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

Answer: Python programming language supports negative indexing of arrays, something which is not available in arrays in most other programming languages. This means that the index value of -1 gives the last element, and -2 gives the second last element of an array. The negative indexing starts from where the array ends. index's are used in arrays in all programming languages.we can access the elements of an array by going through their indexs.but no programming language allows us to use negative index value such as -4.python programming language supports negative indexing of arrays, which is not available in arrays in most other programming languages.this means that the negative index value of -1 gives the last element and -2 gives the second last element of an array.the negative indexing starts from where the array ends.this means that the last element of the array is the first element in the negative indexing which is -1.

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: Best and/or fastest way to create lists in python Simple loop with append : my_list = [] for i in range(50): my_list.append(0) Simple loop with += : my_list = [] for i in range(50): my_list += [0] List comprehension: my_list = [0 for i in range(50)]

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: It's possible to "slice" a list in Python. This will return a specific segment of a given list. For example, the command myList[2] will return the 3rd item in your list (remember that Python begins counting with the number 0)

Q4. Explain the distinctions between indexing and slicing.

Answer: “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?

Answer: 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?

Answer:You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function. Because lists and dictionaries are mutable, changing them (even inside a function) changes the list or dictionary itself, which isn't the case for immutable data types

Q7. What is the concept of an unbalanced matrix?

Answer: 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. In such problems, dummy rows (or columns) are added in the matrix so as to complete it to form a square matrix. The dummy rows or columns will contain all costs elements as zeroes. The Hungarian method may be used to solve the problem

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

Answer: Python is famous for allowing you to write code that’s elegant, easy to write, and almost as easy to read as plain English. One of the language’s most distinctive features is the list comprehension, which you can use to create powerful functionality within a single line of code. However, many developers struggle to fully leverage the more advanced features of a list comprehension in Python. Some programmers even use them too much, which can lead to code that’s less efficient and harder to read.

By the end of this tutorial, you’ll understand the full power of Python list comprehensions and how to use their features comfortably. You’ll also gain an understanding of the trade-offs that come with using them so that you can determine when other approaches are more preferable

Colab paid products - Cancel contracts here