1. my\_list **=** [0,1,2,3,4,5,6,7,8,9,10]

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

**return** in\_list[position]

print('Positive Indexing ->',bi\_index(my\_list,7))

print('Negative Indexing ->',bi\_index(my\_list,**-**2))

Output: Positive Indexing -> 7

Negative Indexing -> 9

2. start\_list **=** [1 **for** x **in** range(1001)]

print(start\_list)

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

print(f'my\_list: {my\_list}')

sliced\_list **=** my\_list[::2]

print(f'sliced\_list: {sliced\_list}')

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

sliced\_list -> [1, 3, 5, 7, 9, 11, 13]

4. Indexing is used when we have to work on index level. While slicing are used over a range of items.

Example:

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

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

print(f'Example of indexing -> {my\_list[1], my\_list[6]}')

print(f'Example of slicing -> {my\_list[1:7]}')

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

Example of indexing -> (2, 7)

Example of slicing -> [2, 3, 4, 5,6,7]

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

Example:

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[25:]}')

print(f'Case #2 -> {my\_list[10:50]}')

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

Case #1 -> []

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

6. Always use **return** statement, if we want to see the changes in the input list.

Example:

my\_list **=** [1,2,3,4,5,6]

**def** modify\_list(in\_list):

in\_list**.**append(100)

**return** in\_list

print(modify\_list(my\_list))

Output: [1, 2, 3, 4, 5, 6, 100]

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

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