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 seperated 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]
# 1st element    print( list1[0])
# last element   print( list1 [ len(list1) -1] )
# 1st element    print ( list1 [- len(list1)] )
# last element   print ( list1[ -1])
                 print ( list1 [15])
```

O/P
10
12
10
12
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 + list2
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])

↓ It will be taken as 0

o/p ['Red', 'Green', 'Blue']

(ii) When stop is not given

print(list1[2:])

↑ It will goes till the length of string

o/p ['Blue', 'Black', 'Yellow', 'Pink']

(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

(1) 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() fun^n

```
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(list1[i])
    i += 1
```

Output

Red
Green
Blue

Nested Lists

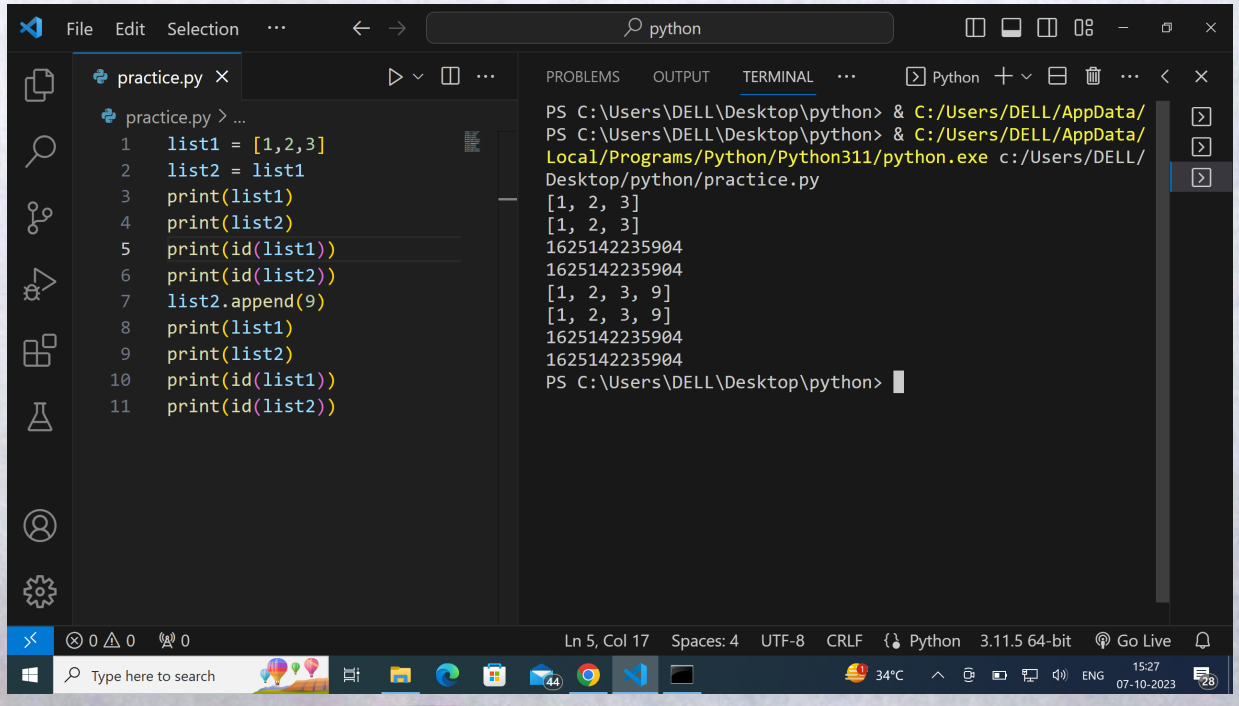
When a list appears as an element of another list, it is called a nested list.

```
list1 = [1, 2, 'a', 'c', [6, 7, 8], 4, 9]
print(list1[4])
print(list1[4][1])
```

o/p

[6, 7, 8]
[7]

copying list

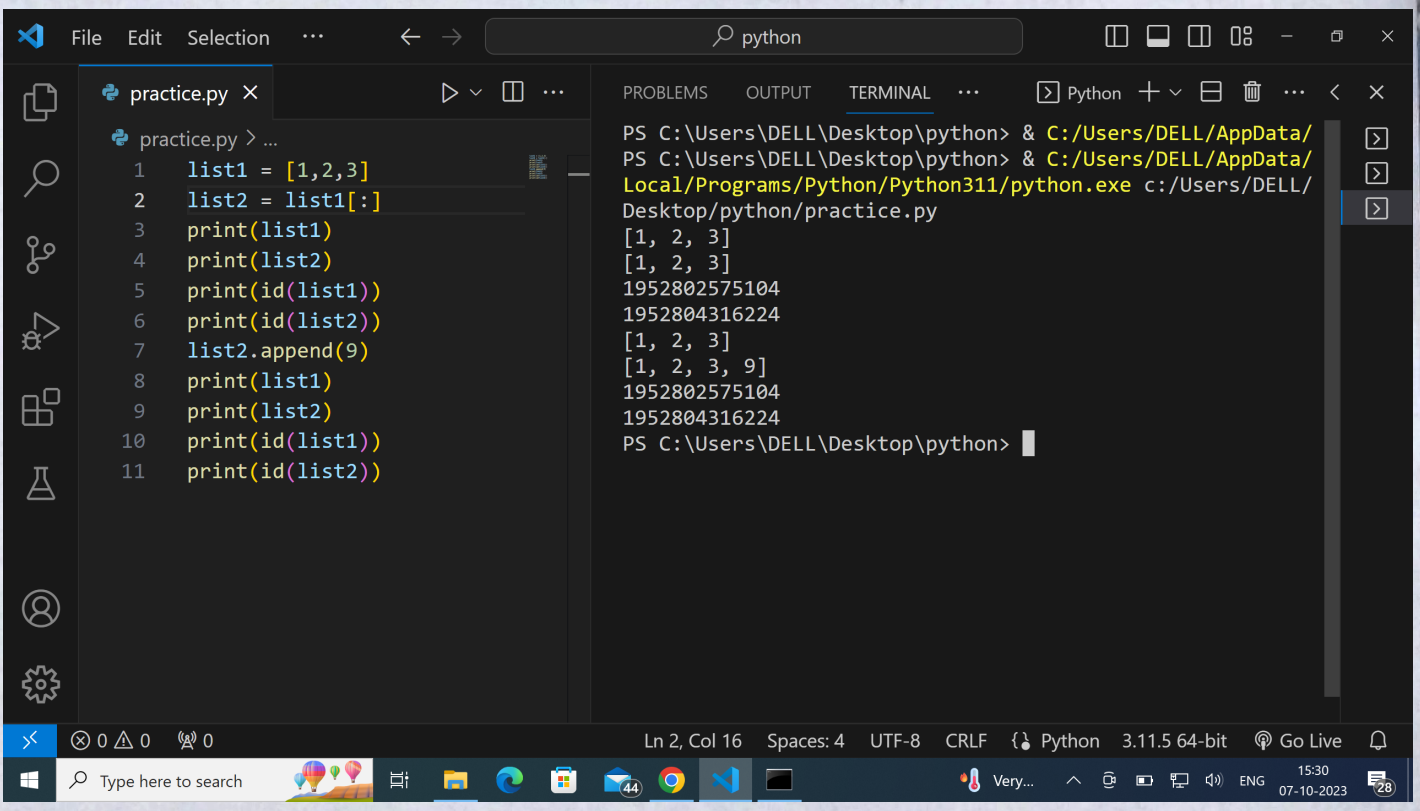


Here we can see that, if we are trying to update in list1 then list2 also get update.
To overcome this problem, we can use these three methods -

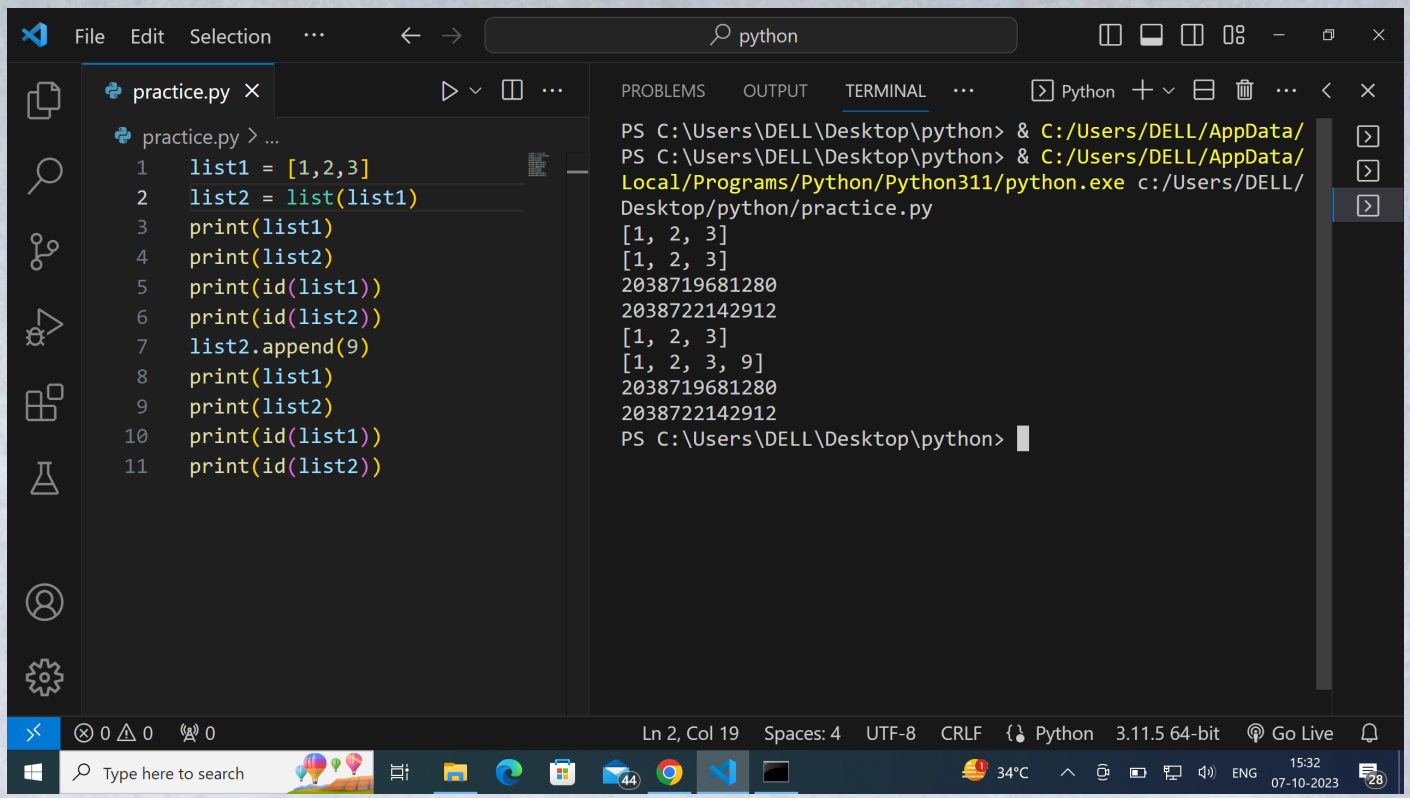
(1) Slicing
(2) list() method
(3) copy() method

1. Slicing

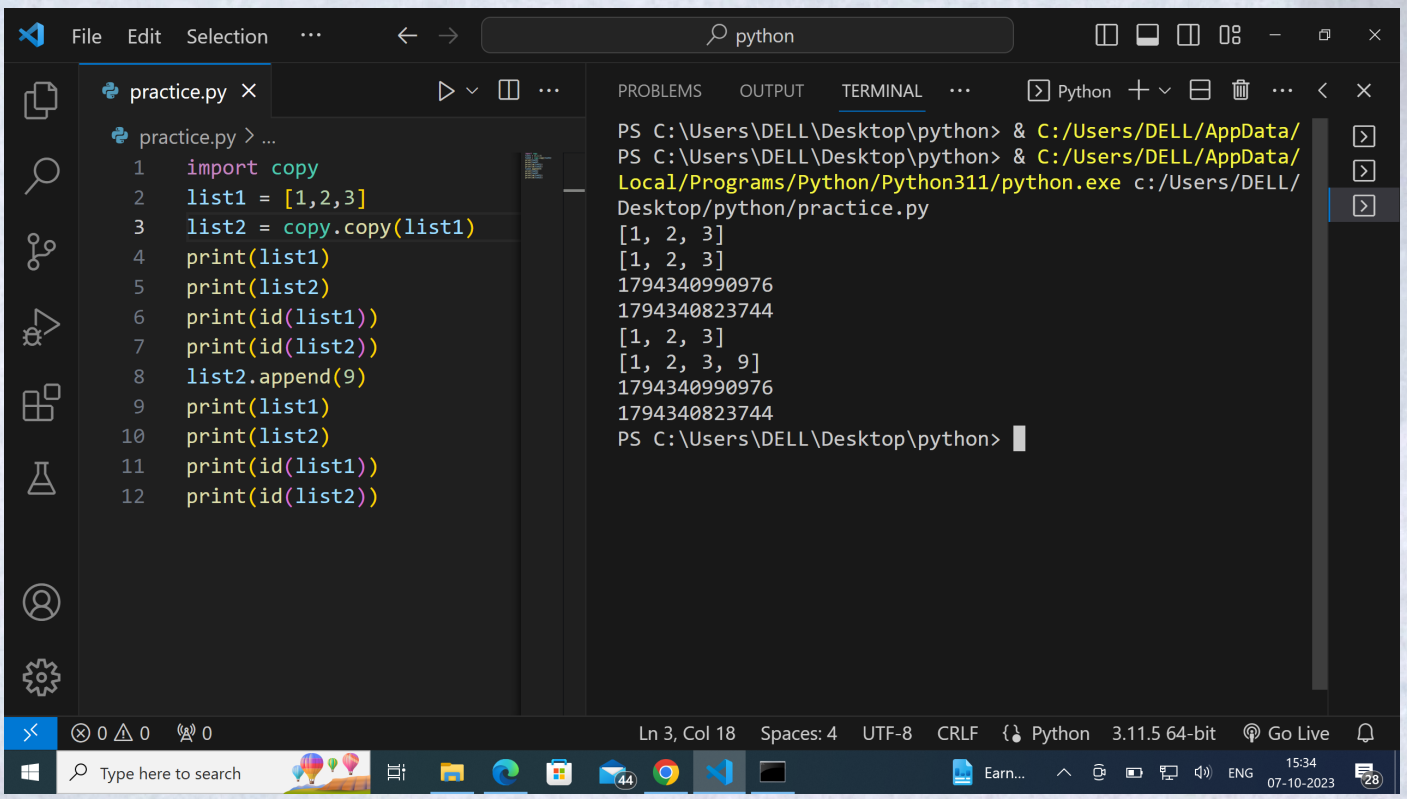
Here we can see that if we are updating in list1 then list2 will not be affected.



2. list() method



3. copy() method



List Methods and Built-in Functions

Method	Description	Example

len()	Returns the length of the list passed as the argument	>>> list1=[10,20,30,40,50]
		>>> len(list1)
		5

list()	Creates an empty list if no argument is passed
	Creates a list if a sequence is passed as
	an argument
		>>> list1 = list()
		>>> list1
		[ ]
		>>> str1 = 'aeiou'
		>>> list1 = list(str1)
		>>> list1
		['a', 'e', 'i', 'o', 'u']

append()	Appends a single element passed as an argument at the end of the list
	
	The single element can also be a list
		>>> list1 = [10,20,30,40]
		>>> list1.append(50)
		>>> list1
		[10, 20, 30, 40, 50]
		>>> list1 = [10,20,30,40]
		>>> list1.append([50,60])
		>>> list1
		[10, 20, 30, 40, [50, 60]]

extend()	Appends each element of the list passed as argument to the end of the given list	>>> list1 = [10,20,30]
		>>> list2 = [40,50]
		>>> list1.extend(list2)
		>>> list1
		[10, 20, 30, 40, 50]

insert()	Inserts an element at a particular index in the list	>>> list1=[10,20,30,40,50]
		>>> list1.insert(2,25)
		>>> list1
		[10, 20, 25, 30, 40, 50]
		>>> list1.insert(0,5)
		>>> list1
		[5, 10, 20, 25, 30, 40, 50]

count()	Returns the number of times a given element appears in the list	>>> list1=[10,20,30,10,40,10]
		>>> list1.count(10)
		3
		>>> list1.count(90)
		0

index()	Returns index of the first occurrence of the element in the list. If the element is not present, ValueError is generated	>>> list1 =
		[10,20,30,20,40,10] >>>
		list1.index(20)
		1
		>>> list1.index(90)
		ValueError: 90 is not in list

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]

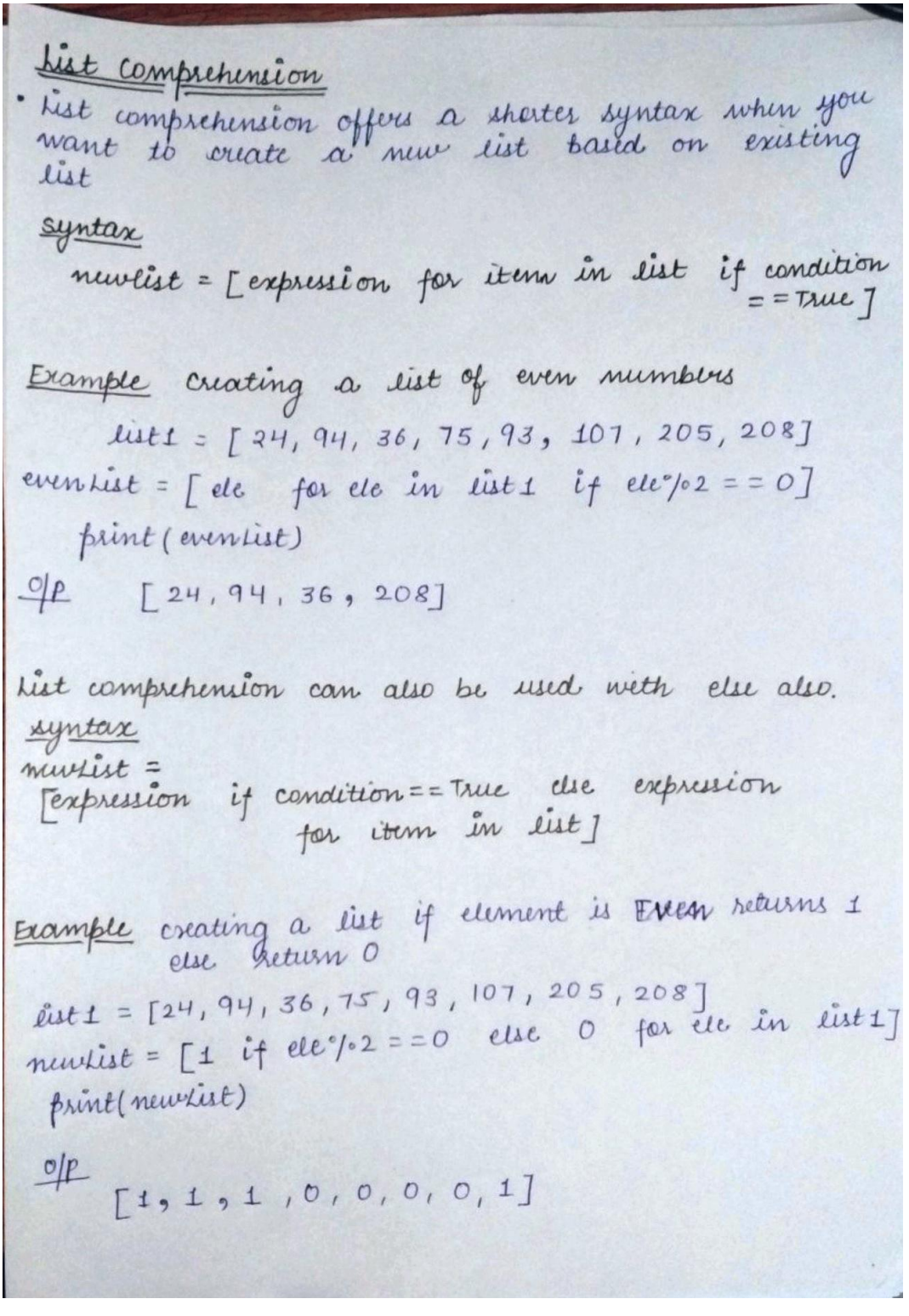
reverse()	Reverses the order of elements in the given list	>>> list1=[34,66,12,89,28,99]
		>>> list1.reverse()
		>>> list1
		[99, 28, 89, 12, 66, 34]
		
		>>> list1=['Tiger' ,'Zebra' , 'Lion' , 'Cat' ,'Elephant','Dog']
		>>> list1.reverse()
		>>> list1
		['Dog', 'Elephant', 'Cat', 'Lion', 'Zebra', 'Tiger']

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]`