Question 1. Write a for loop to iterate through the list A = [1, 2, 3, 4, 5, 6]. Square each element of the list in one by one fashion and print them. After the end of the iteration, print - "The sequence has ended".

```
# Answer01
A = [1, 2, 3, 4, 5, 6]
for i in A:
    c = i*i                          ### Squaring The Numbers.
    print(c)
    if i == A[-1]:                   ### Comparing Last Element of given
list.
        print("The Sequence has Ended")
```

```
1
4
9
16
25
36
The Sequence has Ended
```

```
def pattern(n):           # Define Function.
    if n == 2:            # Condition.
        k = 6
        for i in range(1,k):
            for j in range(k-i):
                print("*", end= " ")
            print()

        for i in range(2,k):
            for j in range(i):
                print("-", end= " ")
            print()
    if n== 1:                # Condition for Reverse star Pattern.
        num_rows = 9;        #  No Of star in First row.
        for i in range(num_rows,0,-1):
            for j in range(0, num_rows-i):
                print(end=" ")
            for j in range(0,i):
                print("* ", end=" ")
            print()


    else:
        print("'Invalid Input'")
n = int(input("Enter the Number"))
pattern(n)         # Function Call.
```

```
Enter the Number1
*  *  *  *  *  *  *  *  *
 *  *  *  *  *  *  *  *
```

```
  *  *  *  *  *  *  *
   *  *  *  *  *  *
    *  *  *  *  *
     *  *  *  *
      *  *  *
       *  *
        *
```

Question 3. Create a tuple t_1 = (1, 4, 9, 16, 25, 36). Square each element of the tuple using tuple comprehension and store the result in a variable known as t_modified. Find element at index position 4 of the tuple t_modified. Now slicethe modified tuple in such a way that the sliced tuple includes only elements from index position 1 to 3 and store thissliced tuple in a variable known ast_sliced.

```
t_1 = (1, 4, 9, 16, 25, 36)
t_modified = tuple(x**2 for x in t_1) ### Tuple Comprehension
print(t_modified)
```

```
(1, 16, 81, 256, 625, 1296)
```

```
type(t_modified)    ### Checking Datatype.
```

```
tuple
```

```
t_modified[4] ### Indx Position 4.
```

```
625
```

```
ast_sliced = t_modified[1:4]  ### Store t_modified.
print(ast_sliced)
```

```
(16, 81, 256)
```

Question 4. Show by raising a error how tuple are immutable and also define what exactly immutability is in your ownwords.

```
t = (1,2,3,4,5) ### Tuple Created.
```

```
type(t)
```

```
tuple
```

```
t[0] = 9 ### Hence Tuple is Immutable.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_12160/2315850721.py in <module>
----> 1 t[0] = 9

TypeError: 'tuple' object does not support item assignment
```

Immutability:

Tuples are basically a data type in python. These tuples are an ordered collection of elements of different data types. We represent them by writing the elements inside the parenthesis separated by commas. We can also define tuples as lists that we cannot change. We can call them immutable tuples. Hence, tuples are not modifiable in nature. These immutable tuples are a kind of group data type. We access elements by using the index starting from zero.

Question 5. Create a frozenset named frozen_set_1 containing the elements: 'A', 'B', 'C' and 'D' and combine it using union with a frozenset named frozen_set_2 containing elements 'A', 2, 'C' and 4. The final combined frozenset must be named frozenset_union. Now find the common elements in frozen_set_1 and frozen_set_2 and store the result in a variable named frozenset_common. Lastly, in a new forzenset named forzenset_difference store the elements of frozen_set_1 which arenot in frozen_set_2 and in a new frozenset named frozenset_distinct store the elements which are unique to frozen_set_1and frozen_set_2.

Final output:

frozen_set_1: frozenset({'C', 'A', 'B', 'D'})

frozen_set_2: frozenset({2, 'A', 'C', 4})

frozenset_union: frozenset({2, 'A', 4, 'C', 'B', 'D'})

frozenset_common: frozenset({'C', 'A'})

frozenset_difference: frozenset({'D', 'B'})

frozenset_distinct: frozenset({2, 'B', 4, 'D'})

```
l1 = ['A','B','C','D']
l2 = ['A',2,'C',4]
frozen_set_1 = frozenset(l1)
frozen_set_2 = frozenset(l2)

print(frozen_set_1)
print(frozen_set_2) ### Element of frozen_set_1 and frozen_set_2.

frozenset({'C', 'B', 'A', 'D'})
frozenset({'C', 2, 'A', 4})

frozen_set_union = frozen_set_1.union(frozen_set_2)
frozen_set_union                  ### Print Common set Elements.

frozenset({2, 4, 'A', 'B', 'C', 'D'})

frozenset_common = frozen_set_1.intersection(frozen_set_2)
frozenset_common                ### Common Element of Both sets.

frozenset({'A', 'C'})
```

```
frozenset_difference =
frozenset_difference
```

```
frozenset({'B', 'D'})
```

```
frozenset_distinct = frozen_set_1.symmetric_difference(frozen_set_2)
frozenset_distinct
```

```
frozenset({2, 4, 'B', 'D'})
```

Question 6. Write a python program to remove items in a list containing the character 'a' or 'A'. Use lambda function for it. For this program pass in as argument the list: list_a = ["car", "place", "tree", "under", "grass", "price"] to the lambda function named remove_items_containing_a_or_A.

Final output:

['tree', 'under', 'price']

```
list_a = ["car", "place", "tree", "under", "grass", "price"]
test_list = ['a','A']
list_1 = []

for i in list_a:
    flag = 0
    for j in i:
        if j in test_list:
            flag = 1
        else:
            list_1.append(i)
```

```
list_1
```

```
['c',
 'r',
 'p',
 'l',
 'c',
 'e',
 't',
 'r',
 'e',
 'e',
 'u',
 'n',
 'd',
 'e',
 'r',
 'g',
 'r',
 's',
 's',
```

'p',
'r',
'i',
'c',
'e',
'c',
'r',
'p',
'l',
'c',
'e',
't',
'r',
'e',
'e',
'u',
'n',
'd',
'e',
'r',
'g',
'r',
's',
's',
'p',
'r',
'i',
'c',
'e',
'c',
'r',
'p',
'l',
'c',
'e',
't',
'r',
'e',
'e',
'u',
'n',
'd',
'e',
'r',
'g',
'r',
's',
's',
'p',
'r',

```
'i',
'c',
'e',
'car',
'car',
'place',
'place',
'place',
'place',
'tree',
'tree',
'tree',
'tree',
'under',
'under',
'under',
'under',
'under',
'grass',
'grass',
'grass',
'grass',
'price',
'price',
'price',
'price',
'price',
'car',
'car',
'place',
'place',
'place',
'place',
'tree',
'tree',
'tree',
'tree',
'under',
'under',
'under',
'under',
'under',
'grass',
'grass',
'grass',
'grass',
'price',
'price',
'price',
'price',
```

```
 'price',
 'car',
 'place',
 'grass',
 'car',
 'car',
 'place',
 'place',
 'place',
 'place',
 'tree',
 'tree',
 'tree',
 'tree',
 'under',
 'under',
 'under',
 'under',
 'under',
 'grass',
 'grass',
 'grass',
 'grass',
 'price',
 'price',
 'price',
 'price',
 'price']
```

Question 7: Create a custom exception class which can handle "IndexError" as well as "ValueError" such that it can display its own custom error message when we use index which is not valid in a list. Take list as list_a = [1, 2, 3, 4, 5].

Final output type 1:

Enter the index = 10

The index 10 is incorrect and index should lie between -5 and 4.

Final output type 2:

Enter the index = abc

Use an Integer value as the input.

```
list_a = [1, 2, 3, 4, 5]
```