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

Ans : 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.

* Ans : Using a for loop and append() ...
* Using a while loop with a counter variable. ...
* Using list comprehensions. ...
* Using the \* operator.

sum() function in Python  
  
Python provides an inbuilt function sum() which sums up the numbers in the list. Syntax: sum(iterable, start) iterable : iterable can be anything list , tuples or dictionaries , but most importantly it should be numbers. start : this start is added to the sum of numbers in the iterable.

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 : The format for list slicing is [start:stop:step]. start is the index of the list where slicing starts. stop is the index of the list where slicing ends. step allows you to select nth item within the range start to stop.

Q4. Explain the distinctions between indexing and slicing.

Ans : What are Indexing and Slicing? Indexing: Indexing is used to obtain individual elements. Slicing: Slicing is used to obtain a sequence of elements. Indexing and Slicing can be be done in Python Sequences types like list, string, tuple, range objects.

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. This is in contrast to simple indexing—if you index an element that is out of bounds, 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?

1. Ans : Use a List and Pass It as an Argument to the Function.
2. Use the tuple() Function in Python.
3. Use the \* Operator in Python.

Python lists have different methods that help you modify a list. This section of the tutorial just goes over various python list methods.

Q7. What is the concept of an unbalanced matrix?

Ans : 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?

Ans : List comprehensions are often not only more readable but also faster than using “for loops.” They can simplify your code, but if you put too much logic inside, they will instead become harder to read and understand.