Python List

Python offers a range of compound data types often referred to as sequences. List is one of the most frequently used and very versatile data type used in Python.

How to create a list?

In Python programming, a list is created by placing all the items (elements) inside a square bracket [ ], separated by commas.

It can have any number of items and they may be of different types (integer, float, string etc.).

```
# empty list
n = []
# list of integers
n = [1, 2, 3]
# list with mixed datatypes
n = [1, “Hello”, 3.4]
```

Also, a list can even have another list as an item. This is called nested list.

```
# nested list
n = [“hello”, [8, 4, 6], [‘a’]]
```

How to access elements from a list?

There are various ways in which we can access the elements of a list.

List Index

We can use the index operator [ ] to access an item in a list. Index starts from 0. So, a list having 5 elements will have index from 0 to 4.

Note : Trying to access an element other then the range will raise an IndexError. The index must be an integer. We can’t use float or other types, this will result into TypeError.

Nested list are accessed using nested indexing.

Examples:

n1=[‘c’ , ‘o’ , ‘m’ , ‘p’ , ‘u’ , ‘t’ , ‘e’ , ‘r’]

0	1	2	3	4	5	6	7
c	o	m	p	u	t	e	r

print(n1)

#output : [‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]

print(n1[0])

#output: c

print(n1[2])

#output : m

print(n1[4])

#output: u

#print(n1[2.5]) #error

#error

Example:
n2=[‘c’ , ‘a’ , ‘t’ , ‘a’ , ‘l’ , ‘y’ , ‘s’ , ‘t’]

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
c	a	t	a	l	y	s	t

```
n2=[‘c’ , ‘a’ , ‘t’ , ‘a’ , ‘l’ , ‘y’ , ‘s’ , ‘t’]
print(n2)
print(n2[0])
print(n2[2])
print(n2[5])
print(n2[7])
print(n2[1])
print(n2[1.7])
```

output:

```
[‘c’, ‘a’, ‘t’, ‘a’, ‘l’, ‘y’, ‘s’, ‘t’]
c
t
y
t
a
```

```
n1=['c','o','m','p','u','t','e','r']
print(n1)
print(n1[0])
print(n1[2])
print(n1[4])
```

Output

```
['c', 'o', 'm', 'p', 'u', 't', 'e', 'r']
c
m
u
```

>>>

```
n2=['c','a','t','a','l','y','s','t']
print(n2)
print(n2[0])
print(n2[2])
print(n2[5])
print(n2[7])
print(n2[1])
#Output
['c', 'a', 't', 'a', 'l', 'y', 's', 't']
c
t
```

y
t
a
>>>
Example of Nested List
n = [“Happy”, [2,0,1,5]]
# Nested indexing
print(n[0][0])
# Output: H
print(n[0][1])
# Output: a
print(n[1][0])
# Output: 2
print(n[1][1])
# Output: 0
print(n[1][3])
# Output: 5
Output:
H
a
2
0
5
Negative indexing
Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.
Example:
Positive indexing (from left side index starts from 0(zero))

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
c	o	m	p	u	t	e	r
-8	**-7**	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**

Negative indexing (from right side index starts from -1)
n1=[‘c’ , ‘o’ , ‘m’ , ‘p’ , ‘u’ , ‘t’ , ‘e’ , ‘r’]
print(n1)
print(n1[-1])
print(n1[-2])
print(n1[-4])
output:
[‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
r

e
u

Example:
n2=[‘c’ , ‘a’ , ‘t’ , ‘a’ , ‘l’ , ‘y’ , ‘s’ , ‘t’]
Positive indexing (from left side index starts from 0(zero))

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
c	a	t	a	l	y	s	t
-8	**-7**	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**

Negative indexing (from right side index starts from -1)
n2=[‘c’ , ‘a’ , ‘t’ , ‘a’ , ‘l’ , ‘y’ , ‘s’ , ‘t’]
print(n2)
print(n2[-0]) # -0 is 0
print(n2[-2])
print(n2[-5])
print(n2[-7])
print(n2[-1])
print(n2[1])
print(n2[4])
print(n2[6])
output:
[‘c’, ‘a’, ‘t’, ‘a’, ‘l’, ‘y’, ‘s’, ‘t’]
c
s
a
a
t
How to slice lists in Python?
We can access a range of items in a list by using the slicing operator (colon).
print(n1[1:3])
#counting starts from 0
#will display elements from **start to position -1**
Example:
n1=[‘c’ , ‘o’ , ‘m’ , ‘p’ , ‘u’ , ‘t’ , ‘e’ , ‘r’]

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
c	o	m	P	u	t	e	R

n1=[‘c’,’o’,’m’,’p’,’u’,’t’,’e’,’r’]
print(n1)
output: [‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
print(n1[1:3])
output: [‘o’, ‘m’]
print(n1[1:4])
output : [‘o’, ‘m’, ‘p’]
print(n1[1:5])
output: [‘o’, ‘m’, ‘p’, ‘u’]
print(n1[2:3])
output: [‘m’]
print(n1[2:4])
output: [‘m’, ‘p’]
print(n1[2:5])
output: [‘m’, ‘p’, ‘u’]

Example:

n1=[‘c’ , ‘o’ , ‘m’ , ‘p’ , ‘u’ , ‘t’ , ‘e’ , ‘r’]

0	1	2	3	4	5	6	7
c	o	m	p	u	t	e	r

print(n1)

output : [‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]

print(n1[:0])

output: []

print(n1[:1])

(Only position 0 with be displayed i.e. : 0 to 1-1)

output : [‘c’]

print(n1[:2])

(only elements of position 0 to 2-1 will be displayed)

output: [‘c’, ‘o’]

print(n1[:3])

output : [‘c’, ‘o’, ‘m’]

print(n1[:4])

output : [‘c’, ‘o’, ‘m’, ‘p’]

print(n1[:5])

output : [‘c’, ‘o’, ‘m’, ‘p’, ‘u’]

Example:

n1=[‘c’ , ‘o’ , ‘m’ , ‘p’ , ‘u’ , ‘t’ , ‘e’ , ‘r’]

Positive indexing (from left side index starts from 0(zero))

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
c	O	m	p	u	t	e	R
-8	**-7**	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**

Negative indexing (from right side index starts from -1)

print(n1)
print(n1[:-3])
print(n1[:-4])
print(n1[:-5])

output:

[‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘c’, ‘o’, ‘m’, ‘p’, ‘u’]
[‘c’, ‘o’, ‘m’, ‘p’]
[‘c’, ‘o’, ‘m’]

How to change or add elements to a list?

List are mutable, meaning, their elements can be changed unlike string or tuple.
We can use assignment operator (=) to change an item or a range of items.

n=[1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10]

Positive index from left to right (starts from 0)

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

print(n)
n[0]=10
print(n)

Output:

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[10, 2, 3, 4, 5, 6, 7, 8, 9, 10]

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

1	2	3	4	5	6	7	8	9	10

```
n=[1,2,3,4,5,6,7,8,9,10]
print(n)
n[2]=300
n[6]=200
n[8]=80
print(n)
```

output:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 300, 4, 5, 6, 200, 8, 80, 10]
```

Group value changes:

n=[1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10]

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	2	3	4	5	6	7	8	9	10

```
print(n)
n[1:4]=[10,20,30]
#position changed will be 1,2,3 (start to pos-1)
print(n)
```

Output:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 10, 20, 30, 5, 6, 7, 8, 9, 10]
```

Example:

n=[1,2,3,4,5,6,7,8,9,10]

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	2	3	4	5	6	7	8	9	10

```
print(n)
n[5:8]=[100,200,300]
print(n)
```

Output:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 100, 200, 300, 9, 10]
```

Example:

n=[1,2,3,4,5,6,7,8,9,10]

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	2	3	4	5	6	7	8	9	10

```
print(n)
n[5:8]=[100,200,300]
print(n)
```

output:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 100, 200, 300, 9, 10]
```

Example: Note: 4 locations are replaced by 3 values

```
n=[1,2,3,4,5,6,7,8,9,10]
print(n)
n[5:9]=[100,200,300]
print(n)
```

Output:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 100, 200, 300, 10]
```

Example: Note: 4 locations are replaced by 2 values

n=[1,2,3,4,5,6,7,8,9,10]
print(n)
n[5:9]=[100,200]
print(n)
Output:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 100, 200, 10]
Example: Note: 4 locations are replaced by 1 value
n=[1,2,3,4,5,6,7,8,9,10]
print(n)
n[5:9]=[100]
print(n)
Output:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 100, 10]

0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10

#group change
n = [1,2,3,4,5,6,7,8,9,10]
print(n)
#[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# change 2nd to 4th items
n[1:4] = [20,30,40,50]
Note: 3 locations are replaced by 4 values
print(n)
#[1, 20, 30, 40, 50, 5, 6, 7, 8, 9, 10]

n = [1,2,3,4,5,6,7,8,9,10]
n[5:8]=[50,60,70,80]
print(n)
Note: 3 locations are replaced by 4 values
[1, 2, 3, 4, 5, 50, 60, 70, 80, 9, 10]

n=[1,2,3,4,5,6,7,8,9,10]
print(n)
n[5:7]=[100,200,300,400,500]
print(n)
Note: 2 locations are replaced by 5 values
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 100, 200, 300, 400, 500, 8, 9, 10]
Python List Methods
Methods that are available with list object in Python programming are given below.
They are accessed as list.method().
Python List Methods
append() – Add an element to the end of the list
extend() – Add all elements of a list to the another list
insert() – Insert an item at the defined index
#append method
n=[1,2,3,4,5]

print(n)
#[1, 2, 3, 4, 5]
n.append(6)
print(n)
#[1, 2, 3, 4, 5, 6]
#extend() method
n=[1,2,3,4,5]
print(n)
#[1, 2, 3, 4, 5]
n.extend([6,7,8])
print(n)
#[1, 2, 3, 4, 5, 6, 7, 8]

0	**1**	**2**	**3**	**4**
1	2	3	4	5

#insert method
Example:
n=[1,2,3,4,5]
print(n)
#Output: [1, 2, 3, 4, 5]
n.insert(1,10)
print(n)
#Output: [1, 10, 2, 3, 4, 5]
n.insert(4,40)
print(n)
#Output: [1, 10, 2, 3, 40, 4, 5]
Example:
#insert method
n=[1,2,3,4,5]
print(n)
#Output: [1, 2, 3, 4, 5]
n.insert(1,10)
print(n)
#Output:[1, 10, 2, 3, 4, 5]
Example:
n=[1,2,3,4,5]
print(n)
n.insert(2,20)
print(n)
#Output : [1, 2, 20, 3, 4, 5]
Example:
n=[1,2,3,4,5]
print(n)
#Output: [1, 2, 3, 4, 5]
n.insert(3,30)
print(n)
#Output : [1, 2, 3, 30, 4, 5]
Furthermore, we can insert one item at a desired location by using the method insert() or insert multiple items by squeezing it into an empty slice of a list.
odd = [1, 9]
odd.insert(1,3)
print(odd)

# Output: [1, 3, 9]
odd[2:2] = [5, 7]
print(odd)
# Output: [1, 3, 5, 7, 9]
We can also use + operator to combine two lists. This is also called concatenation.
n = [1, 3, 5]
print(n + [9, 7, 5])
# Output: [1, 3, 5, 9, 7, 5]
We can also use + operator to combine two lists. This is also called concatenation.
n=[1,2,3,4,5]
n1=[6,7,8,9,10]
print(“list n”)
print(n)
print(“list n1”)
print(n1)
print(“n+n1”)
print(n+n1)
#[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(n+[10,20,30])
output:
list n
[1, 2, 3, 4, 5]
list n1
[6, 7, 8, 9, 10]
n+n1
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 10, 20, 30]
How to delete or remove elements from a list?
remove() – Removes an item from the list
pop() – Removes and returns an element at the given index
clear() – Removes all items from the list
del() : to remove an element. We can delete one or more items from a list using the keyword del. It can even delete the list entirely.
n1=[‘c’,’o’,’m’,’p’,’u’,’t’,’e’,’r’]

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
c	o	m	p	u	t	e	R

#remove (remove the element by value)
Example:1
n1=[‘c’,’o’,’m’,’p’,’u’,’t’,’e’,’r’]
print(n1)
n1.remove(‘m’)
print(n1)
n1.remove(‘p’)
print(n1)
n1.remove(‘t’)
print(n1)
output:
[‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘c’, ‘o’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘c’, ‘o’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘c’, ‘o’, ‘u’, ‘e’, ‘r’]

Example:2

```
n=[0,1,2,3,4,5,6,7,8,9,10]
n.remove(4)
print(n)
#[0, 1, 2, 3, 5, 6, 7, 8, 9, 10]
n.remove(5)
print(n)
#[0, 1, 2, 3, 6, 7, 8, 9, 10]
n.remove(9)
print(n)
#[0, 1, 2, 3, 6, 7, 8, 10]
n.remove(10)
print(n)
#[0, 1, 2, 3, 6, 7, 8]
#n.remove(10)  #error will be displayed as element is not present
#print(n)
```

del (delete an element using index)

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
c	o	m	p	u	t	e	R

```
n1=['c','o','m','p','u','t','e','r']
print(n1)
```

#Output: ['c', 'o', 'm', 'p', 'u', 't', 'e', 'r']

```
del n1[0]
print(n1)
```

#Output: ['o', 'm', 'p', 'u', 't', 'e', 'r']

```
del n1[1]
print(n1)
```

#Output: ['o', 'p', 'u', 't', 'e', 'r']

#count starts from 0(zero)

Example:

```
n=[1,2,3,4,5,6,7,8,9,10]
del n[2]
print(n)
```

#Output: [1, 2, 4, 5, 6, 7, 8, 9, 10]

```
del n[4]
print(n)
```

#output: [1, 2, 4, 5, 7, 8, 9, 10]

```
del n[7]
print(n)
```

#Output: [1, 2, 4, 5, 7, 8, 9]

Delete by range:

```
n=[0,1,2,3,4,5,6,7,8,9,10]
del n[0:2] #elements of position 0 and 1 are deleted
print("n[0:2]",n)
```

#Output: n[0:2] [2, 3, 4, 5, 6, 7, 8, 9, 10]

```
n=[0,1,2,3,4,5,6,7,8,9,10]
del n[0:3]
print("n[0:3]",n) #elements of position 0,1, and 2 are deleted
```

#Output: n[0:3] [3, 4, 5, 6, 7, 8, 9, 10]

```
n=[0,1,2,3,4,5,6,7,8,9,10]
del n[0:5]
```

print(“n[0:5]”,n)
#Output: n[0:5] [5, 6, 7, 8, 9, 10]
n=[0,1,2,3,4,5,6,7,8,9,10]
del n[2:5]
print(“n[2:5]”,n)
#Output: n[2:5] [0, 1, 5, 6, 7, 8, 9, 10]
n=[0,1,2,3,4,5,6,7,8,9,10]
del n[4:8]
print(“n[4:8]”,n)
#Output: n[4:8] [0, 1, 2, 3, 8, 9, 10]
n=[0,1,2,3,4,5,6,7,8,9,10]
print(n)
del n
# delete entire list
print(n)
#Output: error : NameError: name ‘n’ is not defined
Example:
n = [‘p’,’r’,’o’,’b’,’l’,’e’,’m’]
# delete one item
del n[2]
print(n)
# Output: [‘p’, ‘r’, ‘b’, ‘l’, ‘e’, ‘m’]
# delete multiple items
del n[1:5]
print(n)
# Output: [‘p’, ‘m’]
# delete entire list
del n
print(n)
#Output: Error: List not defined
Python List Practice Examples
Example: 1
Give output:
n2=[‘c’,’a’,’t’,’a’,’l’,’y’,’s’,’t’]
Positive indexing

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
c	a	t	a	l	y	s	t
-8	**-7**	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**

Negative indexing
1. print(n2)
2. print(n2[:4])
3. print(n2[1:4])
4. print(n2[:-4])
5. print(n2[:6])
6. print(n2[:-6])
7. print(n2[1:6])
8. print(n2[3:6])
9. print(n2[2:6])
10. print(n2[:-2])

output:
[‘c’, ‘a’, ‘t’, ‘a’, ‘l’, ‘y’, ‘s’, ‘t’]

```
[‘c’, ‘a’, ‘t’, ‘a’]
[‘a’, ‘t’, ‘a’]
[‘c’, ‘a’, ‘t’, ‘a’]
[‘c’, ‘a’, ‘t’, ‘a’, ‘l’, ‘y’]
[‘c’, ‘a’]
[‘a’, ‘t’, ‘a’, ‘l’, ‘y’]
[‘a’, ‘l’, ‘y’]
[‘t’, ‘a’, ‘l’, ‘y’]
[‘c’, ‘a’, ‘t’, ‘a’, ‘l’, ‘y’]
```

Example: Give output

```
n1=[‘c’ , ‘o’ , ‘m’ , ‘p’ , ‘u’ , ‘t’ , ‘e’ , ‘r’]
print(n1)
print(n1[:])
print(n1[0:])
print(n1[1:])
print(n1[2:])
print(n1[3:])
print(n1[4:])
print(n1[5:])
```

output:

```
[‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
[‘u’, ‘t’, ‘e’, ‘r’]
[‘t’, ‘e’, ‘r’]
```

Example:

my_list = [‘c’ , ‘o’ , ‘m’ , ‘p’ , ‘u’ , ‘t’ , ‘e’ , ‘r’]

1. print(my_list[2:5])
2. print(my_list[:-5])
3. print(my_list[5:])
4. print(my_list[:])

Output:

```
# elements 2 to 4th   [‘m’, ‘p’, ‘u’]
# elements beginning to 4th   [‘c’, ‘o’, ‘m’]
# elements 6th to end    [‘t’, ‘e’, ‘r’]
# elements beginning to end      [‘c’, ‘o’, ‘m’, ‘p’, ‘u’, ‘t’, ‘e’, ‘r’]
```

Example: Give the output

```
n=[1,2,3,4,5,6,7,8,9,10]
print(n)
n[1:3]=[100]
print(n)
n[2:5]=[100,200]
print(n)
n[6:9]=[500,700,900]
print(n)
n[2:6]=[170,340]
print(n)
```

output:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 100, 4, 5, 6, 7, 8, 9, 10]
[1, 100, 100, 200, 7, 8, 9, 10]
[1, 100, 100, 200, 7, 8, 500, 700, 900]
[1, 100, 170, 340, 500, 700, 900]
```

Solve:

```
n1=[1,2,3,4,5]
n2=[6,7,8,9,10]
print(n1)
print(n2)
print(n1+n2)
n=n1+n2
print(n1)
print(n2)
print(n)
n=n2+n1
print(n)
```

output:

```
[1, 2, 3, 4, 5]
[6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5]
[6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[6, 7, 8, 9, 10, 1, 2, 3, 4, 5]
```

Example : Give the output

```
n1=[1,2,3,4,5]
n2=[6,7,8,9,10]
print(n1)
print(n2)
n1=n1+[100,200,300]
print(n1)
n2=n1+n2
print(n2)
n2=n2+[23,45,56]+n1
print(n2)
```

output:

```
[1, 2, 3, 4, 5]
[6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 100, 200, 300]
[1, 2, 3, 4, 5, 100, 200, 300, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 100, 200, 300, 6, 7, 8, 9, 10, 23, 45, 56, 1, 2, 3, 4, 5, 100, 200, 300]
```

Example : Give the output

```
n=[0,1,2,3,4,5,6,7,8,9,10]
n.remove(1)
print(n)
n.remove(9)
n.remove(5)
n.remove(4)
print(n)
```

output:

```
[0, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 2, 3, 6, 7, 8, 10]
```

Example : Give the oputput:

```
n=[0,1,2,3,4,5,6,7,8,9,10]
del n[1]
del n[1]
print(n)
del n[4]
del n[3]
print(n)
del n[6]
del n[2]
print(n)
```

output:

```
[0, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 3, 4, 7, 8, 9, 10]
[0, 3, 7, 8, 9]
```

Example : Give the output

```
n=[0,1,2,3,4,5,6,7,8,9,10]
del n[1]
del n[1]
print(n)
n=n+[10,20,30]
del n[7]
del n[9]
print(n)
n1=[5,6,7,3,4]
n=n+n1
del n[9]
del n[2]
print(n)
```

output:

```
[0, 3, 4, 5, 6, 7, 8, 9, 10]
[0, 3, 4, 5, 6, 7, 8, 10, 10, 30]
[0, 3, 5, 6, 7, 8, 10, 10, 5, 6, 7, 3, 4]
```