28) Generate Fibonacci series of N terms

```
def generate_fibonacci(n):
    fibonacci_series = []
    a, b = 0, 1
    for _ in range(n):
        fibonacci_series.append(a)
        a, b = b, a + b
    return fibonacci_series
n = int(input("Enter the number of terms for Fibonacci series: "))
fibonacci_series = generate_fibonacci(n)
print("Fibonacci series of", n, "terms:", fibonacci_series)
```

29) Find the sum of all items in a list

```
numbers = input("Enter a list of numbers separated by spaces: ")
# Convert the input string to a list of integers
numbers_list = list(map(int, numbers.split()))
# Calculate the sum of all items in the list
total_sum = sum(numbers_list)
print("Sum of all items in the list:", total_sum)
```

30) Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.

```
def is_all_even(number):
    return all(int(digit) % 2 == 0 for digit in str(number))


def is_perfect_square(number):
    root = int(number ** 0.5)
    return root * root == number


# Get the range from the user
start_range = int(input("Enter the starting number of the range (four digits): "))
end_range = int(input("Enter the ending number of the range (four digits): "))


# Generate the list of numbers meeting the criteria
result_list = [num for num in range(start_range, end_range + 1) if is_all_even(num) and is_perfect_square(num)]
```

```
print("List of four-digit numbers with all even digits and perfect squares:")
print(result_list)
```

31) Display the given pyramid with step number accepted from user

. Eg: N=4

1

2 4

3 6 9

4 8 12 16

```
# Get the number of steps from the user
n = int(input("Enter the number of steps for the pyramid: "))


# Generate and display the pyramid pattern
for i in range(1, n + 1):
    for j in range(1, i + 1):
        # Print step number * column number
        print(i * j, end=" ")
    # Move to the next line for the next row
    print()
```

32) Count the number of characters (character frequency) in a string.

```
# Get the input string from the user
input_string = input("Enter a string: ")


# Create an empty dictionary to store character frequencies
char_frequency = {}


# Count the frequency of each character in the input string
for char in input_string:
    char_frequency[char] = char_frequency.get(char, 0) + 1


# Print character frequencies
print("Character frequencies in the string:")
for char, frequency in char_frequency.items():
    print(f"'{char}': {frequency}")
```

33) Add ‘ing’ at the end of a given string. If it already ends with ‘ing’, then add ‘ly’

```
def add_ing_ly(input_string):
    if input_string.endswith('ing'):
        result_string = input_string + 'ly'
```

```
    else:
        result_string = input_string + 'ing'
    return result_string

# Get input string from user
input_string = input("Enter a string: ")
# Call the function and print the result
modified_string = add_ing_ly(input_string)
print("Modified string:", modified_string)
```

34) Accept a list of words and return length of longest word.

```
def find_longest_word(words_list):
    longest_word = ""
    for word in words_list:
        if len(word) > len(longest_word):
            longest_word = word
    return len(longest_word)
# Get a list of words from the user
words_list = input("Enter a list of words
separated by spaces: ").split()

# Call the function and print the result
```

```
longest_word_length =
find_longest_word(words_list)

print("Length of the longest word:",
longest_word_length)
```

35) Generate all factors of a number.

```
def find_factors(number):
    factors = []
    for i in range(1, number + 1):
        if number % i == 0:
            factors.append(i)
    return factors
# Get the number from the user
number = int(input("Enter a number: "))
# Call the function and print the result
factors = find_factors(number)
print("Factors of", number, "are:", factors)
```

36) Write lambda functions to find area of square, rectangle and triangle

```
# Lambda function to find the area of a square
square_area = lambda side: side ** 2


# Lambda function to find the area of a
rectangle
```

```
rectangle_area = lambda length, width: length * width

# Lambda function to find the area of a triangle

triangle_area = lambda base, height: 0.5 * base * height

# Get measurements from the user

side_length = float(input("Enter the side length of the square: "))

rectangle_length = float(input("Enter the length of the rectangle: "))

rectangle_width = float(input("Enter the width of the rectangle: "))

triangle_base = float(input("Enter the base length of the triangle: "))

triangle_height = float(input("Enter the height of the triangle: "))

# Calculate and display the areas

print("Area of the square:", square_area(side_length))

print("Area of the rectangle:", rectangle_area(rectangle_length, rectangle_width))

print("Area of the triangle:", triangle_area(triangle_base, triangle_height))
```

37) Construct following pattern using nested loop

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

```
# Number of rows in the pattern
num_rows = 9
# Nested loop to construct the pattern
for i in range(1, num_rows + 1):
    num_stars = min(i, num_rows - i + 1)  # Calculate the number of stars for the current row
    for j in range(num_stars):
        print("*", end=" ")
    print()
```