 40, 50]
>>> A.insert(2,88)
>>> print(A)
[99, 10, 88, 20, 30, 40, 50]
>>> A.insert(10,100)
>>> print(A)
[99, 10, 88, 20, 30, 40, 50, 100]
>>> A.insert(-15,200)
>>> print(A)
[200, 99, 10, 88, 20, 30, 40, 50, 100]
>>> A.insert(-2,77)
>>> print(A)
[200, 99, 10, 88, 20, 30, 40, 77, 50, 100]
```

Inserting multiple values
Inserting multiple values are done using slicing operator

Syntax: <list-name>[start:stop]=iterable/collection

startindex must be equal to stop index

```
>>> A=[10,20,30,40,50]
>>> print(A)
[10, 20, 30, 40, 50]
>>> A[0:0]=[1,2,3]
>>> print(A)
[1, 2, 3, 10, 20, 30, 40, 50]
>>> A[3:3]=[88,99]
>>> print(A)
[1, 2, 3, 88, 99, 10, 20, 30, 40, 50]
>>> A[-2:-2]=[4,5]
>>> print(A)
[1, 2, 3, 88, 99, 10, 20, 30, 4, 5, 40, 50]
>>> A[0:2]=[11,12]
>>> print(A)
[11, 12, 3, 88, 99, 10, 20, 30, 4, 5, 40, 50]
```

Example:
Write a program to remove duplicate values from list

```
A=[10,20,10,10,20,20,30,40,30,40,50]
B=[]

for value in A:
    if value not in B:
        B.append(value)

print(f'List A {A}')
print(f'List B {B}')
```

Output
List A [10, 20, 10, 10, 20, 20, 30, 40, 30, 40, 50]
List B [10, 20, 30, 40, 50]

Example
Write a program to remove duplicate values from list

```
A=[10,20,10,10,20,20,30,40,30,40,50]
i=0
size=len(A)
while i<size:
```

```
value=A[i]
j=i+1
while j<size:
    if A[j]==value:
        del A[j]
        size=size-1
        continue
    j=j+1
i=i+1
```

Output
[10, 20, 30, 40, 50]