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 and Negative Scaling works in exactly opposite scaling system.

Therefore, python interpreter will be confused as to which scaling to use.

The indexing will not work

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: Most effective way will be to use generator function to start with the list of 1000 elements

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:

list1 = ['Dog', 'Cat', 'Horse', 'Bear','Lion','tiger','peacock']  
  
list2 = list1[0:len(list1):2]

Q4. Explain the distinctions between indexing and slicing.

Ans:

Indexing returns one item, Slicing returns new list/tuple

For indexing, attempting to use an index that is too large will result in an IndexError. For Slicing, out of range indexes are handled gracefully when used for slicing.

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

Ans: The slicing operation doesn’t raise an error if both your start and stop indices are larger than the sequence length. if you slice an out-of-bounds sequence, it makes perfect sense to return 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:

Python as a design philosophy supports pass by reference, therefore whatever modification is done to the called function’s argument make changes to the original list.

Q7. What is the concept of an unbalanced matrix?

Ans:

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

Ans: We are reducing 3 lines of code into one, which will be instantly recognizable to anyone who understands list comprehensions. Secondly, the second code is faster, as Python will allocate the list’s memory first, before adding the elements to it, instead of having to resize on runtime. It’ll also avoid having to make calls to ‘append’, which may be cheap but add up. Lastly, code using comprehensions is considered more ‘Pythonic’ — better fitting Python’s style guidelines.