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

Answer: Yes we can access list elements in python in a positive indexing manner starting from 0 and for negative index consideration , we can start from -1 which means starting from ending index and continue back to the front of the list start point.

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.

Answer: l = [i for i in range(1,1001)] , will definitely help in assigning different values from 1 to 1000 in the fastest way possible.

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.)

Answer: new\_list = [a[i] for i in range(a) if i%2 != 0] where a is an older list and we are creating a new list where index is odd and their elements are selected in the new list.

Q4. Explain the distinctions between indexing and slicing.

Answer: Indexing is simply accessing an element from an iterable list without making any changes.

Slicing a list is getting a subset of elements from an iterable based on their indices.

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

Answer: We will get IndexError as we are trying to access an index which is not present in list at all.

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?

Answer: We should not use a single hardcoded value to replace some of the values . Instead, we should be automate the assigning different values for various indexes depending on the situation.

Q7. What is the concept of an unbalanced matrix?

Answer: Matrix is situation where you want to minimize the time taken for a job execution and we use Hungarian method for doing so.

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

Answer: We use list comprehension or loops to create large matrices as we matrix is nothing but 2D lists in any programming language so, when we matrices and that too big ones, we simply consider manipulations of 1D list at the lowest level.