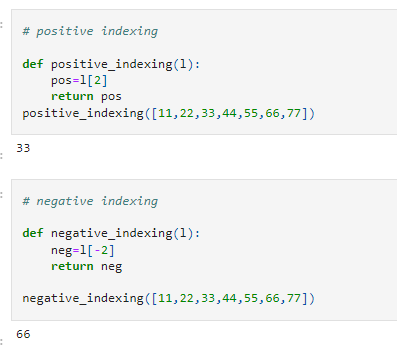
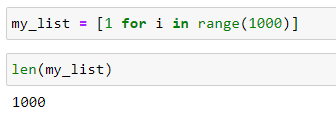
Q1. Can you create a programme or function that employs both positive and negative indexing? Is there any repercussion if you do so?



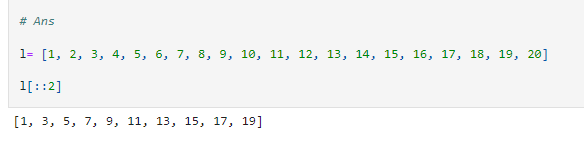
The penalty for doing positive and negative is, positive will count the index position from left side and starts with 0 position.

While in negative indexing, counting is from the right and start with -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.



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.

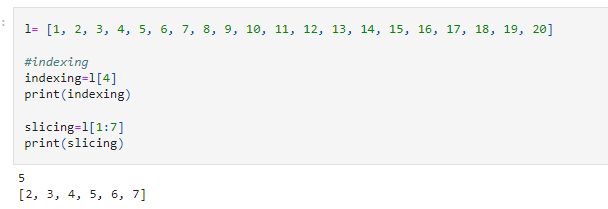
**Indexing:** It is a technique used to get the item of that particuar index position or in cas eof mutable object we can also update the value or add the new value by just giving the index position and assignment operator.

We can use negative indexing also.

**Slicing:** It is use when we wan to extract some range of a item from some object.

We can extract items with some step

We can use negative slicing also.

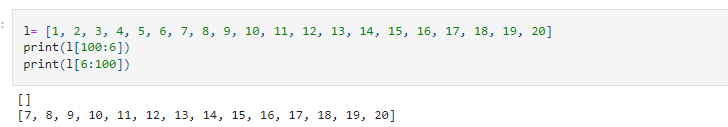


Q5. What happens if one of the slicing expression's indexes is out of range?

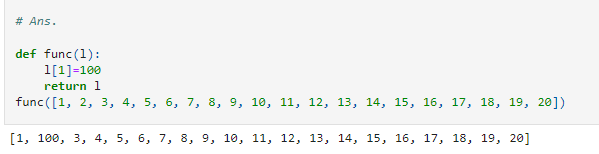
l= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

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

If end index is outof range then it will return the items upto last index position form the start position mention



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?



Q7. What is the concept of an unbalanced matrix?

Unbalanced Matrix is a matrix in which number of rows are not equal to number of columns.

Like 8 x 5 is a unbalance matrix but 8 x 8 is a balanced matrix

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

Because if we do manually by writing every item then firstly this will take a huge time, secondly memory will occupy more, thirdly it is not that much readable to the user and not better understanding. So thats why we use either looping or list comprehension so that in a very short way way we can create a large matrix