

## List

- List is an ordered sequence which is mutable and made up of one or more elements.
- List can have elements of different data type such as integer, float, string tuple or even another list.
- Elements of list are enclosed in square brackets and are separated by comma.
- List indices also starts from 0.

Example

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

O/P

[10, 20, 30, 40, 50]

```
list2 = [10, 3.14, 'Delhi']
print(list2)
```

O/P

[10, 3.14, 'Delhi']

```
list3 = [[10, 20], [30, 40]]
print(list3)
```

O/P

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

## Accessing Elements in a list

The elements of list can be accessed using indexing like strings.

Example list1 = [10, 4, 6, 8, 12]

O/P

# 1st element print(list1[0])

10

# last element print(list1[len(list1) - 1])

12

# 1st element print(list1[-len(list1)])

10

# last element print(list1[-1])

12

print(list1[15])

IndexError: list  
index out of range

## List Operations

Python allows certain operations on list data type which are as following -

- (1) concatenation
- (2) Repetition
- (3) Membership
- (4) slicing

### (1) concatenation

Python allows us to join two or more list using concatenation operator denoted by the symbol +.

Example :-    list\_1 = [10, 20, 30, 'Delhi']

              list\_2 = [40, 50, 60, 'Mumbai']

              list\_3 = list\_1 + list\_2

              print(list\_3)

O/P    [10, 20, 30, 'Delhi', 40, 50, 60, 'Mumbai']

### (2) Repetition

Python allows us to replicate a list using repetition operator denoted by symbol \*.

Example :-    list1 = ['Hello']

              list1 \*= 4

              print(list1)

O/P - ['Hello', 'Hello', 'Hello', 'Hello']

### (3) Membership

Like strings, membership operators can also be used in lists.

in checks if element is present in the list and returns True, else return False.

### Example :-

```
>>> list1 = ['Red', 'Green', 'Blue']
>>> 'Green' in list1
True
>>> 'Black' in list1
False
```

→ `not in` operator returns True if the element is not present in the list else it returns False.

### Example :-

```
>>> list1 = ['Red', 'Green', 'Blue']
>>> 'Black' not in list1
True
>>> 'Green' not in list1
False
```

## 4. slicing

Like strings, slicing operations can also be applied to lists.

syntax      <list-name> [start : stop : step]

Example -    list1 = ['Red', 'Green', 'Blue', 'Black',
                      'Yellow', 'Pink']
                  print(list1[2:4])

O/P      ['Blue', 'Black']

(i) When start is not given

print(list1[:3])      O/P      ['Red', 'Green', 'Blue']
                          ↓  
It will be taken as 0

(ii) When stop is not given

print(list1[2:])      O/P      ['Blue', 'Black', 'Yellow',
                          ↑  
It will goes 'Pink']  
till the length of string

(iii) when step is given

print(list1[0:5:2])

O/P

['Red', 'Blue', 'Yellow']

(iv) when negative index is used

print(list1[-3:-1])

O/P

['Black', 'Yellow']

Note: Reverse of list

print(list1[ : :-1])

O/P

['Pink', 'Yellow',  
'Black', 'Blue', 'Green',  
'Red']

## Traversing a list

(I) using for loop

list1 = ['Red', 'Green', 'Blue', 'Yellow', 'Black']

for item in list1 :

    print(item)

Output

Red  
Green  
Blue  
Yellow  
Black

another way to traverse using range() & len() funn

list1 = ['Red', 'Green', 'Blue']

for index in range(len(list1)):

    print(list1[i])

Output

Red  
Green  
Blue

(2) using while loop

Example :

```
list1 = ['Red', 'Green', 'Blue']
```

$$i=0$$

```
while i < len(list1):
```

print(lut1[i])

$i := 1$

### Output

81d

Gymn

Blue

## Nested lists

Nested lists When a list appears as an element of another list, it is called a nested list.

```
list1 = [1, 2, 'a', 'c', [6, 7, 8], 419]    OP  
print(list1)                                [6, 7, 8]  
print(list1[1])                            [7]
```

## copying - list

The screenshot shows the Visual Studio Code interface with a dark theme. On the left, there's a sidebar with various icons for file operations like copy, paste, find, and settings. The main area has a tab bar with 'practice.py X' selected. Below it is a preview pane showing the code. The terminal tab is active, displaying the following Python session:

```
PS C:\Users\DELL\Desktop\python> & C:/Users/DELL/AppData/Local/Programs/Python/Python311/python.exe c:/Users/DELL/Desktop/python/practice.py
[1, 2, 3]
[1, 2, 3]
1625142235904
1625142235904
[1, 2, 3, 9]
[1, 2, 3, 9]
1625142235904
1625142235904
PS C:\Users\DELL\Desktop\python>
```

Here we can see that, if we are trying to update in list1 then list2 also get updated. To overcome this problem, we can use these three methods -

- (1) slicing
- (2) list() method
- (3) copy() method

## 1. slicing

Here we can see that if we are updating in list1 then list2 will not be affected.

The screenshot shows a Windows desktop environment. In the center is a code editor window titled "practice.py". The code contains the following Python script:

```
1 list1 = [1,2,3]
2 list2 = list1[:]
3 print(list1)
4 print(list2)
5 print(id(list1))
6 print(id(list2))
7 list2.append(9)
8 print(list1)
9 print(list2)
10 print(id(list1))
11 print(id(list2))
```

To the right of the code editor is a terminal window titled "python". The terminal output shows the execution of the script:

```
PS C:\Users\DELL\Desktop\python> & C:/Users/DELL/AppData/Local/Programs/Python/Python311/python.exe c:/Users/DELL/Desktop/python/practice.py
[1, 2, 3]
[1, 2, 3]
1952802575104
1952804316224
[1, 2, 3]
[1, 2, 3, 9]
1952802575104
1952804316224
PS C:\Users\DELL\Desktop\python>
```

At the bottom of the screen is the Windows taskbar, which includes icons for File Explorer, Edge browser, Mail, and Google Chrome. The system tray shows the date and time as 07-10-2023 at 15:30.

## 2. list() method

A screenshot of a Windows desktop environment. In the center is a code editor window titled "practice.py". The code contains the following Python script:

```
list1 = [1,2,3]
list2 = list(list1)
print(list1)
print(list2)
print(id(list1))
print(id(list2))
list2.append(9)
print(list1)
print(list2)
print(id(list1))
print(id(list2))
```

To the right of the code editor is a terminal window titled "python". The terminal output shows the execution of the script and its results:

```
PS C:\Users\DELL\Desktop\python> & C:/Users/DELL/AppData/Local/Programs/Python/Python311/python.exe c:/Users/DELL/Desktop/python/practice.py
[1, 2, 3]
[1, 2, 3]
2038719681280
2038722142912
[1, 2, 3]
[1, 2, 3, 9]
2038719681280
2038722142912
PS C:\Users\DELL\Desktop\python>
```

The desktop taskbar at the bottom shows various pinned icons, including Microsoft Edge, File Explorer, and a file icon for "practice.py". The system tray indicates a battery level of 34% and the date/time as 07-10-2023 15:32.

## 3. copy() method

A screenshot of a Windows desktop environment, similar to the previous one. In the center is a code editor window titled "practice.py". The code contains the following Python script:

```
import copy
list1 = [1,2,3]
list2 = copy.copy(list1)
print(list1)
print(list2)
print(id(list1))
print(id(list2))
list2.append(9)
print(list1)
print(list2)
print(id(list1))
print(id(list2))
```

To the right of the code editor is a terminal window titled "python". The terminal output shows the execution of the script and its results:

```
PS C:\Users\DELL\Desktop\python> & C:/Users/DELL/AppData/Local/Programs/Python/Python311/python.exe c:/Users/DELL/Desktop/python/practice.py
[1, 2, 3]
[1, 2, 3]
1794340990976
1794340823744
[1, 2, 3]
[1, 2, 3, 9]
1794340990976
1794340823744
PS C:\Users\DELL\Desktop\python>
```

The desktop taskbar at the bottom shows various pinned icons, including Microsoft Edge, File Explorer, and a file icon for "practice.py". The system tray indicates a battery level of 34% and the date/time as 07-10-2023 15:34.

## LIST METHODS AND BUILT-IN Functions

<b>Method</b>	<b>Description</b>	<b>Example</b>
len()	Returns the length of the list passed as the argument	<pre>&gt;&gt;&gt; list1=[10,20,30,40,50] &gt;&gt;&gt; len(list1) 5</pre>
list()	Creates an empty list if no argument is passed Creates a list if a sequence is passed as an argument	<pre>&gt;&gt;&gt; list1 = list() &gt;&gt;&gt; list1 [ ] &gt;&gt;&gt; str1 = 'aeiou' &gt;&gt;&gt; list1 = list(str1) &gt;&gt;&gt; list1 ['a', 'e', 'i', 'o', 'u']</pre>
append()	Appends a single element passed as an argument at the end of the list  The single element can also be a list	<pre>&gt;&gt;&gt; list1 = [10,20,30,40] &gt;&gt;&gt; list1.append(50) &gt;&gt;&gt; list1 [10, 20, 30, 40, 50] &gt;&gt;&gt; list1 = [10,20,30,40] &gt;&gt;&gt; list1.append([50,60]) &gt;&gt;&gt; list1 [10, 20, 30, 40, [50, 60]]</pre>

extend()	Appends each element of the list passed as argument to the end of the given list	<pre>&gt;&gt;&gt; list1 = [10,20,30] &gt;&gt;&gt; list2 = [40,50] &gt;&gt;&gt; list1.extend(list2) &gt;&gt;&gt; list1 [10, 20, 30, 40, 50]</pre>
insert()	Inserts an element at a particular index in the list	<pre>&gt;&gt;&gt; list1=[10,20,30,40,50] &gt;&gt;&gt; list1.insert(2,25) &gt;&gt;&gt; list1 [10, 20, 25, 30, 40, 50] &gt;&gt;&gt; list1.insert(0,5) &gt;&gt;&gt; list1 [5, 10, 20, 25, 30, 40, 50]</pre>
count()	Returns the number of times a given element appears in the list	<pre>&gt;&gt;&gt; list1=[10,20,30,10,40,10] &gt;&gt;&gt; list1.count(10) 3 &gt;&gt;&gt; list1.count(90) 0</pre>
index()	Returns index of the first occurrence of the element in the list. If the element is not present, ValueError is generated	<pre>&gt;&gt;&gt; list1 = [10,20,30,20,40,10] &gt;&gt;&gt; list1.index(20) 1 &gt;&gt;&gt; list1.index(90) ValueError: 90 is not in list</pre>

remove()	Removes the given element from the list. If the element is present multiple times, only the first occurrence is removed. If the element is not present, then ValueError is generated	>>> list1=[10,20,30,40,50,30] >>> list1.remove(30) >>> list1 [10, 20, 40, 50, 30] >>> list1.remove(90) ValueError:list.remove(x):x not in list
pop()	Returns the element whose index is passed as parameter to this function and also removes it from the list. If no parameter is given, then it returns and removes the last element of the list	>>> list1 = [10,20,30,40,50,60] >>> list1.pop(3) 40 >>> list1 [10, 20, 30, 50, 60] >>> list1 = [10,20,30,40,50,60] >>> list1.pop() 60 >>> list1 [10, 20, 30, 40, 50]
reverse()	Reverses the order of elements in the given list	>>> list1=[34,66,12,89,28,99] >>> list1.reverse() >>> list1 [ 99, 28, 89, 12, 66, 34]  >>> list1=['Tiger' , 'Zebra' , 'Lion' , 'Cat' , 'Elephant','Dog'] >>> list1.reverse() >>> list1 ['Dog', 'Elephant', 'Cat', 'Lion', 'Zebra', 'Tiger']
sort()	Sorts the elements of the given list in-place	>>>list1 = ['Tiger','Zebra','Lion', 'Cat', 'Elephant' , 'Dog'] >>> list1.sort() >>> list1 ['Cat', 'Dog', 'Elephant', 'Lion', 'Tiger', 'Zebra']  >>> list1=[34,66,12,89,28,99] >>> list1.sort(reverse = True) >>> list1 [99,89,66,34,28,12]
sorted()	It takes a list as parameter and creates a new list consisting of the same elements arranged in sorted order	>>> list1=[23,45,11,67,85,56] >>> list2 = sorted(list1) >>> list1 [23, 45, 11, 67, 85, 56] >>> list2 [11, 23, 45, 56, 67, 85]

min()	Returns minimum or smallest element of the list	>>> list1=[34, 12, 63, 39, 92, 44] >>> min(list1) 12
max()	Returns maximum or largest element of the list	>>> max(list1) 92
sum()	Returns sum of the elements of the list	>>> sum(list1) 284

list comprehension

- list comprehension offers a shorter syntax when you want to create a new list based on existing list

syntax

`newlist = [expression for item in list if condition == True]`

Example creating a list of even numbers

```
list1 = [24, 94, 36, 75, 93, 107, 205, 208]
evenList = [ele for ele in list1 if ele%2 == 0]
print(evenList)
```

O/P [24, 94, 36, 208]

list comprehension can also be used with else also.

syntax

`newlist = [expression if condition==True else expression for item in list]`

Example creating a list if element is Even returns 1 else return 0

```
list1 = [24, 94, 36, 75, 93, 107, 205, 208]
newList = [1 if ele%2 == 0 else 0 for ele in list1]
print(newList)
```

O/P [1, 1, 1, 0, 0, 0, 0, 1]