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

**my\_list = [1,2,3,4,5,6,6,7,8,9,10]**

**def bi\_index(in\_list,position):**

**return in\_list[position]**

**print('Positive Indexing ->',bi\_index(my\_list,5))**

**print('Negative Indexing ->',bi\_index(my\_list,-1))**

**o/p ;**

Positive Indexing -> 6

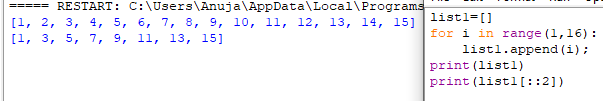
Negative Indexing -> 10

**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.**



**It will print 1 1000 times.**

**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.**

Indexing and slicing are applicable only to sequence data types. The order in which elements are inserted is preserved in sequence type, allowing us to access its elements via indexing and slicing.

**Indexing is used to access individual elements from the sequence.**

The term "**slicing**" refers to obtaining 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?**

If the start index is out of range then it will return empty entity

my\_list **=** [x **for** x **in** range(1,15)]

my\_list **=** [x **for** x **in** range(1,15)]

print(f'my\_list -> {my\_list}')

print(f'Case #1 -> {my\_list[20:]}')

print(f'Case #2 -> {my\_list[10:100]}')

OP:

my\_list -> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

Case #1 -> []

Case #2 -> [11, 12, 13, 14]

**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?**

* **Forget to write return statement**

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

In Unbalanced Matrix number of rows is not same as number of columns.

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

List comprehension or a Loop helps creation of large matrices easy. it also helps to implemeent and avoid manual errors. it also makes reading code easy. Also lot of time for manual feeding is reduced