Don Bosco Institute of Technology

Department of Electronics and Telecommunication Engg.

SUB: Skill Lab (ECL404)

Experiment on Methods and functions.

Expt No:	**03**
Aim:	To make use of various Built-in-functions and library functions in python code
Tool used:	Anaconda Navigator+ Jupiter Notebook

Theory:

Answer in brief (not more than 6 to7 sentences.)
Resource: https://www.w3schools.com/python/

1. List and classify the functions and also explain its declaration in python.

A: In Python, functions are blocks of organized, reusable code that perform a specific task. Functions help in modularizing code, making it more readable, maintainable, and scalable. Functions in Python can be broadly classified into built-in functions and user-defined functions.

1. **Built-in Functions:**

These are functions that are already defined in the Python standard library and can be used directly. Examples are:

- print(): Outputs text or variables to the console.
- len(): Returns the length of an object (e.g., a string, list, or tuple).
- type(): Returns the type of an object.

Declaration:

Built-in functions are part of the Python language, so you can use them directly without declaring them. For example:

```
python
print("Hello, World!")
```

2. **User-Defined Functions:**

These are functions that you define yourself to perform specific tasks. They are created using the def keyword.

Declaration:

```
python
def function_name(parameters):
    # Function body
    # Statements
    return result # Optional return statement
```

- def: Keyword used to declare a function.
- function_name: Name of the function, following Python naming conventions.
- parameters: Input values that the function takes (optional).
- return: Keyword to specify the return value of the function (optional).

- **Calling a Function:**

Once a function is defined, you can call it by using its name followed by parentheses, and passing any required arguments:

```
python
result = add_numbers(3, 4)
```

print(result)

2. What is the significance of functions in python? Illustrate with an example.

A: Functions play a crucial role in Python for several reasons, contributing to code organization, reusability, and maintainability. Here are some key significances of functions in Python, illustrated with an example:

1. **Modularity:**
Functions allow you to break down a large program into smaller, manageable, and modular components. Each function can perform a specific task, making the code more organized and easier to understand.

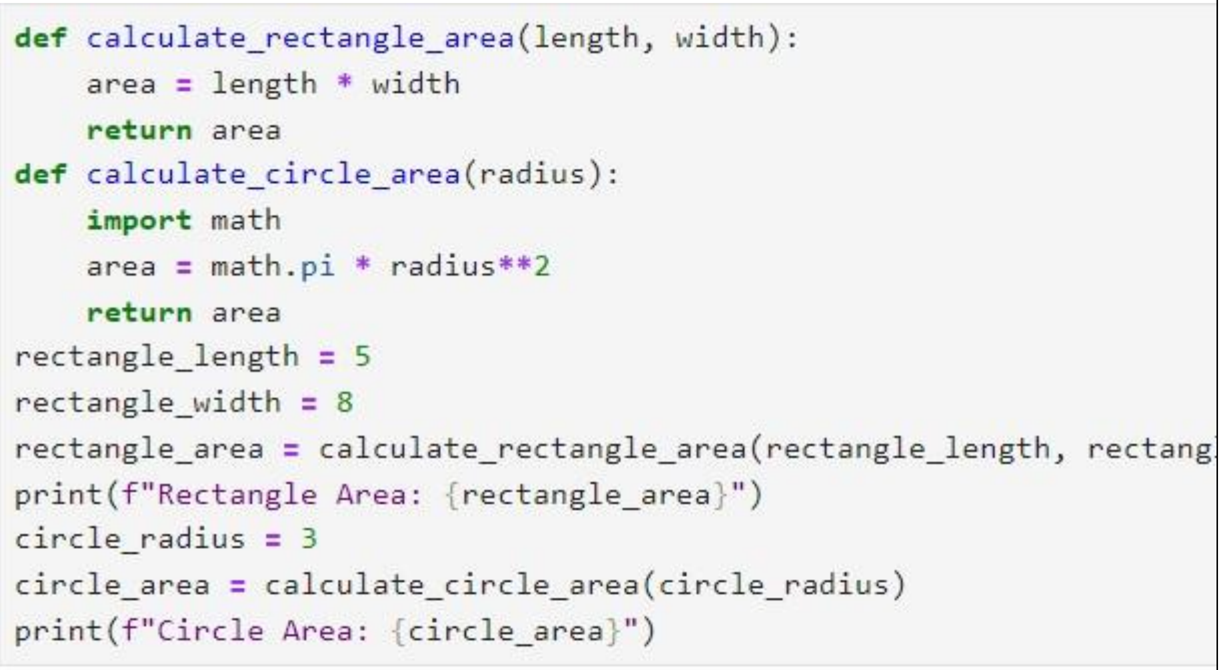
2. **Reusability:**
Once a function is defined, it can be reused in different parts of the code or even in different programs. This promotes code reusability and reduces redundancy.

3. **Readability:**
Functions enhance the readability of code by encapsulating specific functionalities. A well-named function can act as a self-contained unit, making it clear what it does without the need to understand the entire implementation.

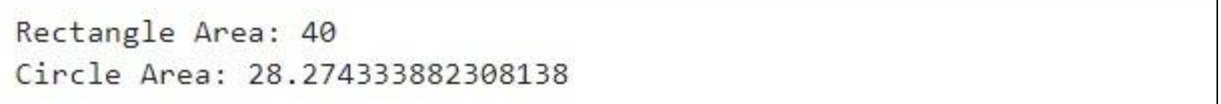
4. **Scalability:**
Functions make it easier to scale and extend a program. If you need to add or modify a feature, you can focus on the relevant function without affecting the rest of the code.

5. **Testing and Debugging:**
Functions simplify the testing and debugging process. Since functions are modular, you can isolate and test individual components, making it easier to identify and fix issues.

Example:



```python
def calculate_rectangle_area(length, width):
    area = length * width
    return area
def calculate_circle_area(radius):
    import math
    area = math.pi * radius**2
    return area
rectangle_length = 5
rectangle_width = 8
rectangle_area = calculate_rectangle_area(rectangle_length, rectang
print(f"Rectangle Area: {rectangle_area}")
circle_radius = 3
circle_area = calculate_circle_area(circle_radius)
print(f"Circle Area: {circle_area}")
```



```
Rectangle Area: 40
Circle Area: 28.274333882308138
```

3. Explain ‘arguments’ in detail.

A: In Python, arguments are values that you pass to a function when calling it. They provide a way to supply input data to a function, allowing the function to perform operations on that data. Arguments are essential for making functions flexible and reusable. There are two types of arguments in Python: positional arguments and keyword arguments.

1. **Positional Arguments:**

These are the most common type of arguments.

- They are passed to a function based on their position or order in the function call.
- The order and number of positional arguments in the function call must match the order and number of parameters in the function definition.

Example:

python

```
def add_numbers(a, b):
    sum_result = a + b
    return sum_result

result = add_numbers(3, 4)  # Here, 3 is assigned to 'a', and 4 is assigned to 'b'
```

2. **Keyword Arguments:**

- These are passed to a function by explicitly specifying the parameter names along with their values.
- This allows you to pass arguments out of order or skip certain arguments, providing more flexibility.

Example:

python

```
def greet(name, greeting):
    print(f"{greeting}, {name}!")
greet(greeting="Hello", name="John")  # Here, the order is different, but keyword arguments are used
```

3. Default Values:

- Parameters in a function can have default values, which are used if the corresponding argument is not provided in the function call.
- Parameters with default values must come after parameters without default values in the function definition.

Example:

python

```
def power(base, exponent=2):
    result = base ** exponent
    return result
print(power(3))      # Uses default exponent (2), result: 9
print(power(3, 3))    # Overrides default exponent, result: 27
```

4. **Any Number of Arguments:**

- You can define functions that accept a variable number of positional or keyword arguments using *args and **kwargs.
- *args allows a function to accept any number of positional arguments.
- **kwargs allows a function to accept any number of keyword arguments.

Example:

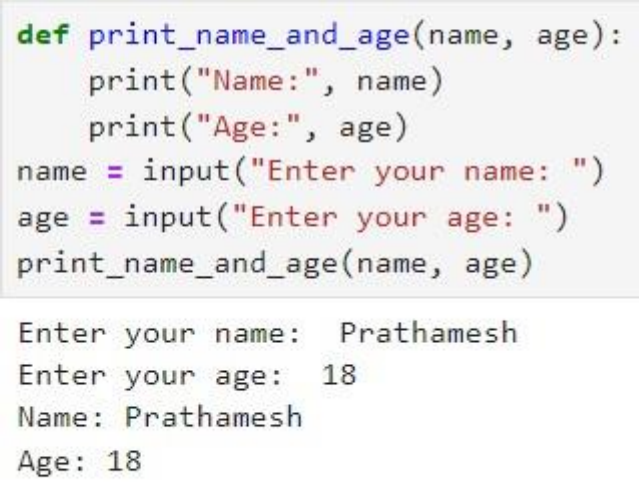
python

```
def print_arguments(*args, **kwargs):
    for arg in args:
        print(arg)
    for key, value in kwargs.items():
        print(f"{key}: {value}")
print_arguments(1, 2, 3, name="John", age=25)
```

Here, *args collects positional arguments into a tuple, and **kwargs collects keyword arguments into a dictionary. In summary, arguments in Python provide a way to pass data into functions. Positional arguments, keyword arguments, default values, and variable-length arguments contribute to the flexibility and versatility of functions in Python.

Tasks and outputs:

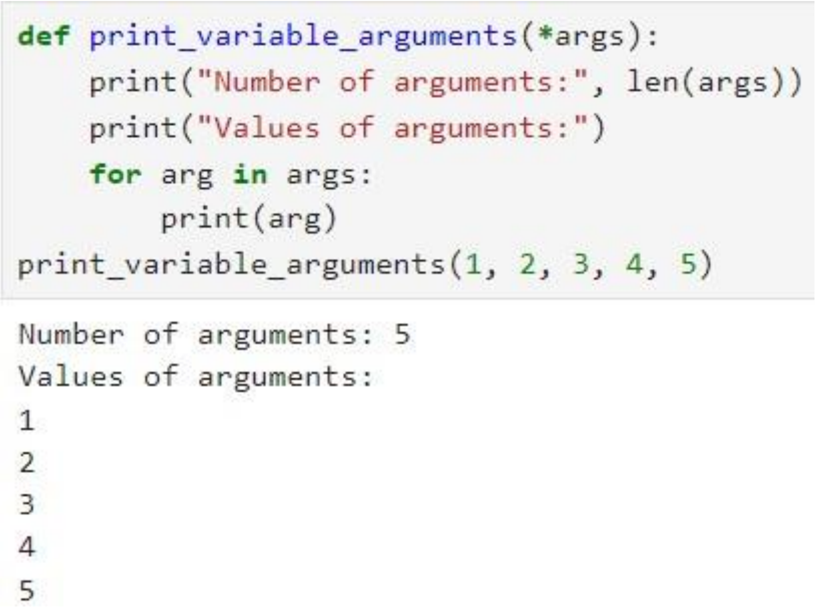
1. Write a program to create a function that takes two arguments, name and age, and print their value.



```
def print_name_and_age(name, age):
    print("Name:", name)
    print("Age:", age)
name = input("Enter your name: ")
age = input("Enter your age: ")
print_name_and_age(name, age)
```

```
Enter your name:  Prathamesh
Enter your age:  18
Name: Prathamesh
Age: 18
```

2. Write a program to create function to accept a variable length of arguments and print their value.

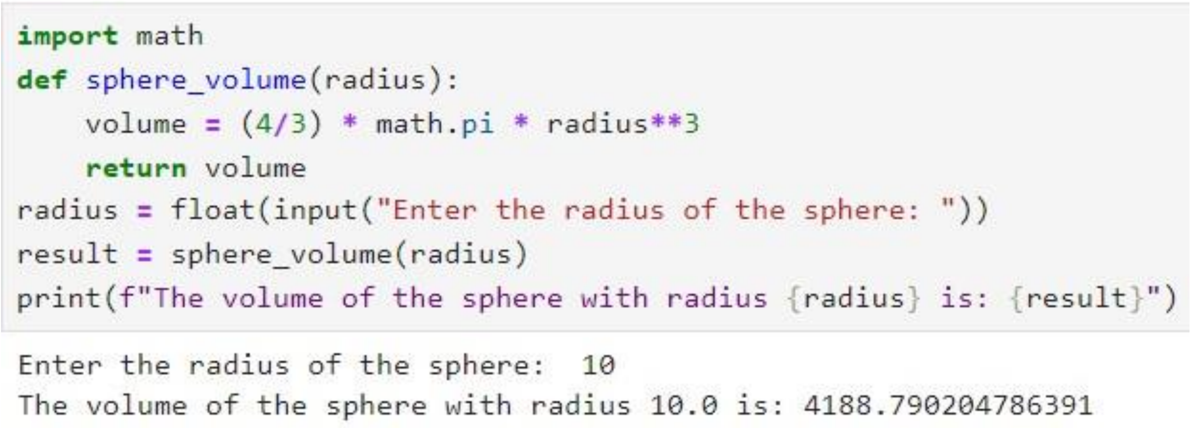


```
def print_variable_arguments(*args):
    print("Number of arguments:", len(args))
    print("Values of arguments:")
    for arg in args:
        print(arg)
print_variable_arguments(1, 2, 3, 4, 5)
```

```
Number of arguments: 5
Values of arguments:
1
2
3
4
5
```

3. Write a function that computes the volume of a sphere given its radius in 3D space.

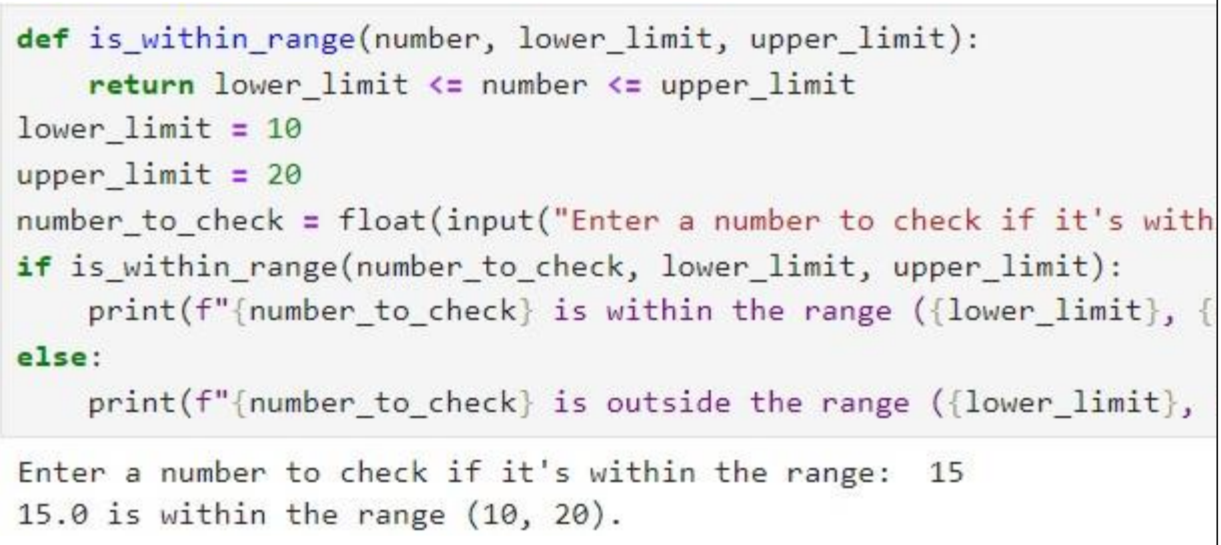
$$v = \frac{4}{3}\pi r^3$$



```
import math
def sphere_volume(radius):
    volume = (4/3) * math.pi * radius**3
    return volume
radius = float(input("Enter the radius of the sphere: "))
result = sphere_volume(radius)
print(f"The volume of the sphere with radius {radius} is: {result}")
```

```
Enter the radius of the sphere:  10
The volume of the sphere with radius 10.0 is: 4188.790204786391
```

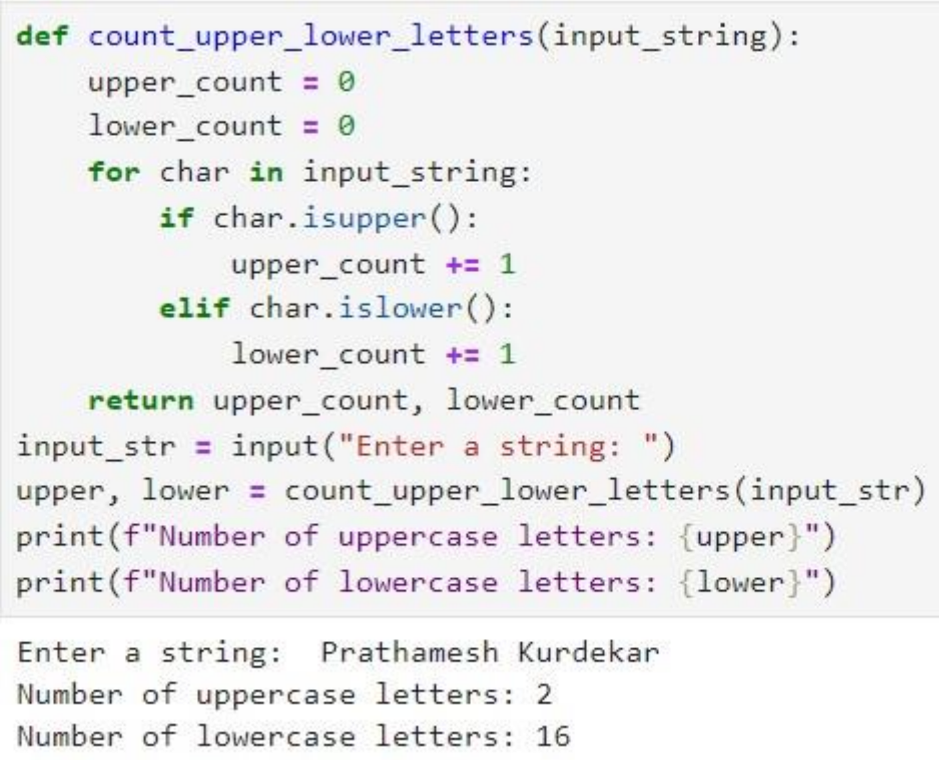
4. Write a function to check if a given number lies within the range limit.



```
def is_within_range(number, lower_limit, upper_limit):
    return lower_limit <= number <= upper_limit
lower_limit = 10
upper_limit = 20
number_to_check = float(input("Enter a number to check if it's with
if is_within_range(number_to_check, lower_limit, upper_limit):
    print(f"{number_to_check} is within the range ({lower_limit}, {
else:
    print(f"{number_to_check} is outside the range ({lower_limit},
```

```
Enter a number to check if it's within the range:  15
15.0 is within the range (10, 20).
```

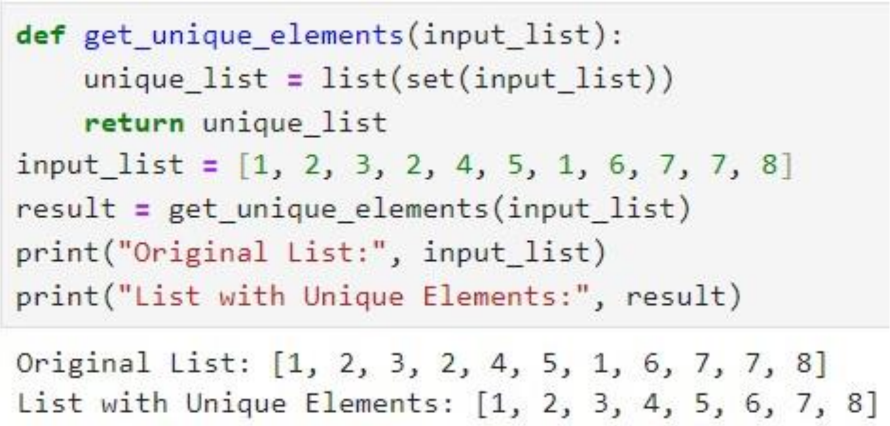
5. Write a code to calculate the number of upper-case letters and lower-case letters in a string.



```
def count_upper_lower_letters(input_string):
    upper_count = 0
    lower_count = 0
    for char in input_string:
        if char.isupper():
            upper_count += 1
        elif char.islower():
            lower_count += 1
    return upper_count, lower_count
input_str = input("Enter a string: ")
upper, lower = count_upper_lower_letters(input_str)
print(f"Number of uppercase letters: {upper}")
print(f"Number of lowercase letters: {lower}")
```

```
Enter a string:  Prathamesh Kurdekar
Number of uppercase letters: 2
Number of lowercase letters: 16
```

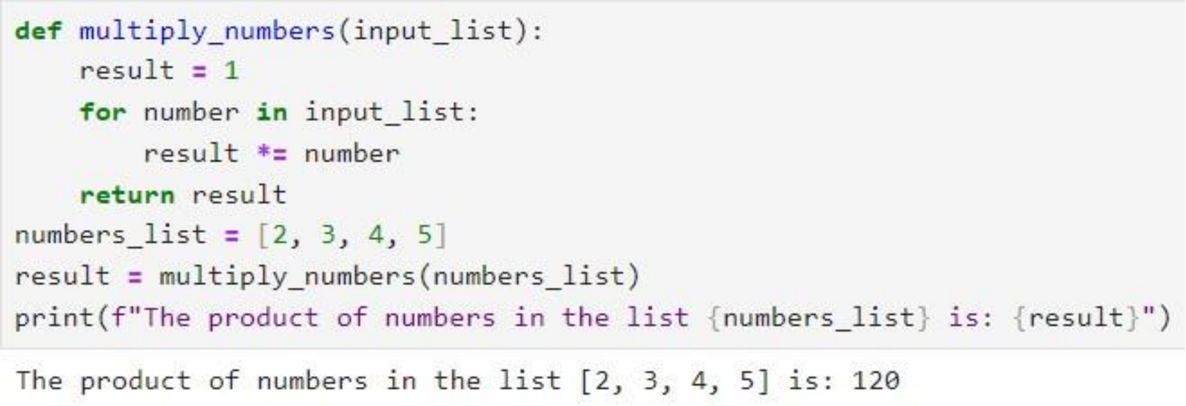
6. Write a function which takes a list and return the same list with unique elements.



```
def get_unique_elements(input_list):
    unique_list = list(set(input_list))
    return unique_list
input_list = [1, 2, 3, 2, 4, 5, 1, 6, 7, 7, 8]
result = get_unique_elements(input_list)
print("Original List:", input_list)
print("List with Unique Elements:", result)
```

```
Original List: [1, 2, 3, 2, 4, 5, 1, 6, 7, 7, 8]
List with Unique Elements: [1, 2, 3, 4, 5, 6, 7, 8]
```

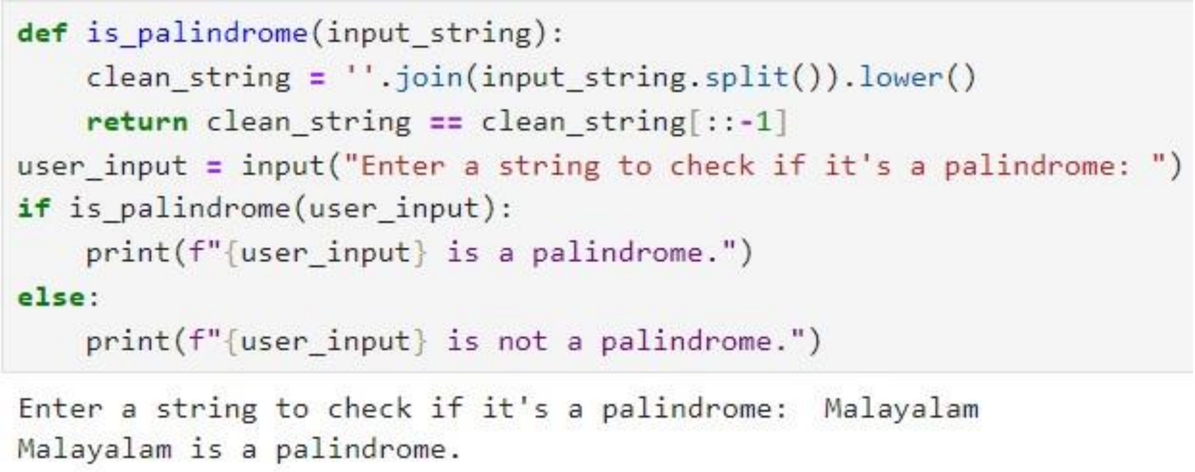
7. Write a function to Multiply all the numbers in a predefined list.



```
def multiply_numbers(input_list):
    result = 1
    for number in input_list:
        result *= number
    return result
numbers_list = [2, 3, 4, 5]
result = multiply_numbers(numbers_list)
print(f"The product of numbers in the list {numbers_list} is: {result}")
```

```
The product of numbers in the list [2, 3, 4, 5] is: 120
```

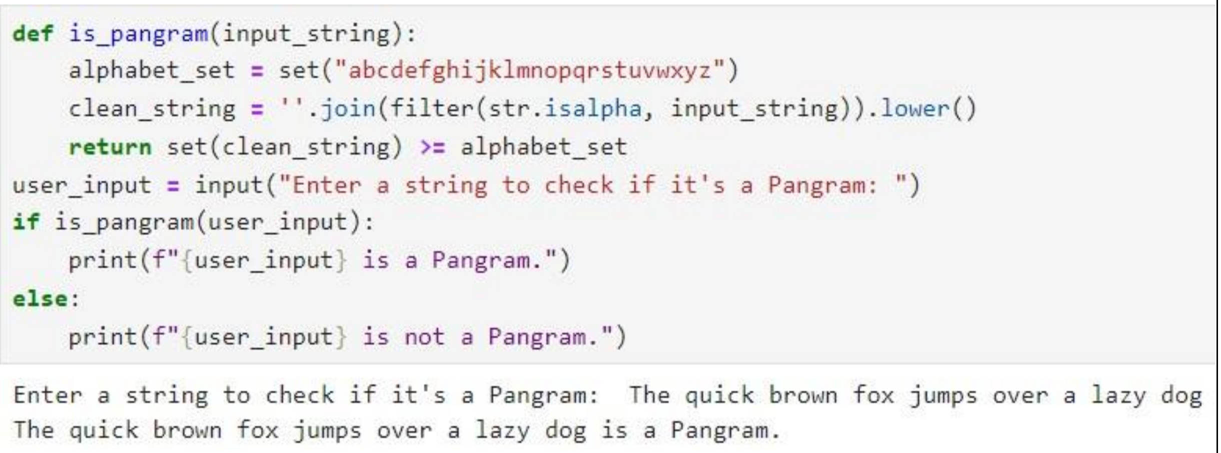
8. Write a function to check if the given string is a Palindrome or not.



```
def is_palindrome(input_string):
    clean_string = ''.join(input_string.split()).lower()
    return clean_string == clean_string[::-1]
user_input = input("Enter a string to check if it's a palindrome: ")
if is_palindrome(user_input):
    print(f"{user_input} is a palindrome.")
else:
    print(f"{user_input} is not a palindrome.")
```

```
Enter a string to check if it's a palindrome:  Malayalam
Malayalam is a palindrome.
```

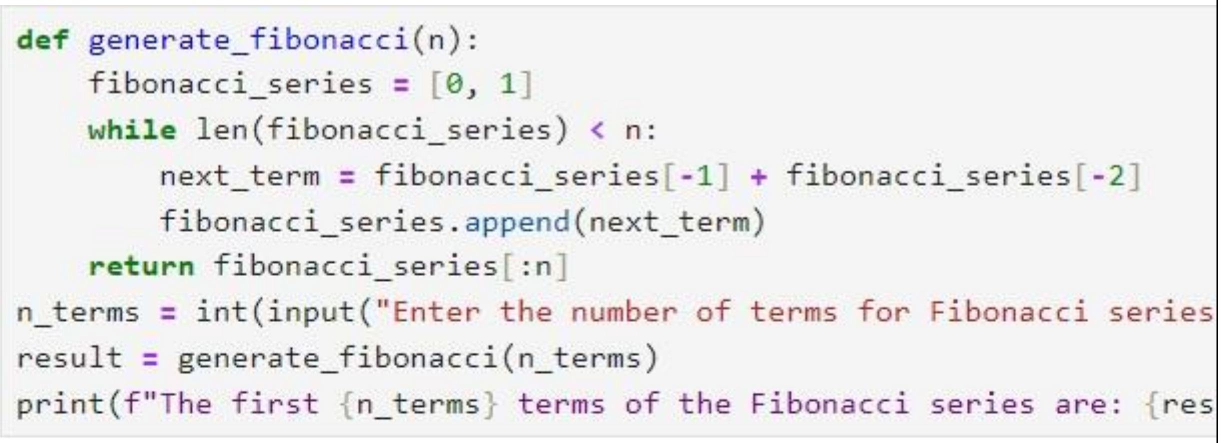
9. Write a function to check if the given string is a Pangram.



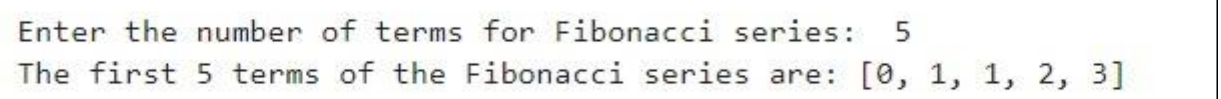
```
def is_pangram(input_string):
    alphabet_set = set("abcdefghijklmnopqrstuvwxyz")
    clean_string = ''.join(filter(str.isalpha, input_string)).lower()
    return set(clean_string) >= alphabet_set
user_input = input("Enter a string to check if it's a Pangram: ")
if is_pangram(user_input):
    print(f"{user_input} is a Pangram.")
else:
    print(f"{user_input} is not a Pangram.")
```

```
Enter a string to check if it's a Pangram:  The quick brown fox jumps over a lazy dog
The quick brown fox jumps over a lazy dog is a Pangram.
```

10. Write a function to generate the first ‘n’ terms of the Fibonacci series.



```
def generate_fibonacci(n):
    fibonacci_series = [0, 1]
    while len(fibonacci_series) < n:
        next_term = fibonacci_series[-1] + fibonacci_series[-2]
        fibonacci_series.append(next_term)
    return fibonacci_series[:n]
n_terms = int(input("Enter the number of terms for Fibonacci series
result = generate_fibonacci(n_terms)
print(f"The first {n_terms} terms of the Fibonacci series are: {res
```

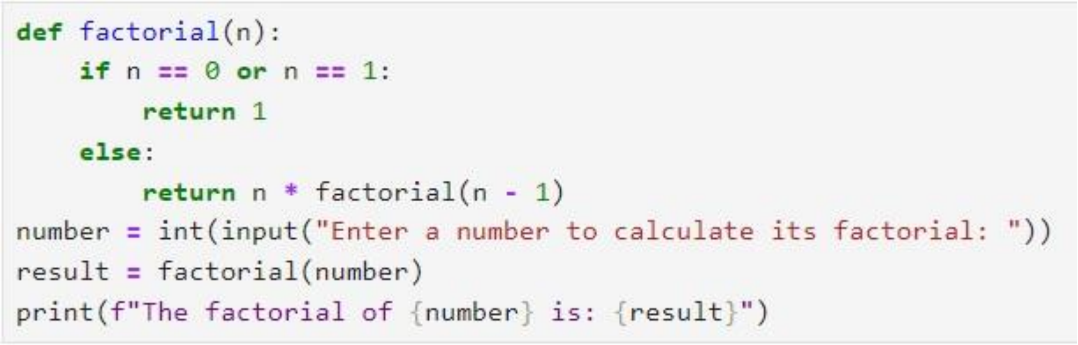


```
Enter the number of terms for Fibonacci series:  5
The first 5 terms of the Fibonacci series are: [0, 1, 1, 2, 3]
```

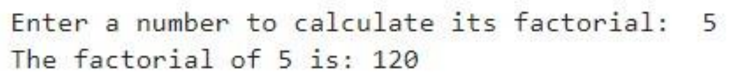
Observation and Conclusion:

Include one additional program with output based on ‘functions’ exclusive of the above exercise. Write about the task and algorithm used.

Program: To calculate the factorial of a given number using a recursive function.



```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
number = int(input("Enter a number to calculate its factorial: "))
result = factorial(number)
print(f"The factorial of {number} is: {result}")
```



```
Enter a number to calculate its factorial:  5
The factorial of 5 is: 120
```

Algorithm:

1. If the input number n is 0 or 1, return 1 (base case).
2. Otherwise, calculate the factorial of n by multiplying n with the factorial of n - 1 (recursive step).
3. Repeat steps 1 and 2 until the base case is reached.