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

Ans: Positive indexing − Increases from 0 to 1. Negative indexing − each traversal moves from tail to head, starting with the last element.

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: Using a for loop and append() ...

Using a while loop with a counter variable. ...

Using list comprehensions. ...

Using the \* operator.

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: Basic usage of slices. [start:stop] ...

Extract from the end with a negative value. Negative values for start and stop. ...

Create slice objects with slice()

Assign values using slices.

Apply slices to a list of lists.

Slices make shallow copy.

Slices for strings and tuples.

Q4. Explain the distinctions between indexing and slicing.

Ans: “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?

Ans: 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?

Ans: def get\_two(func, thing): ... return func(thing), func(thing)

Q7. What is the concept of an unbalanced matrix?

Ans: whenever the number of sources is not equal to the number of destinations.

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

Ans: they use a more optimized internal mechanism for iterating over the collection.