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

In Python, you can start indexing from the end of an iterable. This is known as negative indexing. This means you can use both positive and negative indexes to access iterables.

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.

arr = []

for i in range(1000):

arr.append(0)

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.)

Q4. Explain the distinctions between indexing and slicing.

“Indexing” means referring to an element of an iterable by its position within the iterable. “Slicing” means 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?

The slicing operation doesn't raise an error if both your start and stop indices are larger than the sequence length

Python will throw an index out of bounds error. However, with slicing it simply returns an empty sequence.

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?

Q7. What is the concept of an unbalanced matrix?

Whenever the cost matrix of an assignment problem is not a square matrix, that is, whenever the number of sources is not equal to the number of destinations, the assignment problem is called an unbalanced assignment problem.

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

List comprehensions apply an **arbitrary expression to** items in an iterable rather than applying function. It provides a compact way of mapping a list into another list by applying a function to each of the elements of the list.