| No. | Questions |
| --- | --- |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

| No. | Questions |
| --- | --- |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

| No. | Questions |
|-----|-----------|
|     | Remove elements in a list after a specific index? |
|     | Remove elements in a list before a specific index? |
|     | Remove elements in a list between 2 indices? |
|     | How would you convert a list to an array? |

| No. | Questions |
|-----|-----------|

**Ques. What is List?**

- Lists are used to store multiple items in a single variable.
- List items are **ordered**, **changeable**, and **allow duplicate** values.
- In Python lists are written with **square brackets[]**, separated by commas.
- The list is **changeable**, meaning that we can change, add, and remove items in a list after it has been created.
- list Allow Duplicates values. **Ex:-** list = ["apple", "banana", "cherry", "apple", "cherry"]
- List items can be of any data type. **Ex:-** list = ["abc", 34, True, 40, "male"]

```python
my_list = ["apple", "banana", "cherry", "banana"]
my_list = [1, "Hello", 3.4]
my_list = ["mouse", [8, 4, 6], ['a']]  # A list can also have another list as
                                        # an item. This is called a nested list.

print(my_list)
```

**Ques. List Length**

- use the **len()** function:

```python
thislist = ["apple", "banana", "cherry"]
print(len(thislist)) # Output:- 3
```

**Ques. How to Check if Item Exists?**

```python
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")

Output:- Yes, 'apple' is in the fruits list
```

**Ques. List Comprehension?**

- List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list. **Syntax** newlist = [expression for item in iterable if condition == True]

```python
# Based on a list of fruits, you want a new list, containing only the fruits with
the letter "a" in the name.

# Without list comprehension you will have to write a for statement with a
conditional test inside.
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []
```

```python
    for x in fruits:
      if "a" in x:
        newlist.append(x)

print(newlist)

# With list comprehension you can do all that with only one line of code:
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = [x for x in fruits if "a" in x]

print(newlist)

Output:- ['apple', 'banana', 'mango']
```

## Ques. Access List Items?

```python
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[1])          #Output:- banana
print(thislist[-1])         #Output:- mango
print(thislist[:])          #Output:- ["apple", "banana", "cherry", "orange",
"kiwi", "melon", "mango"]
print(thislist[2:5])        #Output:- ['cherry', 'orange', 'kiwi']
print(thislist[:4])         #Output:- ['apple', 'banana', 'cherry', 'orange']
print(thislist[2:])         #Output:- ['cherry', 'orange', 'kiwi', 'melon',
'mango']
print(thislist[-4:-1])      #Output:- ['orange', 'kiwi', 'melon']
print(thislist[:-1])        #Output:- ['apple', 'banana', 'cherry', 'orange',
'kiwi', 'melon']
print(thislist[-1:2])       #Output:- []
print(thislist[-1:-2])       #Output:- []

# Get the Items at Specified Intervals
list = [9,3,6,4,7,3,1,4]
print(list[::2])    # Output:- [9, 6, 7, 1]
print(list[::-2])   # Output:- [4, 3, 4, 3]
print(list[::-1])   # Output:- [4, 1, 3, 7, 4, 6, 3, 9]  #reverse the List using
slice
print(list[4::])        # output: - [5, 6, 7, 8, 9]
print(list[:4:])        # output: - [1, 2, 3, 4]

# Accessing elements from a multi-dimensional list
List = [['my', 'name'], ['is'],['mohit','saxena']]
print(List[0][1]) # Output:- name
print(List[1][0]) # Output:- is
```

**Ques. Add List Items?**

- **Append method:-** add an item to the **end of the list**, use the **append()** method.

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist) # Output:- ['apple', 'banana', 'cherry', 'orange']
```

- **Insert method:-** The **insert()** method inserts an item at the *specified index*.

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist) #Output:- ['apple', 'orange', 'banana', 'cherry']
```

- **Extend method:-** The **extend()** method does not have to append lists, you can **add any iterable** object (list, tuples, sets, dictionaries etc.).

```
thislist = ["apple", "banana", "cherry"]
sets = ("mango", "pineapple", "papaya")

thislist.extend(sets)
print(thislist) #Output:- ['apple', 'banana', 'cherry', 'mango', 'pineapple',
'papaya']
```

**Ques. Change or Update List Items?**

- **Change Item Value:-** To change the value of a specific item, refer to the index number.

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"

print(thislist)     # Output:- ['apple', 'blackcurrant', 'cherry']
```

- **Change a Range of Item Values:-** To change the value of items within a **specific range**, define a list with the new values, and refer to the range of index numbers where you want to insert the new values.
- 1 se 2 wale range ke element hut jaynge

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3] = ["blackcurrant", "watermelon"]
print(thislist)     # Output:- ['apple', 'blackcurrant', 'watermelon', 'orange',
'kiwi', 'mango']
```

**Ques. Remove List Items?**

```python
# * Remove **Specified Item** from the List using remove() method.
# remove() method the first instance of a matching object.
thislist = ["apple", "banana", "cherry", "banana"]
thislist.remove("banana")
print(thislist)      # Output:- ['apple', 'cherry', "banana"]



# If item not exist in remove method then show the error
thislist.remove("banana1")
print(thislist) # Output:- error item not in the list



# using **pop() method:-** The pop() method **removes the specified index**.
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)      # Output:- ['apple', 'cherry']



# pop() method without index:- if we do not specify the index, the pop() method
**removes the last item**.
thislist.pop()
print(thislist)      # Output:- ['apple', 'banana']



# **del() method:-** The del keyword also **removes the specified index.**
del thislist[0]
print(thislist)      # Output:- ['banana', 'cherry']



# The **del** keyword can also **delete the list completely.**
del thislist
print(thislist) #this will cause an error because you have succsesfully deleted
"thislist".



#  **clear() method:-** The clear() method **empties the list.**
thislist.clear()
print(thislist)      # Output:- []
```

**Ques. Copy Lists?**

- You cannot copy a list simply by typing list2 = list1, because: list2 will only be a reference to list1, and changes made in list1 will automatically also be made in list2.
- So Two method of the copy below.
- **copy() method**

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)

Output:- ['apple', 'banana', 'cherry']
```

- **list() method**

```
thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)

Output:- ['apple', 'banana', 'cherry']
```

**Ques. Join Lists?**

- using the **plas(+)** operator.

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)    # Output:- ['a', 'b', 'c', 1, 2, 3]
```

- Using **append** method all the items from list2 into list1, one by one.

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

for x in list2:
  list1.append(x)

print(list1)    # Output:- ['a', 'b', 'c', 1, 2, 3]
```

- The **extend()** method adds the specified list elements (or any iterable) to the end of the current list.

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list1.extend(list2)
print(list1)    # Output:-['a', 'b', 'c', 1, 2, 3]
```

**Ques. Loop Lists?**

```
# Loop Through a List
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)

Output:-
apple
banana
cherry
-----------------------------------------------------------------------------

# Loop Through the Index Numbers
# You can also loop through the list items by referring to their index number.
#Use the range() and len() functions to create a suitable iterable.
thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
  print(thislist[i])

Output:-
 apple
 banana
 cherry
```

- Using a **While Loop**

```
thislist = ["apple", "banana", "cherry"]
i = 0
while i < len(thislist):
  print(thislist[i])
  i = i + 1

Output:-
apple
banana
cherry
```

- Looping Using **List Comprehension**

```
thislist = ["apple", "banana", "cherry"]
[print(x) for x in thislist]

Output:-
apple
banana
cherry
```

**Ques. Sort Lists?**

- Case Insensitive Sort:- By default the **sort()** method is case sensitive, resulting in all capital letters being sorted before lower case letters.

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.sort()
print(thislist)

Output:- ['Kiwi', 'Orange', 'banana', 'cherry']

# Example2:-
thislist = [100, 50, 65, 82, 23]
thislist.sort()
print(thislist)

Output:- [23, 50, 65, 82, 100]
```

- So if you want a **case-insensitive sort** function, use str.lower as a key function.

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.sort(key = str.lower)
print(thislist)

Output:- ['banana', 'cherry', 'Kiwi', 'Orange']
```

- **Sort Descending:-** To sort descending, use the keyword argument reverse = True.

```
thislist = ["orange", "mango", "Kiwi", "Pineapple", "banana"]
thislist.sort(reverse = True)
print(thislist)

Output:- ['orange', 'mango', 'banana', 'Pineapple', 'Kiwi']
```

- Reverse Order:- The **reverse() method** reverses the current sorting order of the elements.

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.reverse()
print(thislist)

Output:- ['cherry', 'Kiwi', 'Orange', 'banana']
```

# List Methods

| Method | Description |
| --- | --- |
| append() | Adds an element at the end of the list |
| insert() | Adds an element at the specified position |
| extend() | Add the elements of a list (or any iterable), to the end of the current list |
| copy() | Returns a copy of the list |
| count() | Returns the number of elements with the specified value |
| index() | Returns the index of the first element with the specified value |
| pop() | Removes the element at the specified position |
| remove() | Removes the item with the specified value |
| clear() | Removes all the elements from the list |
| reverse() | Reverses the order of the list |
| sort() | Sorts the list |

- **append():-** Adds an element at the end of the list

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)

Output:- ['apple', 'banana', 'cherry', 'orange']


# Example2:-
a = ["apple", "banana", "cherry"]
b = ["Ford", "BMW", "Volvo"]
a.append(b)
print(a)

Output:- ['apple', 'banana', 'cherry', ["Ford", "BMW", "Volvo"]]
```

- **extend():-** Add the elements of a list (or any iterable), to the end of the current list

```
fruits = ['apple', 'banana', 'cherry']
cars = ['Ford', 'BMW', 'Volvo']
fruits.extend(cars)
print(fruits)

output:- ['apple', 'banana', 'cherry', 'Ford', 'BMW', 'Volvo']
```

- **insert():-** Adds an element at the specified position

```
fruits = ['apple', 'banana', 'cherry']
fruits.insert(1, "orange")
print(fruits)

Output:- ['apple', 'orange', 'banana', 'cherry']
```

- **copy():-** Returns a copy of the list

```
fruits = ["apple", "banana", "cherry"]
x = fruits.copy()
print(x)

Output:- ['apple', 'banana', 'cherry']
```

- **count():-** Returns the number of elements with the specified value

```
fruits = [1, 4, 2, 9, 7, 8, 9, 3, 1]
x = fruits.count(9)
print(x)

Output:- 2
```

- **index():-** Returns the index of the first element with the specified value

```
fruits = [4, 55, 64, 32, 16, 32]
x = fruits.index(32)
print(x)

Output:- 3
```

- **pop():-** Removes the element at the specified position
- if you do not specify the index, the **pop()** method removes the last item.

```
fruits = ['apple', 'banana', 'cherry']
x = fruits.pop(1)
print(x)
print(fruits)

output:-
banana
['apple', 'cherry']
```

- **remove():-** Removes the item with the specified value

```
fruits = ['apple', 'banana', 'cherry']
fruits.remove("banana")
print(fruits)

Output:- ['apple', 'cherry']
```

- **clear():-** Removes all the elements from the list

```
fruits = ["apple", "banana", "cherry"]
fruits.clear()
print(fruits)

Output:- []
```

- **reverse():-** Reverses the order of the list

```
fruits = ['apple', 'banana', 'cherry']
fruits.reverse()
print(fruits)

Output:- ['cherry', 'banana', 'apple']
```

- **sort():-** Sorts the list

```
cars = ['Ford', 'BMW', 'Volvo']
cars.sort()
print(cars)

Output:- ['BMW', 'Ford', 'Volvo']

# Parameter Values in sort()
 1. reverse:- optional. reverse=True will sort the list descending. Default is
reverse=False
 2. key Optional. A function to specify the sorting criteria(s)

Example:-
cars = ['Ford', 'BMW', 'Volvo']
cars.sort(reverse=True)
print(cars)

Output:- ['Volvo', 'Ford', 'BMW']

Example2:-
```

```python
def myFunc(e):
  return len(e)

cars = ['Ford', 'Mitsubishi', 'BMW', 'VW']
cars.sort(key=myFunc)
print(cars)

Output:- ['VW', 'BMW', 'Ford', 'Mitsubishi']

Example3:-
def myFunc(e):
  return len(e)

cars = ['Ford', 'Mitsubishi', 'BMW', 'VW']
cars.sort(reverse=True, key=myFunc)
print(cars)

Output:- ['Mitsubishi', 'Ford'', 'BMW', 'VW']

Example4:-
def myFunc(e):
  return e['year']

cars = [
  {'car': 'Ford', 'year': 2005},
  {'car': 'Mitsubishi', 'year': 2000},
  {'car': 'BMW', 'year': 2019},
  {'car': 'VW', 'year': 2011}
]

cars.sort(key=myFunc)

print(cars)

Output:-
[{'car': 'Mitsubishi', 'year': 2000}, {'car': 'Ford', 'year': 2005}, {'car': 'VW',
'year': 2011}, {'car': 'BMW', 'year': 2019}]
```

## Ques. Difference between List and Tuples in Python?

| List | Tuples |
| --- | --- |
| List is mutable. i.e they can be edited. | Tuple is immutable. (tuples are lists which can't be edited). |
| List iteration is slower and is time consuming. | Tuple iteration is faster. |
| List is useful for insertion and deletion operations. | Tuple is useful for readonly operations like accessing elements. |
| List has a large memory. | Tuple has a small memory. |
| List is stored in two blocks of memory (One is fixed sized and the other is variable sized for storing data) | Tuple is stored in a single block of memory. |
| List provides many in-built methods. | Tuples have less in-built methods. |
| List operations are more error prone | Tuples operations are safe. |
| A list has data stored in square brackets [] brackets. For example, list_1 = [10, 'Chelsea', 20] | A tuple has data stored in parantheses () brackets. For example, tup_1 = (10, 'Chelsea' , 20) |

**flatten list**

- Flattening a list in Python refers to the process of converting a nested list (a list containing other lists as elements) into a single, one-dimensional list. This operation removes the nested structure and combines all elements into a single sequence.
- Several methods can achieve this, including list comprehensions, loops, and recursion.

```python
# ex1
def flat(lis):
    flatList = []
    # Iterate with outer list
    for element in lis:
        if type(element) is list:
            # Check if type is list than iterate through the sublist
            for item in element:
                flatList.append(item)
        else:
            flatList.append(element)
    return flatList


lis = [1, [11, 22, 33, 44], [55, 66, 77], [88, 99, 100]]
print('List', lis)
print('Flat List', flat(lis))
Output:-
List [1, [11, 22, 33, 44], [55, 66, 77], [88, 99, 100]]
Flat List [1, 11, 22, 33, 44, 55, 66, 77, 88, 99, 100]

#ex2:-
def flatten_list(nested_list):
    flattened = []
    for item in nested_list:
        if isinstance(item, list):
            flattened.extend(flatten_list(item))
        else:
            flattened.append(item)
    return flattened

nested_list = [1, [2, 3], [4, [5, 6]]]
flattened_list = flatten_list(nested_list)
print(flattened_list)  # Output: [1, 2, 3, 4, 5, 6]
```

# List Questions

**Find the Length of a List?**

- Using **len() Function** or **length_hint** function

```python
# Using lentgh function
li = [10, 20, 30]
n = len(li)
print("The length of list is: ", n)
Output:- The length of list is:  3

# Using length_hint Function
from operator import length_hint
test_list = [1, 4, 5, 7, 8]
list_len_hint = length_hint(test_list)
print("Length of list using length_hint() is : " + str(list_len_hint))
Output:- Length of list using length_hint() is : 5

# using for loop
test_list = [1, 4, 5, 7, 8]
counter = 0
for i in test_list:
    counter = counter + 1
print("Length of list using naive method is : " + str(counter))
Output:- Length of list using naive method is : 5
```

**Ques. find the single number(unique element) of the list?**

```python
mylist = [1,2,2,3,3,4,5,5,5,6,6,6,6]
new_list = []
for num in mylist:
    if(mylist.count(num) == 1):
        new_list.append(num)

print(new_list) # Output:-[1, 4]

# Using Comp
print([num for num in mylist if mylist.count(num)==1]) # Output:-[1, 4]
```

## Ques. Interchange first and last elements in a list?

```python
# Without temp varibale
-----------------------
list = [12, 35, 9, 56, 24]
list[0] = list[-1]
list[-1] = list[0]
print(list) # Output:- [24, 35, 9, 56, 24]

# With temp variable
--------------------
list = [12, 35, 9, 56, 24]
length = len(list)
temp = list[0]
list[0] = list[length - 1]
list[length - 1] = temp
print(list) # Output:- [24, 35, 9, 56, 12]

# Using comma function
----------------------
def swapList(newList):
    newList[0], newList[-1] = newList[-1], newList[0]
    return newList

# Driver code
newList = [12, 35, 9, 56, 24]
print(swapList(newList))    # Output:- [24, 35, 9, 56, 12]

# Using * operand.
------------------
list = [1, 2, 3, 4]

a, *b, c = list

print(a)
print(b)
print(c)

Output:-
1
[2, 3]
4
# Using * operand 2 approch.
def swapList(list):
    start, *middle, end = list
    list = [end, *middle, start]
    return list

newList = [12, 35, 9, 56, 24]
print(swapList(newList))    # Output:- [24, 35, 9, 56, 12]
```

## Swap Two Elements in a List?

- using **comma** assignment

```python
def swapPositions(list, pos1, pos2):

    list[pos1], list[pos2] = list[pos2], list[pos1]
    return list

# Driver function
List = [23, 65, 19, 90]
pos1, pos2  = 1, 3

print(swapPositions(List, pos1-1, pos2-1))  # Output:- [19, 65, 23, 90]
```

- Using **temp** variable

```python
def swapPositions(lis, pos1, pos2):
    temp=lis[pos1]
    lis[pos1]=lis[pos2]
    lis[pos2]=temp
    return lis
# Driver function
List = [23, 65, 19, 90]
pos1, pos2 = 1, 3

print(swapPositions(List, pos1-1, pos2-1))  # Output:- [19, 65, 23, 90]
```

- Using **enumerate**

```python
def swapPositions(lis, pos1, pos2):
    for i, x in enumerate(lis):
        if i == pos1:
            elem1 = x
        if i == pos2:
            elem2 = x
    lis[pos1] = elem2
    lis[pos2] = elem1
    return lis

List = [23, 65, 19, 90]
pos1, pos2 = 1, 3
print(swapPositions(List, pos1-1, pos2-1))  # Output:- [19, 65, 23, 90]
```

## Ques. find the even number from the list?

```python
numberList = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ansList = []
unique_lst = []
for num in numberList:
    for i in range(2, num):
        if num % i == 0:
            break
        else:
            ansList.append(num)
# ansList = list(dict.fromkeys(ansList)) # remove duplicate item using dict method
# ansList = [*set(ansList)] # remove duplicate item convert into set.
for ele in ansList:
    if ele not in unique_lst:
        unique_lst.append(ele)
print(unique_lst)
```

## Ques. Print duplicate list, Find Even Or Odd Number?

```python
list = [9,3,6,4,7,3,1,4]
duplicate = []
even = []
odd = []
for i in list:
    if list.count(i) > 1 and i not in duplicate:
        duplicate.append(i)
    elif i%2 == 0:
        even.append(i)
    else:
        odd.append(i)
print(duplicate)    # Output:- [3, 4]
print(even)     # Output:- [6, 4]
print(odd)       # Output:- [9, 7, 3, 1]

# Using List comprehension
[duplicate.append(i) for i in list if list.count(i) > 1 and i not in duplicate]
print(duplicate)
```

Find the duplicate element from list?

```
list = [9,3,6,4,7,3,1,4]
duplicate = []
for i in list:
    if list.count(i) > 1 and i not in duplicate:
        duplicate.append(i)

print(duplicate)

Output:-
[3,4]
```

```
l=[1,2,3,4,5,2,3,4,7,9,5]
l1=[]
for i in l:
    if i not in l1:
        l1.append(i)
    else:
        print(i,end=' ')

Output:- 2 3 4 5
```

Ques. Remove duplicate item from list using List comprehension?

```
lstnum = [12, 36, 56, 36, 36, 50, 56, 12]
unique_lst = []

[unique_lst.append(ele) for ele in lstnum if ele not in unique_lst]
print ("unique elements list  : " ,unique_lst)
```

Ques. Convert a list into string?

```
list = ['my','name','is','Mohit','Saxena']
listtostring = ' '.join(list)
print('list after shuffling =',listtostring)

Output:-
list after shuffling = my name is Mohit Saxena
```

Ques. Write a program to print a list in reverse order?

- **using slice method**

```python
def revlist(list):
    return list[::-1]

list = [24,55,78,64,25,12,22,11,1,2,44]
print(revlist(list))

Output:- [44, 2, 1, 11, 22, 12, 25, 64, 78, 55, 24]
```

- **Using For loop**

```python
list1 = [1, 2, 4, 5, 8, 9]
list2 = []
for item in list1:
    list2.insert(0, item)
print(list2)

Output:- [9, 8, 5, 4, 2, 1]
```

Ques. even values from a list using list comprehension?

```python
# normal function
lstnum = [12, 18, 14,17,15,6]
evenNum = []
for ele in lstnum:
    if ele%2==0:
        evenNum.append(ele)
print(evenNum)

# comprehension
evenNum1 = [ele for ele in lstnum if ele%2==0]
print(evenNum1)

Output:- [12, 18, 14, 6]
```

Ques. find the max, min number from the list user input?

```python
number = int(input('enter the number of items in list '))
list = []
for num in range(number):
    item = int(input('Entered number '))
    list.append(item)
print('entered list=', list)
print('Max Number= ', max(list))
print('min number= ', min(list))

Output:-
```

```
enter the number of items in list 5
Entered number 6
Entered number 4
Entered number 15
Entered number 85
Entered number 5
entered list= [6, 4, 15, 85, 5]
Max Number=  85
min number=   4
```

Ques. find the sum of list elements?

```
num = [12, 36, 56, 36, 36, 50, 56, 12]
sum = 0
for ele in range(len(num)):
    sum = sum + num[ele]
print(sum)

Output:- 294
```

Ques. Generate a number list between two ranges?

```
listnum = list(range(1, 7))
print ("list between two range : " ,listnum)

Output:- list between two range :  [1, 2, 3, 4, 5, 6]
```

Ques. Remove elements in a list after a specific index?

```
li = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,10]
remove_item = li[:10]
print(remove_item)
#=> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Ques. Remove elements in a list before a specific index?

```
li = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,10]
li[15:]
#=> [16, 17, 18, 19, 10]
```

Ques. Remove elements in a list between 2 indices?

```
li = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,10]
li[12:17]
#=> [13, 14, 15, 16, 17]
```

## Ques. Count the occurrence of a specific object in a list?

- The count() method returns the number of occurrences of a specific object.

```
pets = ['dog','cat','fish','fish','cat']
index = pets.count('fish')
print(index)

Output:- 2
```

## Ques. Remove the negative index from the list?

```
lstnum = [-5, 27, 1000, -4, 0, -80,56,-67]
# //Removing negative values
posNum = []
for item in lstnum:
    if item >= 0:
        posNum.append(item)
print(posNum)

res_lst = [item for item in lstnum if item >= 0]
print('list after removing negative values =',res_lst)

Output:-
[27, 1000, 0, 56]
```

## Ques. What is the difference between an array and a list?

| #  | List | Array |
|----|------|-------|
| 1  | It Contains elements of different data types | It Contains elements of same data types |
| 2  | Cannot handle arithmetic operations | Can handle arithmetic operations |
| 3  | We can print the entire list without the help of an explicit loop | To print or access array elements, we will require an explicit loop |
| 4  | It consumes a large memory | It is a more compact in memory size comparatively list. |

## Split the strings and store into a list

```
string = input("Enter string: ")
lst = string.split()
print('The list is:', lst)

Output:-
Enter string: my name is mohit saxena
The list is: ['my', 'name', 'is', 'mohit', 'saxena']
```

## Ques. list() Constructor?

It is also possible to use the **list()** constructor to make a new list.

```
thislist = list(("apple", "banana", "cherry"))
print(thislist)

Output:- ['apple', 'banana', 'cherry']
```

## Ques. Multiply a Python List by a Number Using a for loop?

```
numbers = [1, 2, 3, 4, 5]
multiplied = []
for number in numbers:
    multiplied.append(number * 2)
print(multiplied)
Output:- [2, 4, 6, 8, 10]
```

## Ques. Multiply a Python List by a Number Using a list comprehension?

```
numbers = [1, 2, 3, 4, 5]
multiplied = [number * 2 for number in numbers]
print(multiplied)
Output:- [2, 4, 6, 8, 10]
```

## Ques. **Convert a list into a tuple?**

- Using **tuple()** builtin function

```
list = [1,2,3,4]
result = tuple(list)
print(type(result))

Output:- <class 'tuple'>
```

- Using **loop** inside the tuple

```
sample_list = ['Compile', 'With', 'Favtutor']
tuple1 = tuple(i for i in sample_list)
print(tuple1)


Output:- ('Compile', 'With', 'Favtutor')
```

- **Unpack** list inside the parenthesis

```
sample_list = ['Compile', 'With', 'Favtutor']

#unpack list items and form tuple
tuple1 = (*sample_list,)

print(tuple1)
print(type(tuple1))

Output:-
('Compile', 'With', 'Favtutor')
<class 'tuple'>
```

## Ques. Check if a list contains an element?

```
li = [1,2,3,'a','b','c']
print('a' in li)

Output:- True
```

## Ques. How to flatten a list of lists with a list comprehension?

```
def flatten_list(d_list):
    flat_list = []
    # Iterate through the outer list
    for element in d_list:
        if type(element) is list:
            # If the element is of type list, iterate through the sublist
            for item in element:
                flat_list.append(item)
        else:
            flat_list.append(element)
    return flat_list

nested_list = [[1, 2, 3, 4], [5, 6, 7], [8, 9, 10]]
print('Original List', nested_list)
print('Transformed Flat List', flatten_list(nested_list))
```

```
-----------------------------------------------------------------
listnum = [[5,6,7,'C#'], ['C++',2,3]]
flatten_list = [ele for sublist in listnum for ele in sublist]
print('flatten list =',flatten_list)

Output:-
Original List [[1, 2, 3, 4], [5, 6, 7], [8, 9, 10]]
Transformed Flat List [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

**Ques. How to Intersect two list?**

```
listnum = ['C++',2,3,6,7,5,'C#']
listnum1 = ['C++',5,6,7,'C#']
intersect_res= []
for ele in listnum:
    if ele in listnum1:
        intersect_res.append(ele)
print(intersect_res)

# Using comprehension
intersect_res = [item for item in listnum if item in listnum1]

print('intersect of two list =',intersect_res)  # output:- ['C++', 6, 7, 5, 'C#']
```

**Ques. get the difference between two List using comprehension?**

```
lstnum = [15, 78, 4]
lstnum1 = [80, 4, 89]
diffra = []
for num in lstnum:
    if num not in lstnum1:
        diffra.append(num)

print(diffra)   # Output:- Output:- [15, 78]
```

**Ques. How to iterate over 2+ lists at the same time?**

```
name = ['Snowball', 'Chewy', 'Bubbles', 'Gruff']
animal = ['Cat', 'Dog', 'Fish', 'Goat']
age = [1, 2, 2, 6]
z = zip(name, animal, age)
for name,animal,age in z:
    print("%s the %s is %d" % (name, animal, age))

Output:-
Snowball the Cat is 1
```

```
Chewy the Dog is 2
Bubbles the Fish is 2
Gruff the Goat is 6
```

## Ques. Combine 2 lists into a list of tuples with the zip function?

```
name = ['Snowball', 'Chewy', 'Bubbles', 'Gruff']
animal = ['Cat', 'Dog', 'Fish', 'Goat']
print(list(zip(name,animal)))

Output:- [('Snowball', 'Cat'), ('Chewy', 'Dog'), ('Bubbles', 'Fish'), ('Gruff',
'Goat')]
```

## Ques. How to Zip two lists

- Using map() + **add**

```
test_list1 = [[1, 3], [4, 5], [5, 6]]
test_list2 = [[7, 9], [3, 2], [3, 10]]

print("The original list 1 is : " + str(test_list1))
print("The original list 2 is : " + str(test_list2))

res = list(map(list.__add__, test_list1, test_list2))

print("The modified zipped list is : " + str(res))

Output:-
The original list 1 is : [[1, 3], [4, 5], [5, 6]]
The original list 2 is : [[7, 9], [3, 2], [3, 10]]
The modified zipped list is : [[1, 3, 7, 9], [4, 5, 3, 2], [5, 6, 3, 10]]
```

## Ques. List Sorting in descending order?

```
list = [24,55,78,64,25,12,22,11,1,2,44]
list.sort(reverse = True)
print(list)

# 2nd Option Using For Loop
# list = [24,55,78,64,25,12,22,11,1,2,44]
list = []
intlistTot = int(input("Total Number of List Items to Sort = "))
for i in range(1, intlistTot + 1):
    intlistvalue = int(input("Please enter the %d List Item = "  %i))
    list.append(intlistvalue)

for i in range(len(list)):
```

```
        for j in range(i + 1, len(list)):
            if(list[i] < list[j]):
                temp = list[i]
                list[i] = list[j]
                list[j] = temp

print(list)

Output:- [78, 64, 55, 44, 25, 24, 22, 12, 11, 2, 1]
```

## Ques . sort the list on the basis of length?

```
def Sorting(lst):
    lst2 = sorted(lst, key=len)
    return lst2

lst = ["rohan", "amy", "sapna", "muhammad", "aakash", "raunak", "chinmoy"]
print(Sorting(lst))

Output:- ['amy', 'rohan', 'sapna', 'aakash', 'raunak', 'chinmoy', 'muhammad']
```

## Ques. Check if a list contains an element?

- The in operator will return True if a specific element is in a list.

```
li = [1,2,3,'a','b','c']
'a' in li

Output:- True
```

## Ques. Find the index of the 1st matching element?

- you want to find the first "apple" in a list of fruit. Use the **index()** method.

```
fruit = ['pear', 'orange', 'apple', 'grapefruit', 'apple', 'pear']
a = fruit.index('apple') #=> 2
b = fruit.index('pear') #=> 0
```

## Ques. Iterate over both the values in a list and their indices?

- enumerate() adds a counter to the list passed as an argument.

```
grocery_list = ['flour','cheese','carrots']
for id,val in enumerate(grocery_list):
```

```
        print("%s: %s" % (id, val))

Output:-
0: flour
1: cheese
2: carrots
```

## Ques. How to manipulate every element in a list with list comprehension?

```python
# using comprehension
li = [0,25,50,100]
b = [i+1 for i in li]
print(b)

# Using for loop
for i in li:
    a = i+1;
    print(a)

Output:-
[1, 26, 51, 101]
1
26
51
101
```

## Ques. Remove negative values from a list with the filter function?

```python
def remove_negatives(x):
    return True if x >= 0 else False

a = [-10, 27, 1000, -1, 0, -30]
b  = [x for x in filter(remove_negatives, a)]
print(b)

# Using for Comprehension
res = [ele for ele in test_list if ele > 0]
print("List after filtering : " + str(res))
```

### Replace elements in String list?

```python
s = ["Tutor","joes","Computer","Education"]
print("Before Swap :",s)
res = [sub.replace("joes","Joe's").replace("Computer",
"Software").replace("Education", "Solutions") for sub in s]
print ("After Swap : ",res)
```

```
Output:-
Before Swap : ['Tutor', 'joes', 'Computer', 'Education']
After Swap :  ['Tutor', "Joe's", 'Software', 'Solutions']
```

## Ques. How would you convert a list to an array?

- Python list is a linear data structure that can hold heterogeneous elements(hold items of different data types). Python does not have a built-in array data type. If you want to create an array in Python, then use the numpy library.
- To install **numpy** in your system, type the following command.

```
python3 -m pip install numpy
```

- **1. Using numpy.array()**

```python
import numpy as np

elon_list = [11, 21, 19, 18, 29]
elon_array = np.array(elon_list)

print(elon_array)
print(type(elon_array))

Output:-
[11 21 19 18 29]
<class 'numpy.ndarray'>

# 2. Using numpy.asarray()
import numpy as np

elon_list = [11, 21, 19, 18, 29]
elon_array = np.asarray(elon_list)

print(elon_array)
print(type(elon_array))

Output:- [11 21 19 18 29]
<class 'numpy.ndarray'>
```