Q1. Can you create a programme or function that employs both positive and negative indexing? Is there any repercussion if you do so?

Ans:

list **=** [10,20,30,40,50,60,70,80,90,100]

**def** 2way\_index(in\_list, index\_position):

**return** in\_list[index\_position]

print('Positive Indexing ->',2way\_index(list,5))

print('Negative Indexing ->',2way\_index(list,**-**1))

Q2. What is the most effective way of starting with 1,000 elements in a Python list? Assume that all elements should be set to the same value.

Ans: the effective way of starting with 1000 elements in python list is list comprehension

starting\_list = [ 1 for i in range(1001)]

print(starting\_list)

Q3. How do you slice a list to get any other part while missing the rest? (For example, suppose you want to make a new list with the elements first, third, fifth, seventh, and so on.)

Ans:

list **=** [x for x in range(1,15)]

print(f’list -> {list}')

s\_list **=** list[::2]

print(f's\_list -> {s\_list}')

Q4. Explain the distinctions between indexing and slicing.

Ans:

list **=** [x for x in range(1,15)]

print(f'list -> {list}')

print(f'indexing -> {list[1], list[5]}')

print(f'slicing -> {my\_list[1:10]}')

Q5. What happens if one of the slicing expression's indexes is out of range?

Ans: If start index is out of range then it will return empty list

list = [x for x in range(1,15)]

list = [x for x in range(1,15)]

print(f'list -> {list}')

print(f'Case #1 -> {list[20:]}')

print(f'Case #2 -> {list[10:100]}')

Q6. If you pass a list to a function, and if you want the function to be able to change the values of the list—so that the list is different after the function returns—what action should you avoid?

Ans:use return statement

list = [1,2,3,4,5,6]

def modify\_list(in\_list):

in\_list.append(200)

return in\_list

print(modify\_list(list))

Q7. What is the concept of an unbalanced matrix?

Ans: Number of rows is not same as number of column is called unbalanced matrix

Q8. Why is it necessary to use either list comprehension or a loop to create arbitrarily large matrices?

Ans: List comprehensions or loops make it easy to create large matrices. It also helps in implementing and avoiding manual errors. It also makes your code easier to read. Manual feed time is also greatly reduced.