 Assignment 1: Celebal Summer Internship

Submitted by:

Mahi Verma

Mody University

1. Create lower triangular, upper triangular and pyramid containing the "\*" character.
2. **Lower Triangle**: A lower triangular pattern has stars only on or below the main diagonal.

**Brute Force**: Using nested loops, outer For loop for rows and inner For loop for columns.

**Code:**

def lower\_triangular(n):

# Outer loop for each row

for i in range(1, n + 1):

# Inner loop to print stars in each row

for j in range(i):

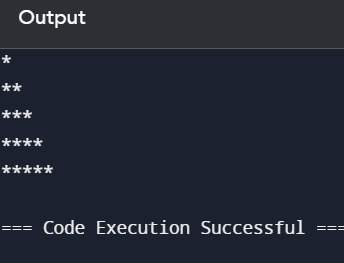
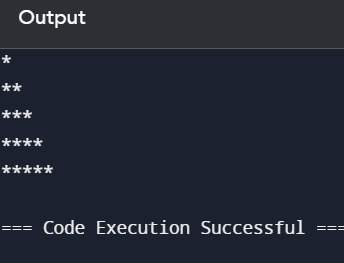
print('\*', end='')

print() # Newline after each row

# Example usage

lower\_triangular(5)

Output:



Time complexity : O(n^2)

Space complexity : O(1)

**Optimised Approch** : The for loop runs from 1 to n, and in each iteration, it prints i stars(\*).

Since the number of stars increases with each row, the result is a lower triangular pattern.

**Code:**

def lower\_triangular(n):

# Loop from 1 to n (inclusive)

for i in range(1, n + 1):

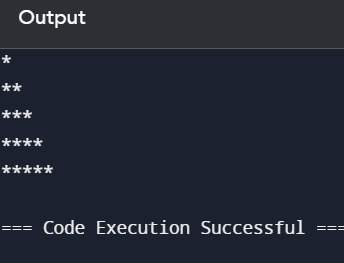
# Print i stars in each row

print('\*' \* i)

# Example usage

lower\_triangular(5)

**Output:**



**Time complexity** : O(n)

**Space complexity** : O(1)

b**) Upper Triangle**: An upper triangular pattern has stars only on or above the main diagonal.

**Brute Force** : Using nested loops, outer for loop for rows and inner for loop for columns.

**Code**:

def upper\_triangular(n):

# Outer loop for each row

for i in range(n, 0, -1):

# Inner loop to print stars in each row

for j in range(i):

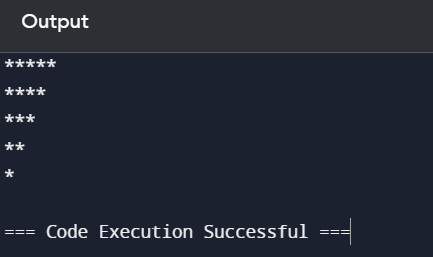
print('\*', end='')

print() # Newline after each row

# Example usage

upper\_triangular(5)

**Output:**



**Time complexity:** O(n^2)

**Space complexity:** O(1)

**Optimized Approach:** The for loop runs from n to 1, and in each iteration, it prints i stars (\*). Since the number of stars decreases with each row, the result is an upper triangular pattern.

**Code:**

def upper\_triangular(n):

# Loop from n to 1 (inclusive)

for i in range(n, 0, -1):

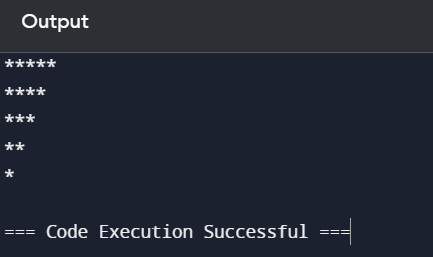
# Print i stars in each row

print('\*' \* i)

# Example usage

upper\_triangular(5)

Output :



**Time complexity:** O(n)

**Space complexity:** O(1)

**c)Pyramid Triangle:**

A pyramid pattern is symmetrical and centered, akin to Pyramid.

**Brute Force:** Using nested loops, outer for loop for rows and inner for loop for columns.

**Code:**

def pyramid(n):

# Outer loop for each row

for i in range(1, n + 1):

# Inner loop to print leading spaces

for j in range(n - i):

print(' ', end='')

# Inner loop to print stars

for k in range(2 \* i - 1):

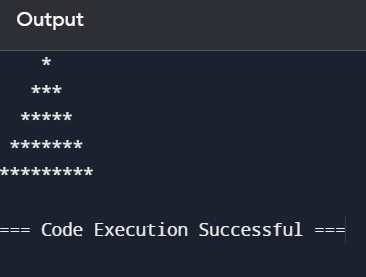
print('\*', end='')

print() # Newline after each row

# Example usage

pyramid(5)

Output :



**Time complexity:** O(n^2)

**Space complexity:** O(1)

**Optimised Approach** :

The for loop runs from 1 to n, and in each iteration, it calculates and prints:

* (n - i) spaces to center the stars
* (2 \* i - 1) stars to form the pyramid shape This ensures the stars are centered and form a symmetrical pyramid.

**Code:**

def pyramid(n):

# Loop from 1 to n (inclusive)

for i in range(1, n + 1):

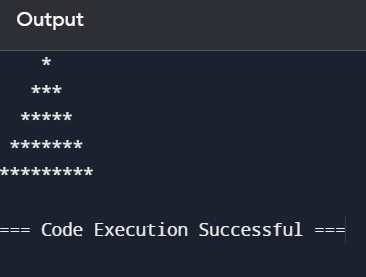
# Print (n - i) spaces followed by (2 \* i - 1) stars

print(' ' \* (n - i) + '\*' \* (2 \* i - 1))

# Example usage

pyramid(5)

**Output :**



**Time complexity:** O(n)

**Space complexity:** O(1)