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

Ans.

Yes, you can use both positive and negative indexing in a Python program or function. Positive indexing starts from 0 and goes up to n-1, while negative indexing starts from -1 and goes down to -n. The only repercussion is that it can make the code harder to read and maintain.

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 most effective way to create a list with 1,000 elements set to the same value is to use a list comprehension. For example, you can create a list of 1,000 zeros as follows:

my\_list = [0] \* 1000

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.

You can use slice notation to achieve this. The slice notation takes three arguments: start, stop, and step. To get every other element starting from the first element, you can use the following slice notation:

my\_list[::2]

This will return a new list with the elements first, third, fifth, seventh, and so on.

Q4. Explain the distinctions between indexing and slicing.

Ans.

Indexing is the process of retrieving a single element from a list using its index position. Slicing, on the other hand, is the process of extracting a portion of a list using a range of index positions. Indexing returns a single element, while slicing returns a new list.

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

Ans.

If one of the slicing expression's indexes is out of range, Python will not raise an error but instead will return an empty list. For example, if you try to slice a list with an index that is greater than its length, you will get an empty list:

my\_list = [1, 2, 3]

print(my\_list[4:]) # Output: []

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.

If you want a function to modify a list and have the changes reflected outside the function, you should avoid creating a new list inside the function. Instead, you should modify the existing list in place using its index positions or methods.

Q7. What is the concept of an unbalanced matrix?

Ans.

An unbalanced matrix is a two-dimensional array in which each row can have a different number of columns. In other words, the number of columns in each row is not the same.

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

Ans.

It is necessary to use either list comprehension or a loop to create arbitrarily large matrices because Python does not provide a built-in data type for matrices. Lists are the closest data type to a matrix in Python, and they can be nested to create two-dimensional arrays. List comprehension or loops can be used to initialize the elements of the list.