**Q1. Can you create a programme or function that employs both positive and negative indexing? Is**

**there any repercussion if you do so?**

**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.**

**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.**

**Q5. What happens if one of the slicing expression&#39;s indexes is out of range?**

**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?**

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

**Solutions**

*1. Yes, it is possible to create a program or function that employs both positive and negative indexing in Python. Negative indexing allows you to access elements from the end of a list, where -1 refers to the last element, -2 refers to the second-to-last element, and so on. Positive indexing starts at 0, where 0 refers to the first element, 1 refers to the second element, and so on. There is no repercussion if you use both positive and negative indexing in the same program or function, but it's important to understand the difference between the two and to use the appropriate indexing in the right context.*

*2. One effective way of starting with 1,000 elements in a Python list and setting all elements to the same value is to use the* ***\**** *operator to repeat a list of a single value 1,000 times. For example:*

***x = [0] \* 1000***

*3. To slice a list and get any other part while missing the rest, you can use slicing with a step value of 2. For example:*

***x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]***

***y = x[::2]***

***print(y)***

***[1, 3, 5, 7, 9]***

*4. Indexing and slicing are two different ways to access elements in a list or an array in Python. Indexing is used to access a single element of a list, while slicing is used to access a sub-list or a range of elements in a list. Indexing is performed by specifying the index of the element you want to access in square brackets (e.g.* ***x[3]****). Slicing is performed by specifying the start and end indices of the sub-list separated by a colon (e.g.* ***x[2:5]****).*

*A5. If one of the slicing expression's indexes is out of range, an* ***IndexError*** *will be raised. For example:*

*x = [1, 2, 3, 4, 5]*

*x[10:15]*

*Traceback (most recent call last):*

*File "<stdin>", line 1, in <module>*

*IndexError: list index out of range*

*6. If you pass a list to a function and want the function to be able to change the values of the list, you should avoid using the* ***list*** *function to create a copy of the list, as this will create a new, separate list that is not connected to the original list. Instead, you can pass the list to the function as a reference, which will allow the function to modify the original list.*

*7. An unbalanced matrix is a matrix that does not have the same number of rows and columns. In other words, it is a matrix with an unequal number of elements in its rows and columns.*

*8. It is necessary to use either list comprehension or a loop to create arbitrarily large matrices because creating a matrix with a large number of elements manually is not feasible. List comprehension and loops allow you to create matrices of any size dynamically and efficiently, based on certain conditions or iteration over a set of elements.*