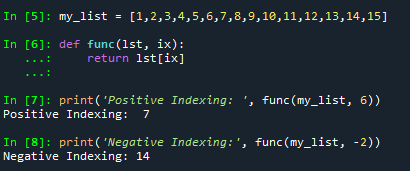
Q1. Can you create a programme or function that employs both positive and negative indexing? Is there any repercussion if you do so?

Answer:

Yes, we can create both positive and negative indexes in the same program. Python supports negative indexing of iterable objects. We can access the elements of the object from the end using indices starting from -1. There is no such repercussion if we use both indexing.

Example:

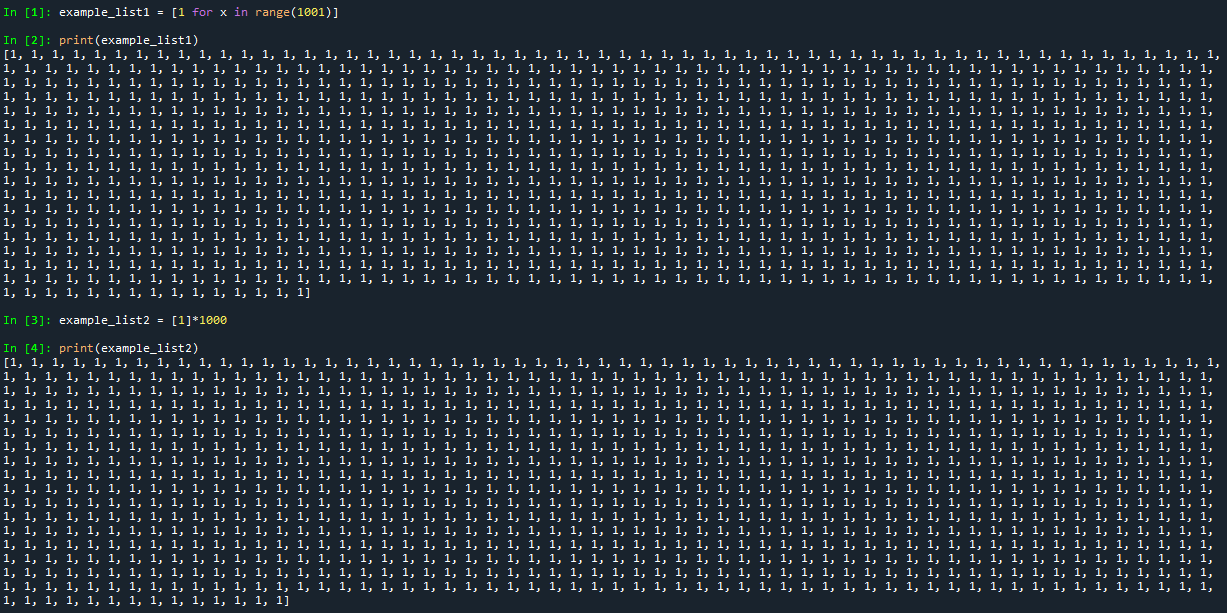


Q2. What is the most effective way of starting with 1,000 elements in a Python list? Assume that all elements should be set to the same value.

Answer:

We can use the multiplication operator (\*) or list Comprehension to accomplish this.

Example:

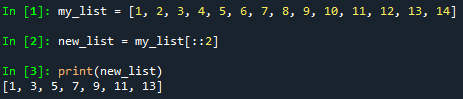


Q3. How do you slice a list to get any other part while missing the rest? (For example, suppose you want to make a new list with the elements first, third, fifth, seventh, and so on.)

Answer:

To achieve this, we use list slicing operations.

Example:



Q4. Explain the distinctions between indexing and slicing.

Answer:

Indexing is a way of accessing a single element at a particular index in a list. Slicing refers to getting a sub-list of items from one index position to another.

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

Answer:

Slice operations do not throw an error if any of the indices are out of bounds. This does not apply to indexing. If we index an element out of bounds, Python will throw an index out of bounds error. However, slicing just returns the existing elements we access. Accessing elements stops when the end of the object is reached.

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:

Always use the return statement if we want to see changes in the input list. If we want the function to be able to change the value of the list, we should avoid creating a new list with the same name inside the function. Instead, we should modify the original list directly, using indexing or slicing.

Q7. What is the concept of an unbalanced matrix?

Answer:

In an unbalanced matrix, the number of rows does not equal the number of columns.

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

Answer:

List comprehensions or loops make it easy to create large matrices. It also helps in implementing and avoiding manual errors. It also makes our code easier to read. Manual feed time is also greatly reduced.