Lab Tasks

1. Write a program that inputs a line of text. Output the text in uppercase letters and in lowercase letters using built-in functions.

In [1]:
```
text = input("Enter a line of text: ")
uppercase_text = text.upper()
print("Uppercase:", uppercase_text)
lowercase_text = text.lower()
print("Lowercase:", lowercase_text)
```

```
Enter a line of text: my name is noor baig
Uppercase: MY NAME IS NOOR BAIG
Lowercase: my name is noor baig
```

2. A simple way to estimate the number of words in a string is to count the number ofspaces in the string. Write a program that asks the user for a string and returns an estimate of how many words are in the string.

In [4]:
```
string = input("Enter a string: ")
word_count = string.count(' ') + 1
print("Estimated number of words:", word_count)
```

```
Enter a string: """""""""""" """""
Estimated number of words: 2
```

3. Write a program that asks the user to enter a string s and then converts s to lowercase, removes all the periods and commas from s, and prints the resulting string.(Hint: use the split method)

In [7]:
```
string = input("Enter a string: ")
lowercase_string = string.lower()
clean_string = lowercase_string.replace('.', '').replace(',', '')
print("Resulting string:", clean_string)
```

```
Enter a string: """"" """ """""
Resulting string: """"" """ """""
```

4. A variable ids refers to the list [4353, 2314, 2956, 3382, 9362, 3900]. Using list

Loading [MathJax]/extensions/Safe.js

methods, do the following:

a. Remove 3382 from the list.

b. Get the index of 9362.

c. Insert 4499 in the list after 9362.

d. Extend the list by adding [5566, 1830] to it.

e. Reverse the list.

f. Sort the list

In [8]:
```
ids = [4353, 2314, 2956, 3382, 9362, 3900]

ids.remove(3382)
print("After removing 3382:", ids)

index_9362 = ids.index(9362)
print("Index of 9362:", index_9362)

ids.insert(index_9362 + 1, 4499)
print("After inserting 4499:", ids)

ids.extend([5566, 1830])
print("After extending the list:", ids)

ids.reverse()
print("Reversed list:", ids)

ids.sort()
print("Sorted list:", ids)
```

```
After removing 3382: [4353, 2314, 2956, 9362, 3900]
Index of 9362: 3
After inserting 4499: [4353, 2314, 2956, 9362, 4499, 3900]
After extending the list: [4353, 2314, 2956, 9362, 4499, 3900, 5566, 1830]
Reversed list: [1830, 5566, 3900, 4499, 9362, 2956, 2314, 4353]
Sorted list: [1830, 2314, 2956, 3900, 4353, 4499, 5566, 9362]
```

5. When playing games where you have to roll two dice, it is nice to know the odds of each

roll. For instance, the odds of rolling a 12 are about 3%, and the odds of rolling a 7 are about 17%. You can compute these mathematically, but if you don’t know the math, you

can write a program to do it. To do this, your program should simulate rolling two dice

about 10,000 times and compute and print out the percentage of rolls that come out to

be 2, 3, 4, . . . , 12. Print the results in a table format.

In [10]:
```
import random

def roll_dice():
    die1 = random.randint(1, 6)
    die2 = random.randint(1, 6)
    return die1 + die2

sum_counts = {x: 0 for x in range(2, 13)}
for _ in range(10000):
    roll_sum = roll_dice()
    sum_counts[roll_sum] += 1

for i in range(2, 13):
    percentage = (sum_counts[i] / 10000) * 100
    print(f"{i}\t{percentage:.2f}%")
```

```
2       2.54%
3       5.49%
4       8.21%
5       10.91%
6       15.04%
7       16.21%
8       13.84%
9       10.92%
10      8.59%
11      5.30%
12      2.95%
```

6. Write a function each for mean, median and mode that accepts a list as an argument.

The mean, median, and mode functions return an integer.

Loading [MathJax]/extensions/Safe.js

a. Using these functions, write a program that generates a 99-element list of

random numbers between 1 and 100 (inclusive) and computes its mean, median,

mode.

In [11]:
```
import random
from collections import Counter
from statistics import median

def calculate_mean(numbers):
    total = sum(numbers)
    return total // len(numbers)

def calculate_mode(numbers):
    counts = Counter(numbers)
    mode = max(counts, key=counts.get)
    return mode

random_list = [random.randint(1, 100) for _ in range(99)]

mean = calculate_mean(random_list)

median = median(random_list)

mode = calculate_mode(random_list)

print("Mean:", mean)
print("Median:", median)
print("Mode:", mode)
```

```
Mean: 45
Median: 44
Mode: 24
```

7. Write a program that rotates the elements of a list so that the element at the first index

moves to the second index, the element in the second index moves to the third index,

etc., and the element in the last index moves to the first index.

In [12]:
```
def rotate_list(lst):
    rotated_lst = [lst[-1]] + lst[:-1]
    return rotated_lst
```

Loading [MathJax]/extensions/Safe.js

```
my_list = [1, 2, 3, 4, 5]
rotated_list = rotate_list(my_list)
print(rotated_list)
```

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

8. Write a program that generates 100 random integers that are either 0 or 1. Then find the longest run of zeros, the largest number of zeros in a row. For instance, the longest run of zeros in [1,0,1,1,0,0,0,0,1,0,0] is 4.

In [13]:

```
import random
random_list = [random.randint(0, 1) for _ in range(100)]

max_zeros = 0
current_zeros = 0

for num in random_list:
    if num == 0:
        current_zeros += 1
        max_zeros = max(max_zeros, current_zeros)
    else:
        current_zeros = 0
print("Longest run of zeros:", max_zeros)
```

```
Longest run of zeros: 5
```

9. Read the size of the matrix from the user. Populate it with data from the user. Print the

original ,transpose and product of the matrices on the screen.

In [17]:

```
def read_matrix(rows, cols):
    matrix = []
    for i in range(rows):
        row = []
        for j in range(cols):
            element = int(input(f"Enter element at position ({i+1},{j+1}): "))
            row.append(element)
        matrix.append(row)
    return matrix
def print_matrix(matrix):
    for row in matrix:
        print(row)

def transpose_matrix(matrix):
    rows = len(matrix)
    cols = len(matrix[0])
    transposed_matrix = [[matrix[j][i] for j in range(rows)] for i in range(cols)]
    return transposed_matrix
```

Loading [MathJax]/extensions/Safe.js

```
def multiply_matrices(matrix1, matrix2):
    rows1, cols1 = len(matrix1), len(matrix1[0])
    rows2, cols2 = len(matrix2), len(matrix2[0])
    if cols1 != rows2:
        return None

    product_matrix = []
    for i in range(rows1):
        row = []
        for j in range(cols2):
            element = sum(matrix1[i][k] * matrix2[k][j] for k in range(cols1))
            row.append(element)
        product_matrix.append(row)
    return product_matrix

rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))

matrix = read_matrix(rows, cols)

print("Original Matrix:")
print_matrix(matrix)

transposed_matrix = transpose_matrix(matrix)
print("Transposed Matrix:")
print_matrix(transposed_matrix)

product = multiply_matrices(matrix, transposed_matrix)

print("Product of the Matrix and its Transpose:")
if product:
    print_matrix(product)
else:
    print("Matrices cannot be multiplied.")
```

```
Enter the number of rows: 2
Enter the number of columns: 3
Enter element at position (1,1): 4
Enter element at position (1,2): 4
Enter element at position (1,3): 2
Enter element at position (2,1): 5
Enter element at position (2,2): 4
Enter element at position (2,3): 3
Original Matrix:
[4, 4, 2]
[5, 4, 3]
Transposed Matrix:
[4, 5]
[4, 4]
[2, 3]
Product of the Matrix and its Transpose:
[36, 42]
[42, 50]
```

```
In [ ]:
```

Loading [MathJax]/extensions/Safe.js