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

Answer 1:

def mix\_indexing(my\_list):

first\_element = my\_list[0]

last\_element = my\_list[-1]

return first\_element, last\_element

my\_list = [1, 2, 3, 4, 5]

result = mix\_indexing(my\_list)

print(result)

**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 2: The most effective way to start with 1,000 elements in a Python list, all set to the same value, is to use a list comprehension:

my\_list = [initial\_value] \* 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.)**

Answer 3: To get every other element starting from the first, you can use slice notation with a step size of 2:

new\_list = original\_list[::2]

**Q4. Explain the distinctions between indexing and slicing.**

Answer 4: Indexing refers to accessing a single element in a list, while slicing involves extracting a portion of the list using a range of indices.

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

Answer 5: If one of the slicing expression's indexes is out of range, Python will not raise an error. It will adjust the index to the nearest valid value. If the start index is greater than the length of the list, an empty list is returned.

**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 6: To allow a function to change the values of a list after it returns, you should avoid reassigning the entire list to a new object within the function. Modifying the existing list in place will reflect the changes outside the function.

**Q7. What is the concept of an unbalanced matrix?**

Answer 7: An unbalanced matrix is a matrix where the number of elements in each row is not the same. In Python, you might represent it using a list of lists where the inner lists have different lengths.

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

Answer 8: List comprehension or loops are necessary to create arbitrarily large matrices because they provide a concise and readable way to generate elements dynamically. Without them, you would need repetitive manual code to initialize each element, making the process cumbersome and error-prone.