Replacing multiple values using slicing

Syntax: list-name[start:stop:step]=iteable

Example:

```
>>> A=[10,20,30,40,50,60,70,80,90,100]
>>> print(A)
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> A[0:2]=[11,22]
>>> print(A)
[11, 22, 30, 40, 50, 60, 70, 80, 90, 100]
>>> A[-2:]=[99,111]
>>> print(A)
[11, 22, 30, 40, 50, 60, 70, 80, 99, 111]
>>> A[0::2]=range(1,6)
>>> print(A)
[1, 22, 2, 40, 3, 60, 4, 80, 5, 111]
>>> B=[1,2,3,4,5,6,7,8,9,10]
>>> print(B)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> B[-1:-5:-1]="ABCD"
>>> print(B)
[1, 2, 3, 4, 5, 6, 'D', 'C', 'B', 'A']
```

Example:

```
# Write a program to input n integers into list
# and sort them in ascending order

A=[]
n=int(input("How many integer?"))

for i in range(n):
```

```
    value=int(input("Enter any value"))
    A.append(value)

print(f'Before Sorting {A}')

# Bubble Sorting

for i in range(n):
    for j in range(0,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?5
Enter any value4
Enter any value1
Enter any value5
Enter any value2
Enter any value3
Before Sorting [4, 1, 5, 2, 3]
After Sorting [1, 2, 3, 4, 5]
```

Deleting elements from list

Deleting elements 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” is keyword which is used to delete one or more than one element or value.
“del” keyword required,

1. Index for deleting one value/element
2. Slicing for deleting multiple values

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

Example:

```
A=list(range(10,110,10))
print(A)
del A[0]
print(A)
del A[-2]
print(A)
#del A[10] IndexError
```

Output

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

Example:

```
A=list(range(10,110,10))
print(A)
del A[0:3]
print(A)
B=list(range(10,110,10))
print(B)
del B[::2]
```

```
print(B)
C=list(range(10,110,10))
print(C)
del C[::-2]
print(C)
```

Output

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

Example:

```
# Write a program to remove None values from the following
# list using index

A=[10,20,None,30,40,None,60,70,80,None,None]
x=len(A)
i=0
while i<x:
    if A[i]==None:
        del A[i]
        x=x-1
        continue
    i=i+1

print(f'After Deleting {A}')
```

Output

After Deleting [10, 20, 30, 40, 60, 70, 80]

Using remove method
This method deletes or remove element from list.
This method required input as value/element
If element exists within list it remove value else raises ValueError

Syntax: list-name.remove(value)

This method remove only first occurrence of value

Example:

```
A=[10,20,30,10,20,20,40,20,20,50]
print(A)
A.remove(20)
print(A)
A.remove(10)
print(A)
A.remove(100)
```

Output

```
[10, 20, 30, 10, 20, 20, 40, 20, 20, 50]
[10, 30, 10, 20, 20, 40, 20, 20, 50]
[30, 10, 20, 20, 40, 20, 20, 50]
Traceback (most recent call last):
  File "D:/python11amfasttrack/test162.py", line 7, in <module>
    A.remove(100)
ValueError: list.remove(x): x not in list
```

Example:

```
# Write a program to delete or remove all occurences
# 20 from list
```

```
A=[10,20,30,10,20,20,40,20,20,50]
print(f'Before Deleting {A}')
while True:
    if 20 in A:
        A.remove(20)
    else:
        break
print(f'After Deleting {A}')

while 10 in A:
    A.remove(10)
print(f'After Deleting {A}')
```

Output

```
Before Deleting [10, 20, 30, 10, 20, 20, 40, 20, 20, 50]
After Deleting [10, 30, 10, 40, 50]
After Deleting [30, 40, 50]
```

Example:

```
# Write a program to remove duplicate values from list

A=[1,2,1,2,3,4,5,2,3,4,3,4,5]
B=[]

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

print(A)
print(B)
```

Output

[1, 2, 1, 2, 3, 4, 5, 2, 3, 4, 3, 4, 5]
[1, 2, 3, 4, 5]

 # Write a program to count of each value in list

```
A=[1,2,1,2,3,4,5,2,3,4,3,4,5]
B=[]

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

print(A)
print(B)
for value in B:
    c=0
    for value1 in A:
        if value==value1:
            c=c+1
    print(f'{value}--{c}')
```

pop() method

This method perform 2 operations

1. read value
2. remove

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

This method always removes last element.
This method is used to implement **STACK** data structure. Stack follows LIFO (Last In First Out).

Example:

```
>>> 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(z)
10
>>> print(A)
[20, 30]
```

clear()

This method is used for removing all the values from list (OR) empty list

Example:

```
>>> A=list(range(10,110,10))
>>> print(A)
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> A.clear()
>>> print(A)
[]
```

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

sort()

it is a mutable method of list

this method sort elements within list (OR) in place

Syntax: list-name.sort(key=function,reverse=False)

```
>> A=[4,1,8,3,5,2,6,7]
>>> print(A)
[4, 1, 8, 3, 5, 2, 6, 7]
>>> A.sort()
>>> print(A)
[1, 2, 3, 4, 5, 6, 7, 8]
>>> A.sort(reverse=True)
>>> print(A)
[8, 7, 6, 5, 4, 3, 2, 1]
>>> B=["D","c","a","d","B","A","C","b"]
>>> print(B)
['D', 'c', 'a', 'd', 'B', 'A', 'C', 'b']
>>> B.sort()
>>> print(B)
['A', 'B', 'C', 'D', 'a', 'b', 'c', 'd']
>>> B.sort(key=str.upper)
>>> print(B)
['A', 'a', 'B', 'b', 'C', 'c', 'D', 'd']
```

```
>>> B.sort(key=str.upper,reverse=True)
>>> print(B)
['D', 'd', 'C', 'c', 'B', 'b', 'A', 'a']
>>> C=["4","1","8","3","5","2","6","7"]
>>> print(B)
['D', 'd', 'C', 'c', 'B', 'b', 'A', 'a']
>>> print(C)
['4', '1', '8', '3', '5', '2', '6', '7']
>>> C.sort(key=int)
>>> print(C)
['1', '2', '3', '4', '5', '6', '7', '8']
```

Example:

```
# Write a program to find second maximum in a given
# list of values

A=[10,40,60,20,70,50,80]
A.sort()
print(f'List is {A}')
print(f'Second maximum is {A[-2]}')
print(f'Second minimum is {A[1]}')
```

Output

```
List is [10, 20, 40, 50, 60, 70, 80]
Second maximum is 70
Second minimum is 20
```

Creating copy of the list

Python allows creating copy using 2 different methods

1. shallow copy
2. deep copy