CDPS Python Labs:

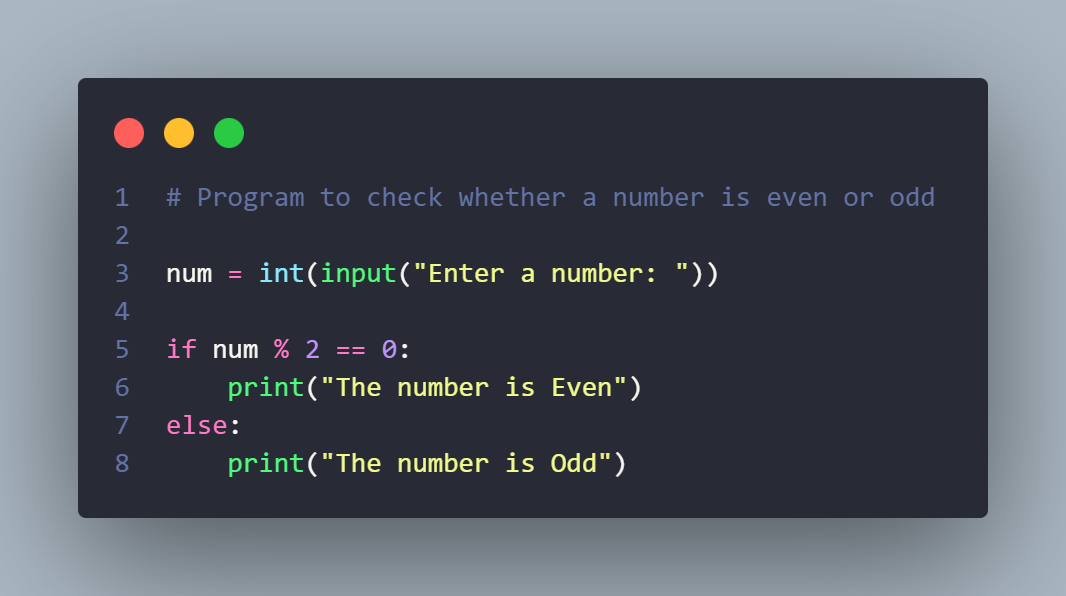
Week 1:

**Question 1:** Write a Python program to check whether a given number is even or odd.

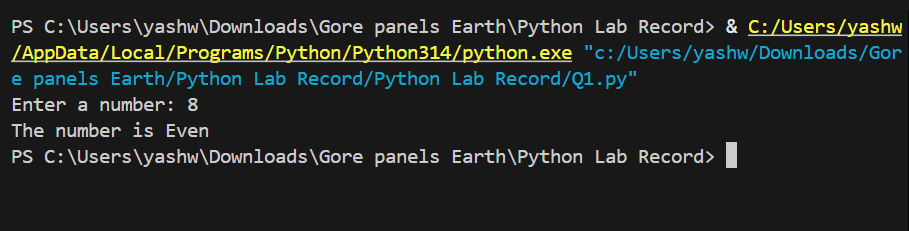
Algorithm:

1. Start
2. Read an integer number from the user
3. Check if the number modulo 2 equals 0
4. If true, print “The number is Even”
5. Else, print “The number is Odd”
6. Stop

**Code:**



**Output:**

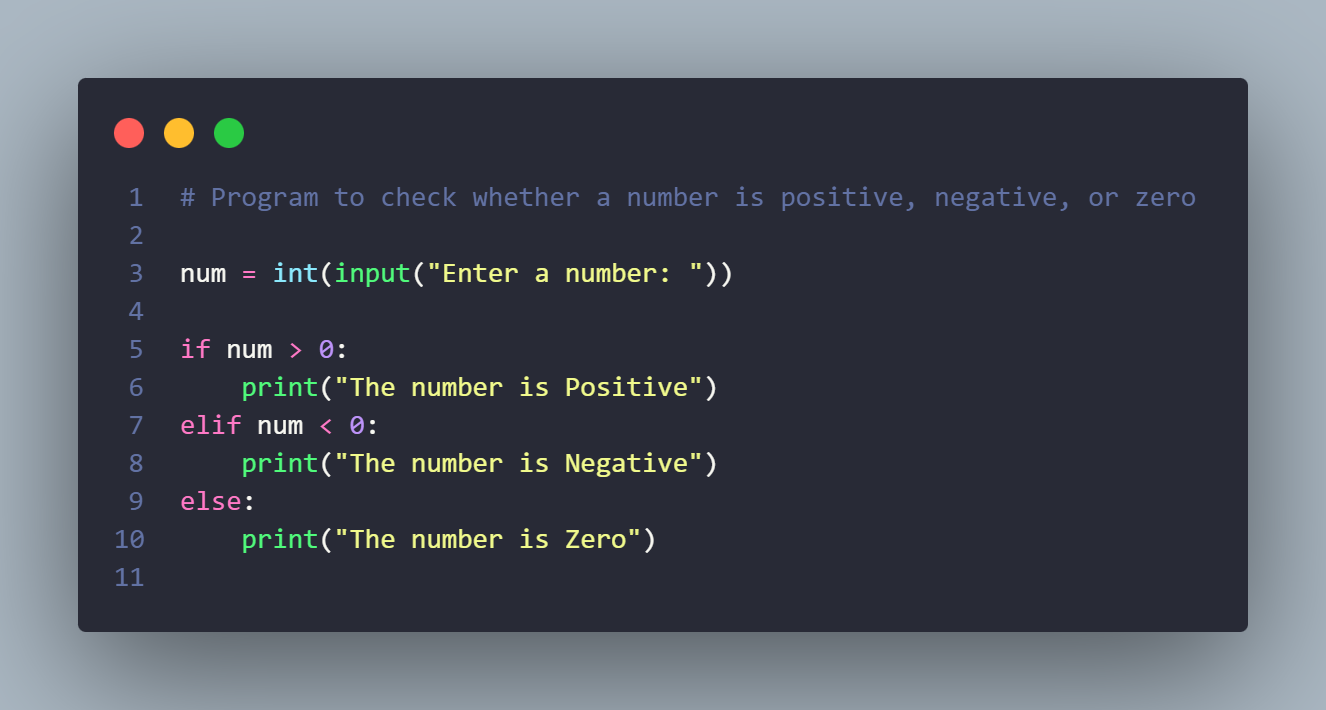


**Question 2:** Write a Python program to check whether a number is positive, negative, or zero.

**Algorithm:**

1. Start
2. Read an integer number from the user
3. If the number is greater than 0, print “The number is Positive”
4. Else if the number is less than 0, print “The number is Negative”
5. Else, print “The number is Zero”
6. Stop

**Code:**



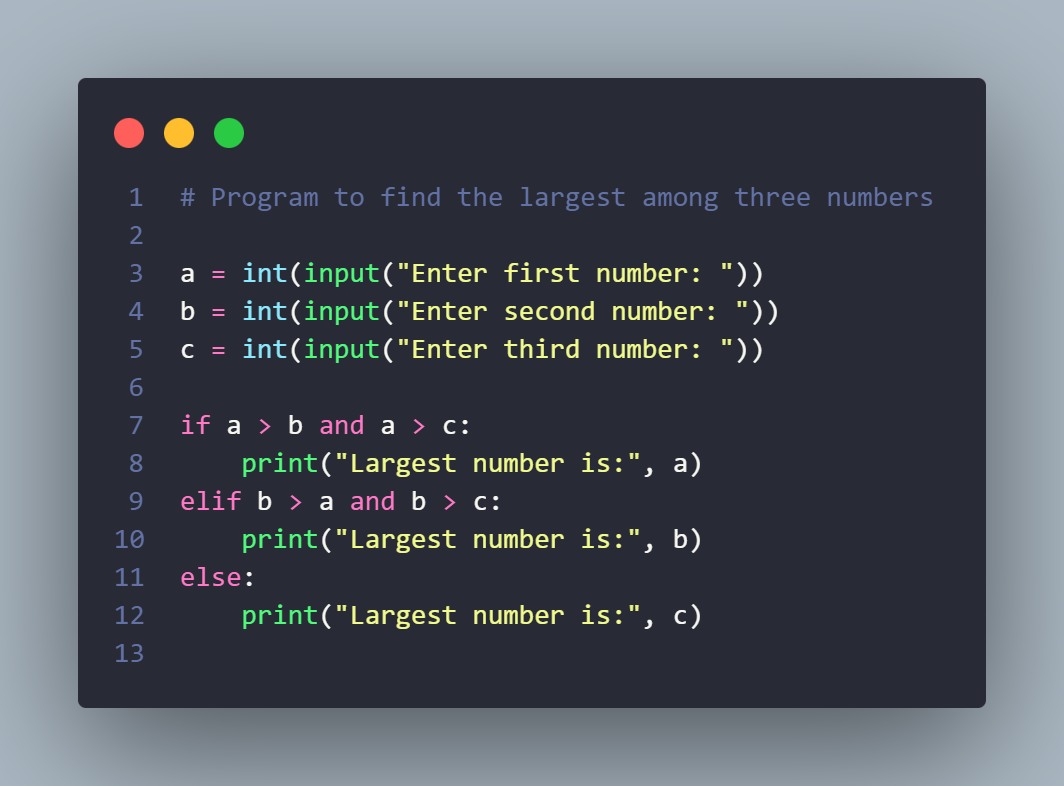
**Output:**

**Question 3:** Write a Python program to find the largest among three numbers.

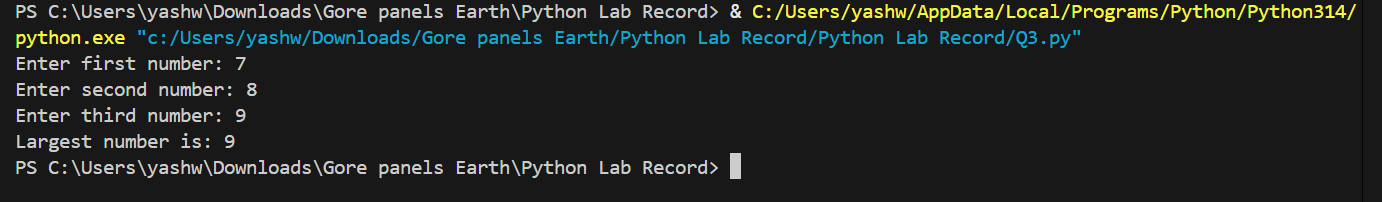
**Algorithm:**

1. Start
2. Read three numbers from the user
3. Compare the first number with the second and third
4. If the first number is greater than both, print it as the largest
5. Else if the second number is greater than the first and third, print it as the largest
6. Else, print the third number as the largest

**Code:**



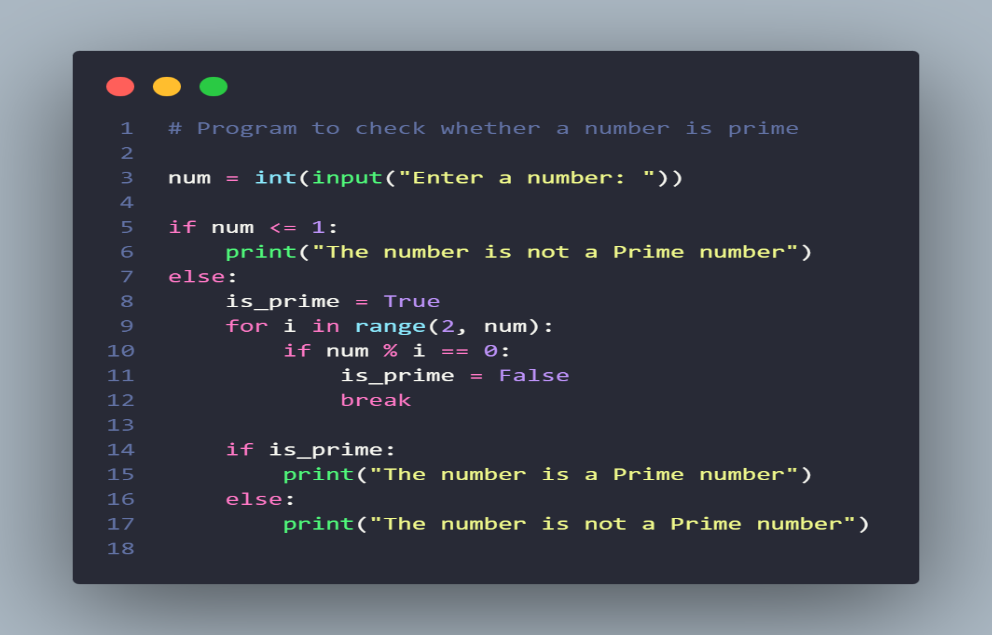
**Output:**



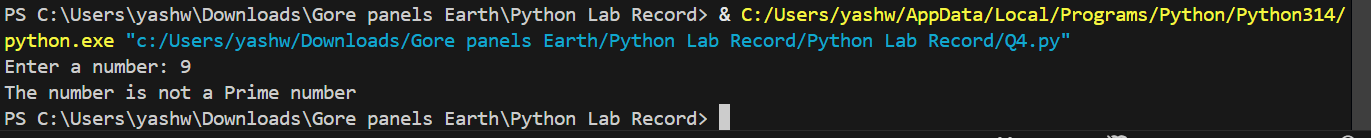
**Question 4:** Write a Python program to check whether a given number is a prime number.

**Algorithm:**

1. Start
2. Read an integer number from the user
3. If the number is less than or equal to 1, it is not a prime number
4. Else, check divisibility of the number from 2 to number − 1
5. If the number is divisible by any value, it is not a prime number
6. If no divisors are found, it is a prime number
7. Stop

**Code:** 

**Output:**



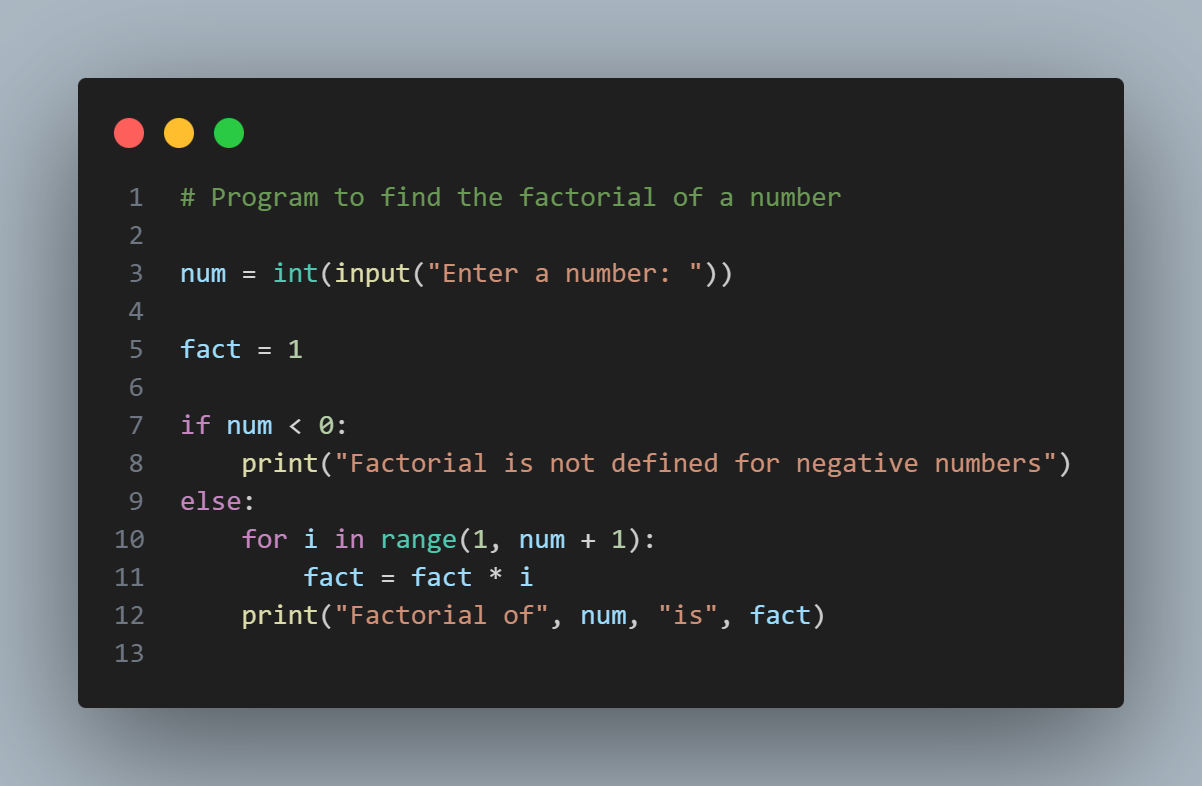
**Week 2:**

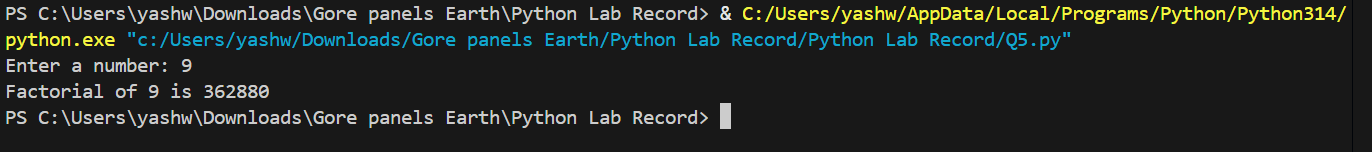
**Question 5:** Write a Python program to find the factorial of a number.

**Algorithm:**

1. Start
2. Read an integer number from the user
3. Initialize a variable fact to 1
4. If the number is negative, print that factorial is not defined
5. Else, multiply fact with numbers from 1 to the given number
6. Print the factorial value
7. Stop

**Code:**

****

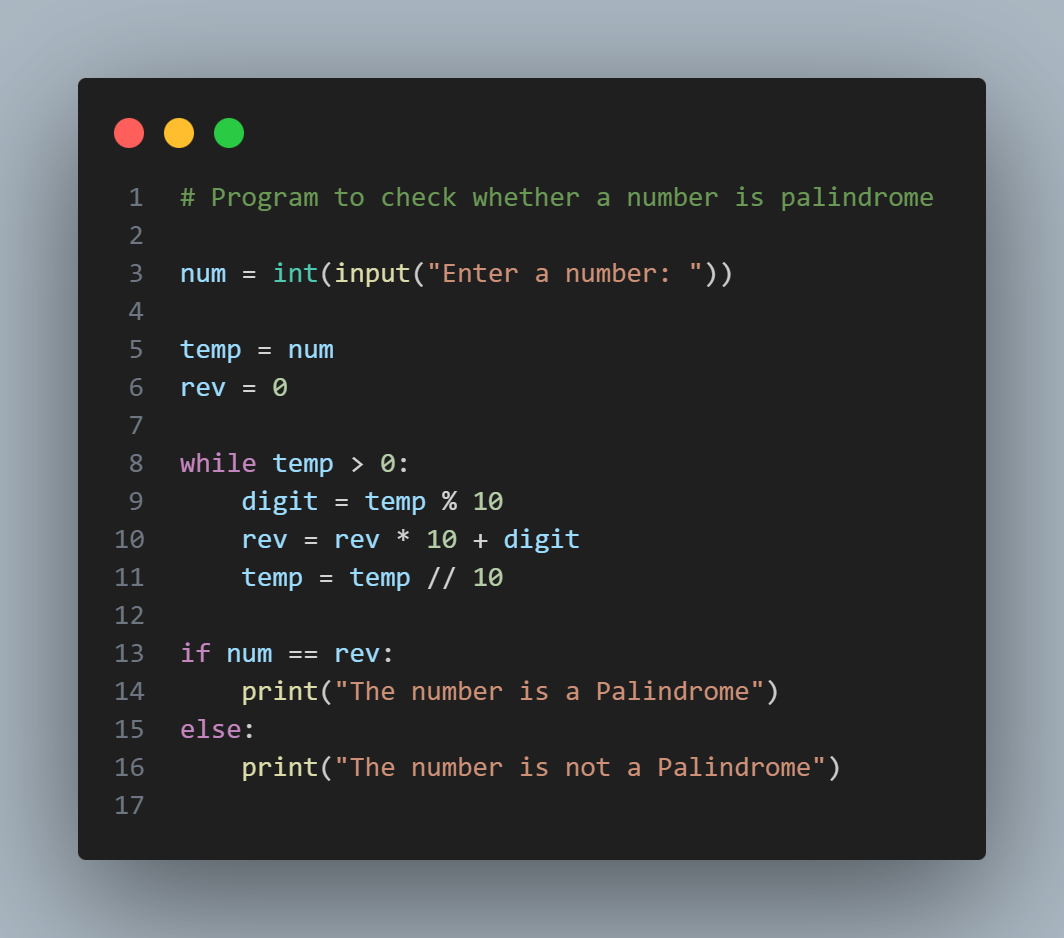
**Output:**

**Question 6:** Write a Python program to check whether a number is a palindrome.

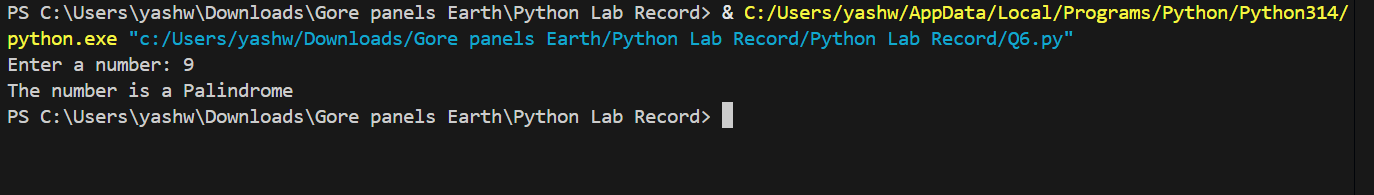
**Algorithm:**

1. Start
2. Read an integer number from the user
3. Store the number in a temporary variable
4. Reverse the number using a loop
5. Compare the reversed number with the original number
6. If both are equal, print that the number is a palindrome
7. Else, print that the number is not a palindrome
8. Stop

**Code:**



**Output:**

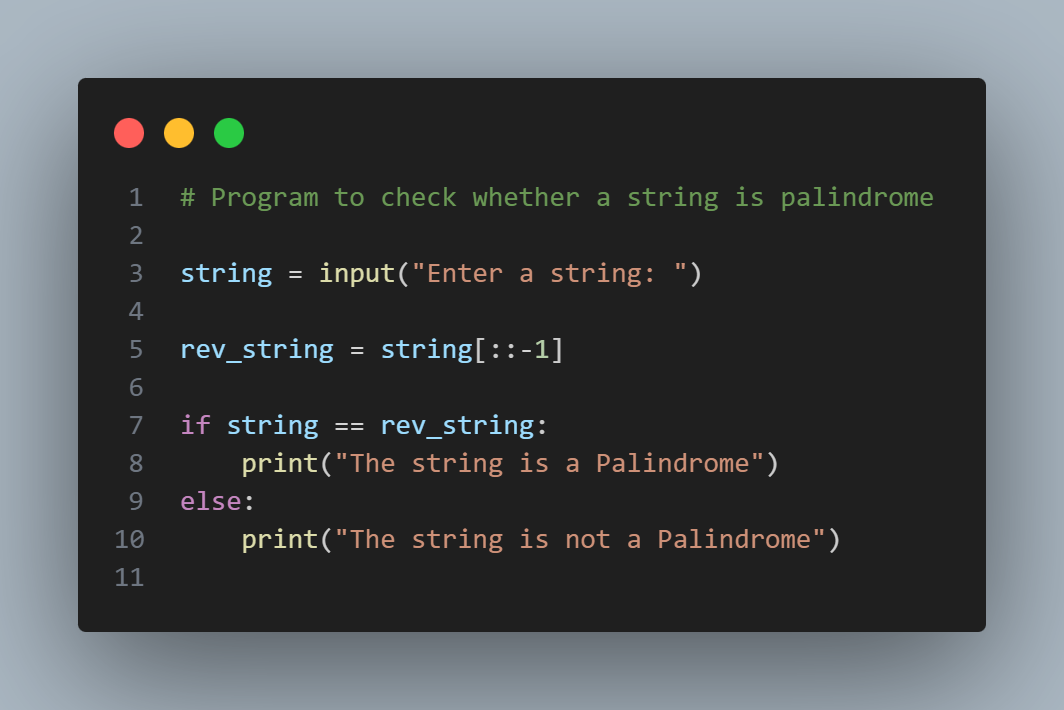


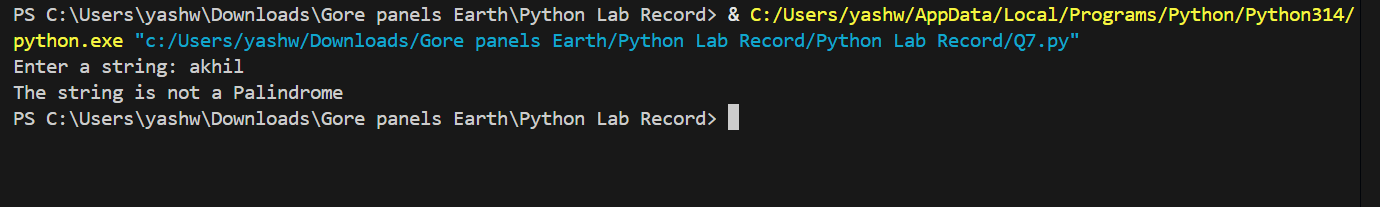
**Question 7:** Write a Python program to check whether a given string is a palindrome.

**Algorithm:**

1. Start
2. Read a string from the user
3. Reverse the string
4. Compare the reversed string with the original string
5. If both are equal, print that the string is a palindrome
6. Else, print that the string is not a palindrome

**Code:**



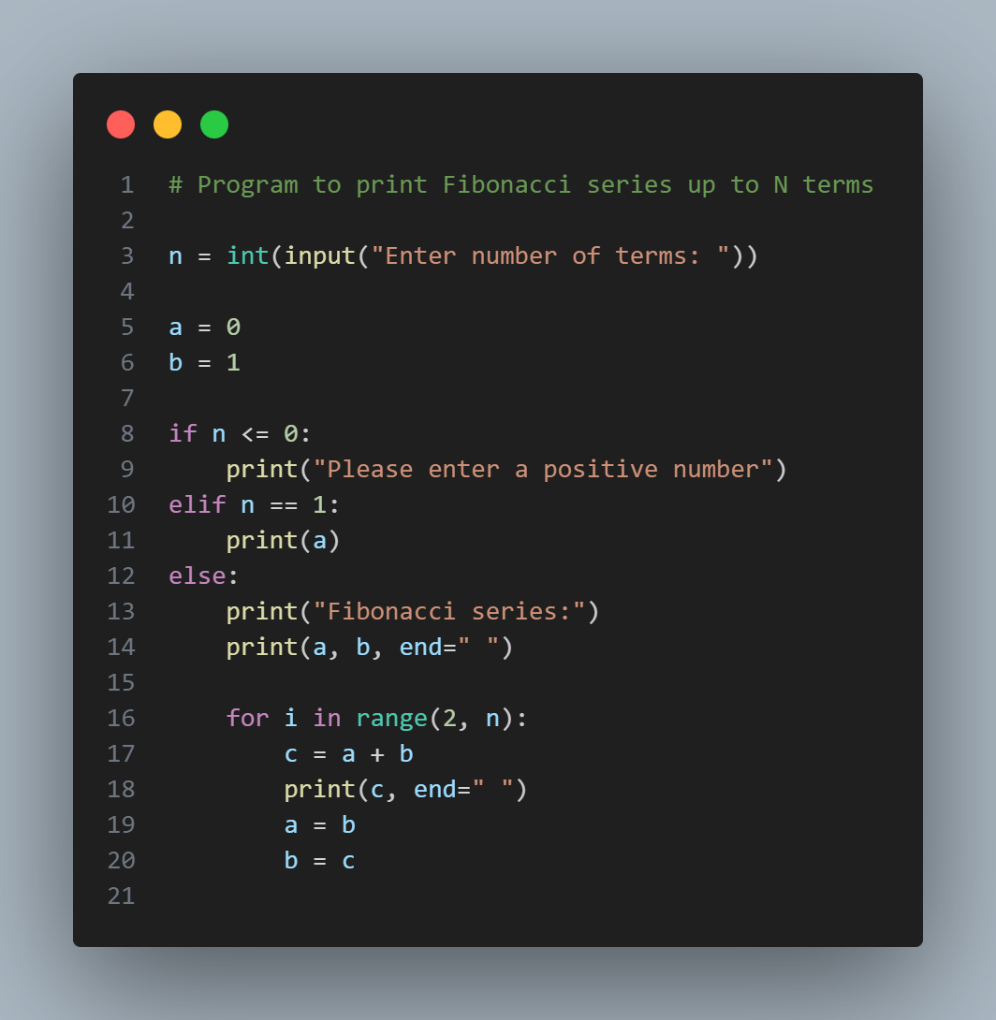
**Output:**

**Week 3:**

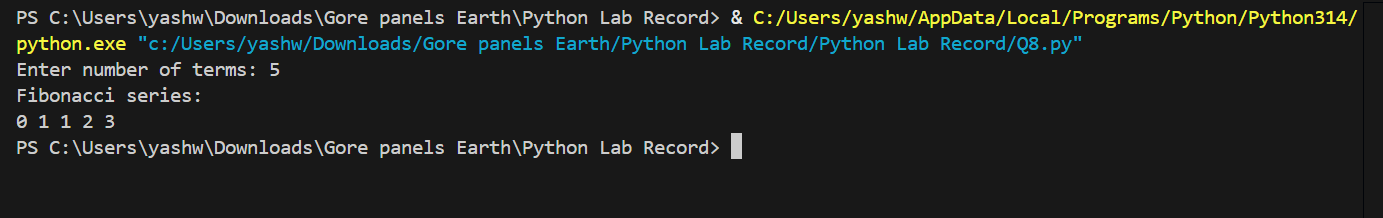
**Question 8:** Write a Python program to print the Fibonacci series up to *N* terms.

**Algorithm:**

1. Start
2. Read the number of terms n from the user
3. Initialize first two numbers as 0 and 1
4. Print the first two numbers
5. Use a loop to generate the next terms by adding the previous two numbers
6. Print each term until n terms are printed
7. Stop

**Code:**

**Output:**

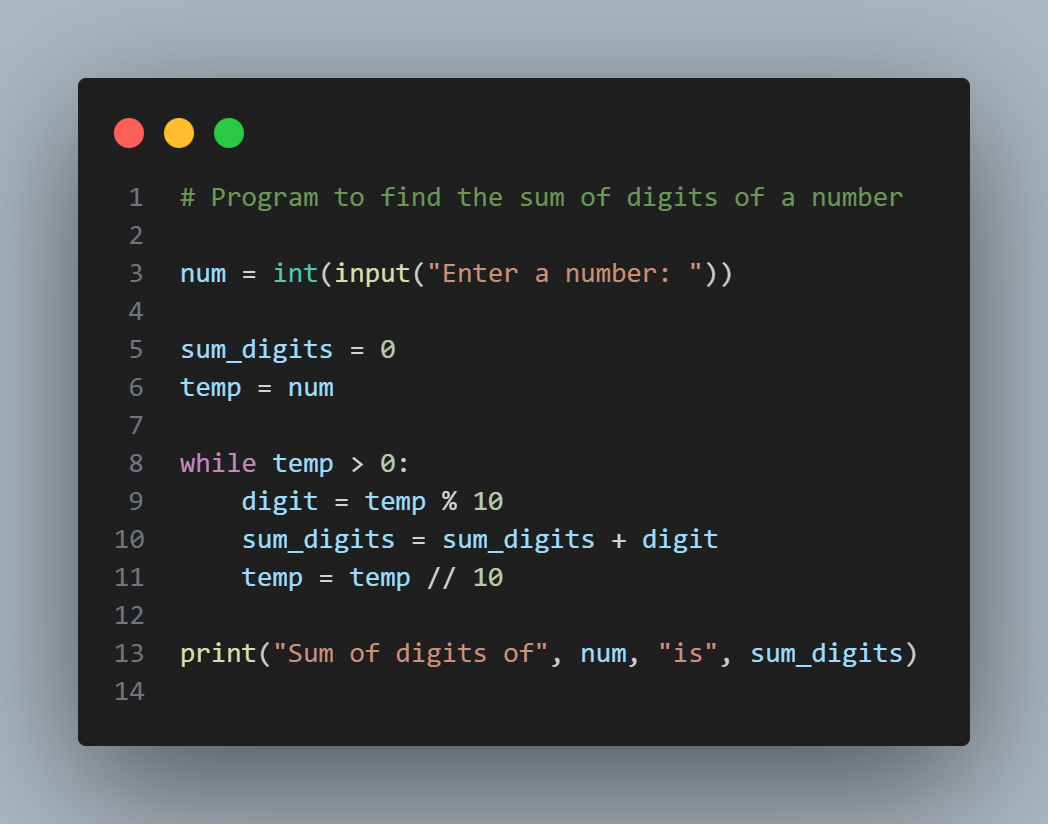


**Question 9:** Write a Python program to find the sum of digits of a number.

**Algorithm:**

1. Start
2. Read an integer number from the user
3. Initialize a variable sum to 0
4. Extract the last digit of the number
5. Add the digit to sum
6. Remove the last digit from the number
7. Repeat steps 4–6 until the number becomes 0
8. Print the sum of digits
9. Stop

**Code:**

****

**Output:**



**Question 10:** Write a Python program to count vowels and consonants in a string.

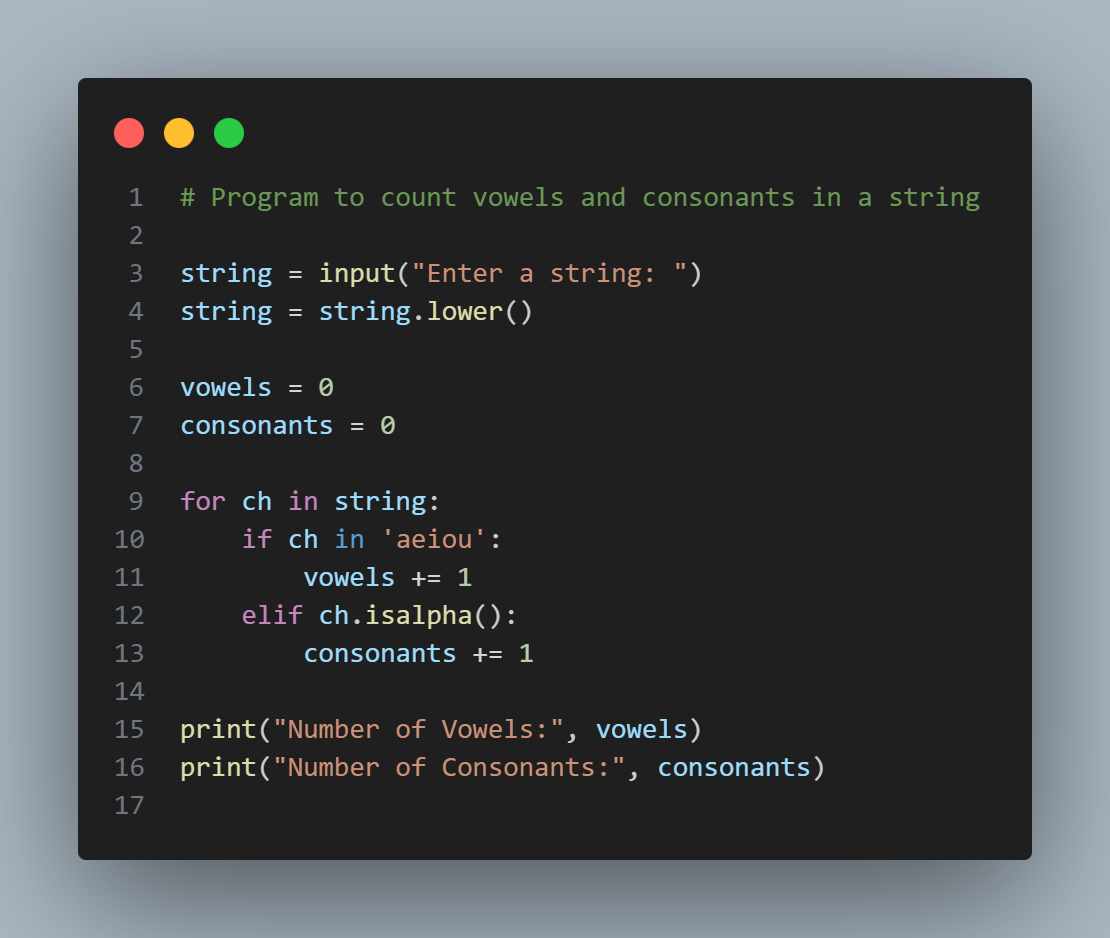
**Algorithm:**

1. Start
2. Read a string from the user
3. Initialize two variables vowels and consonants to 0
4. Convert the string to lowercase
5. For each character in the string:

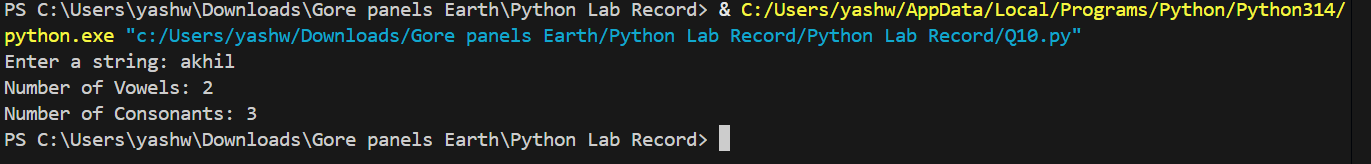
* If the character is a vowel, increment vowel count
* Else if the character is an alphabet, increment consonant count

1. Print the number of vowels and consonants
2. Stop

**Code:**

****

**Output:**

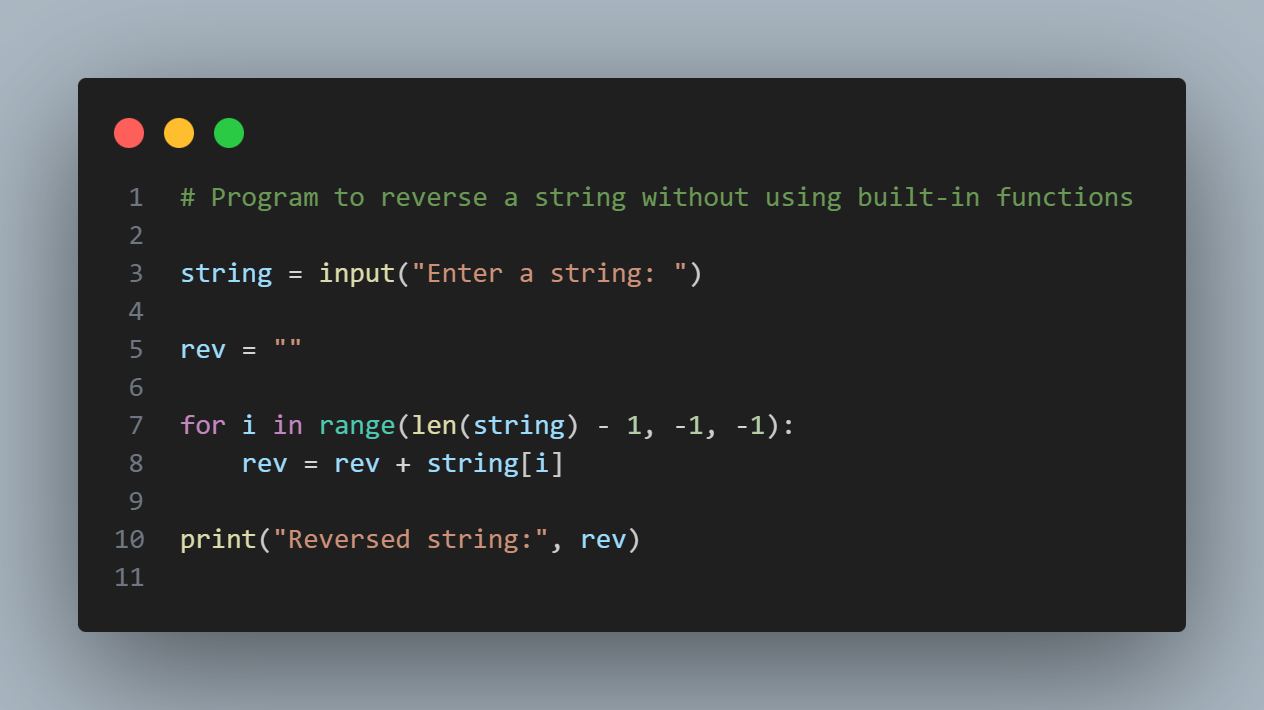


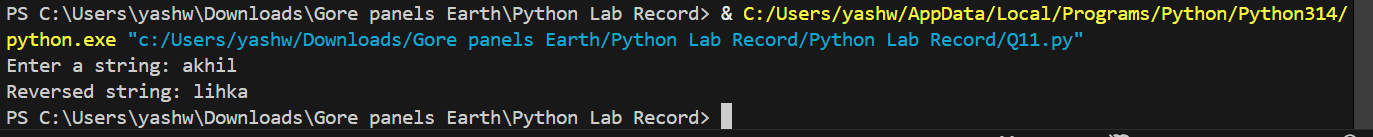
**Week 4:**

**Question 11:** Write a Python program to reverse a string without using built-in functions.

**Algorithm:**

1. Start
2. Read a string from the user
3. Initialize an empty string rev
4. Traverse the string from the last character to the first
5. Append each character to rev
6. Print the reversed string
7. Stop

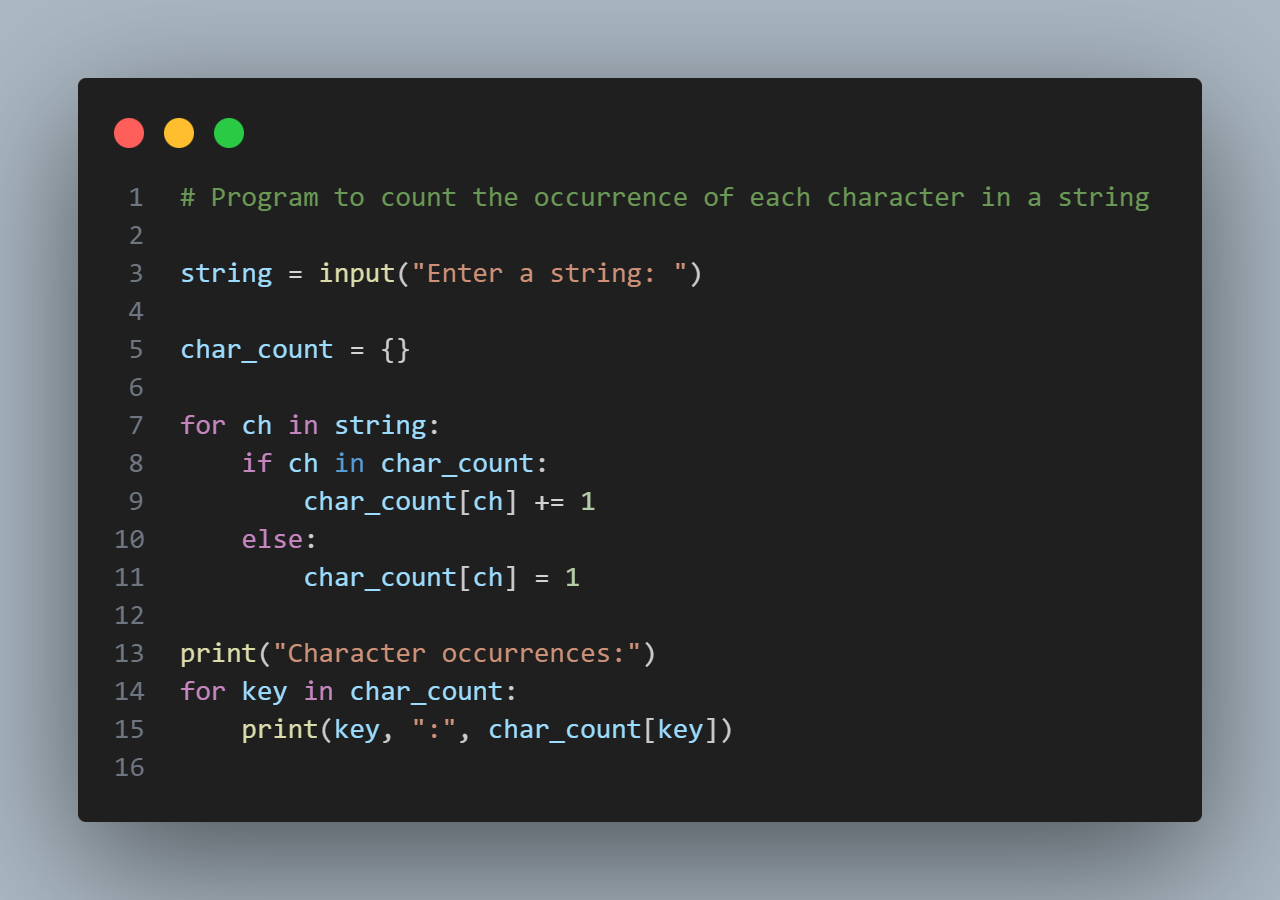
**Code:**

**Output:**

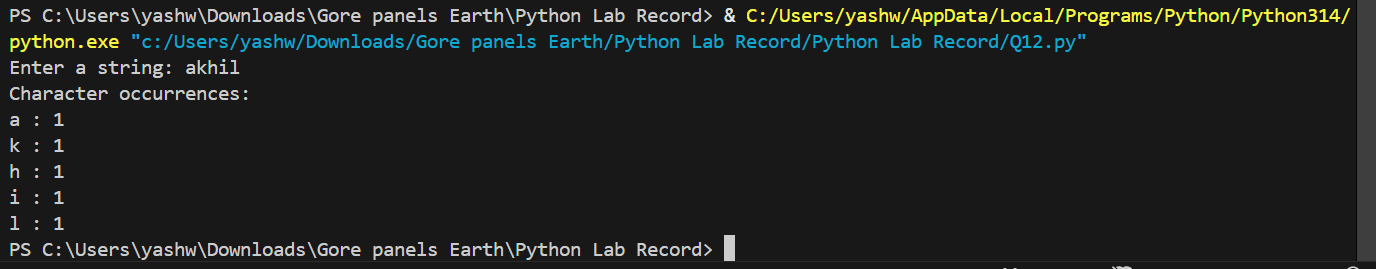
**Question 12:** Write a Python program to count the occurrence of each character in a string.

**Algorithm:**

1. Start
2. Read a string from the user
3. Initialize an empty dictionary
4. For each character in the string:
5. If the character exists in the dictionary, increment its count
6. Else, add the character to the dictionary with count 1
7. Display each character and its count
8. Stop

**Code:**

**Output:**



**Question 13:** Write a Python program to create a simple calculator using conditional statements.

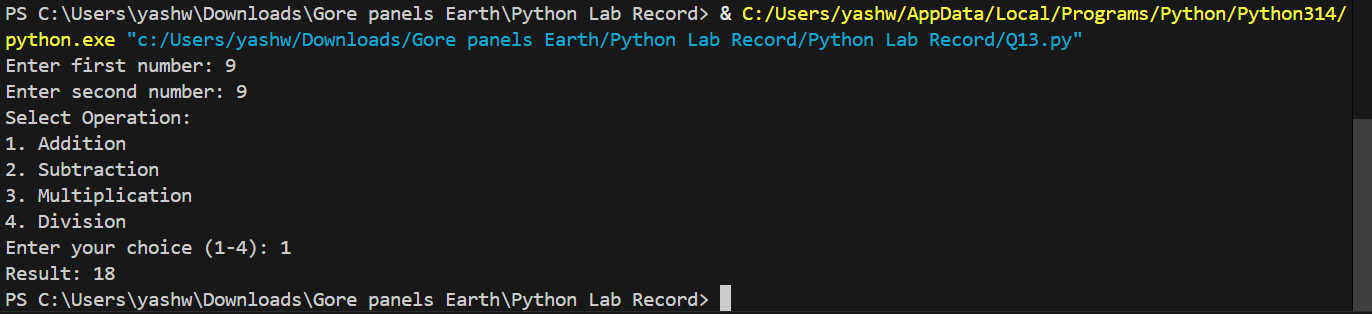
**Algorithm:**

1. Start
2. Read two numbers from the user
3. Display the list of operations (Addition, Subtraction, Multiplication, Division)
4. Read the user’s choice
5. If the choice is addition, add the two numbers and display the result
6. Else if the choice is subtraction, subtract and display the result
7. Else if the choice is multiplication, multiply and display the result
8. Else if the choice is division, divide and display the result
9. Else, display an invalid choice message
10. Stop

**Code:**

****

**Output:**



**Week 5:**

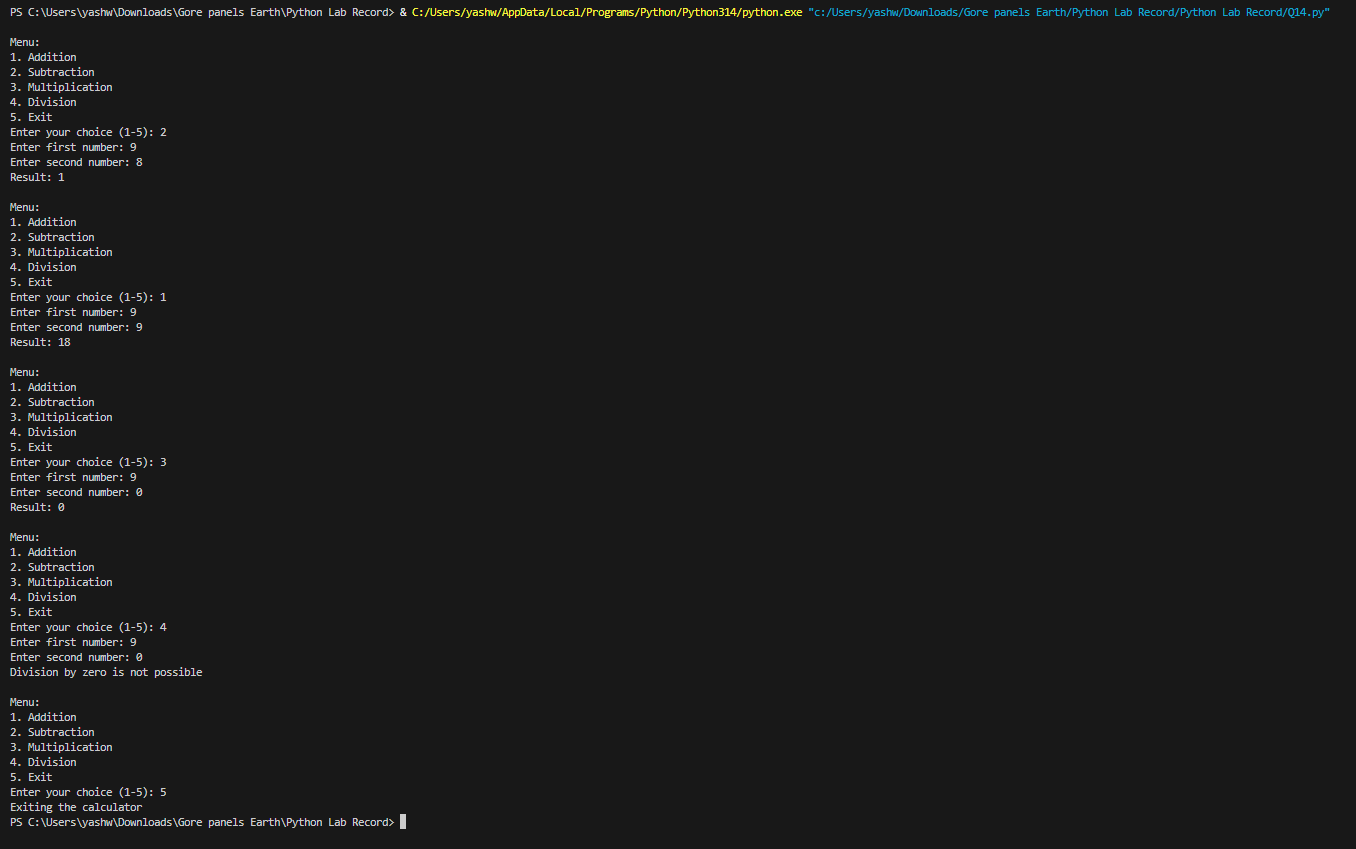
**Question 14:** Write a Python program to implement a menu-driven calculator using a loop (repeat until the user exits).

**Algorithm:**

1. Start
2. Display the menu of operations
3. Read the user’s choice
4. If the choice is exit, terminate the program
5. Else, read two numbers from the user
6. Perform the selected operation using conditional statements
7. Display the result
8. Repeat steps 2–7 until the user exits
9. Stop

**Code:**

****

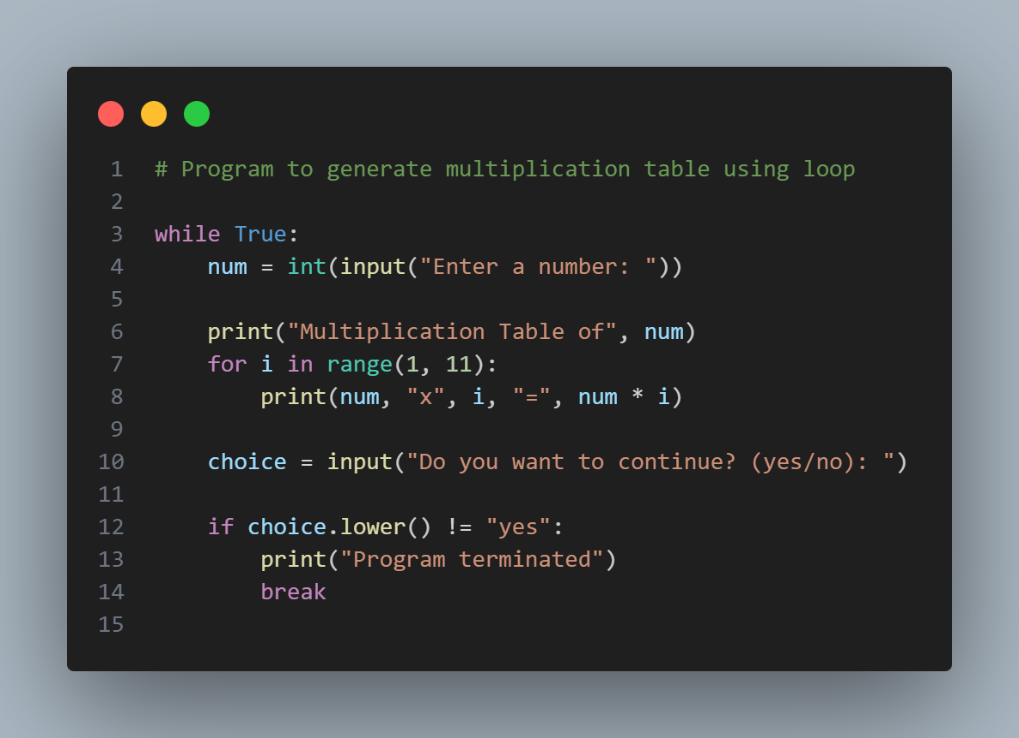
**Output:**

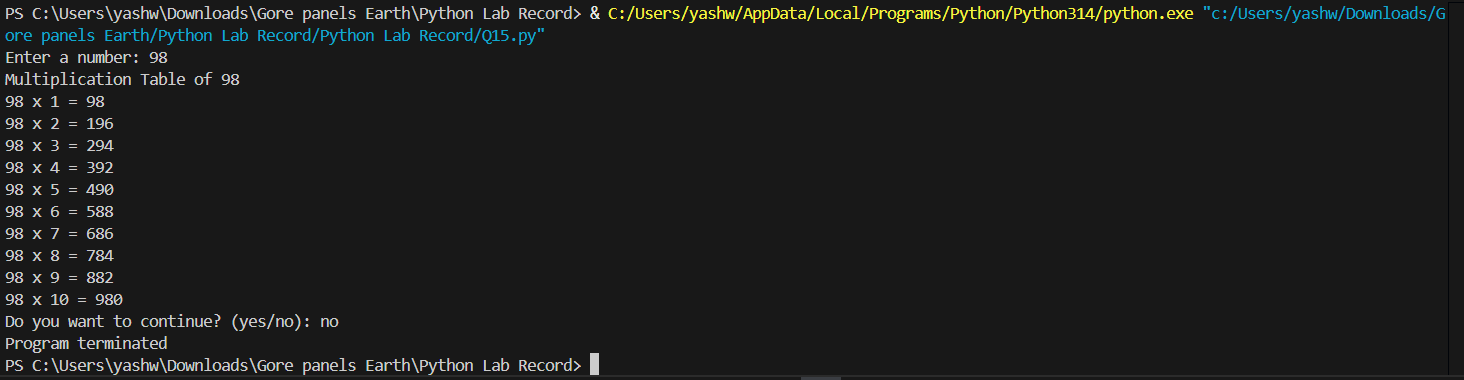
**Question 15:** 15. Write a Python program to generate a multiplication table for a given number (loop until the user stops).

**Algorithm:**

1. Start
2. Repeat the following steps until the user chooses to stop
3. Read a number from the user
4. Use a loop to print the multiplication table of the number from 1 to 10
5. Ask the user whether to continue or stop
6. If the user chooses to stop, exit the loop
7. Stop

**Code:**

****

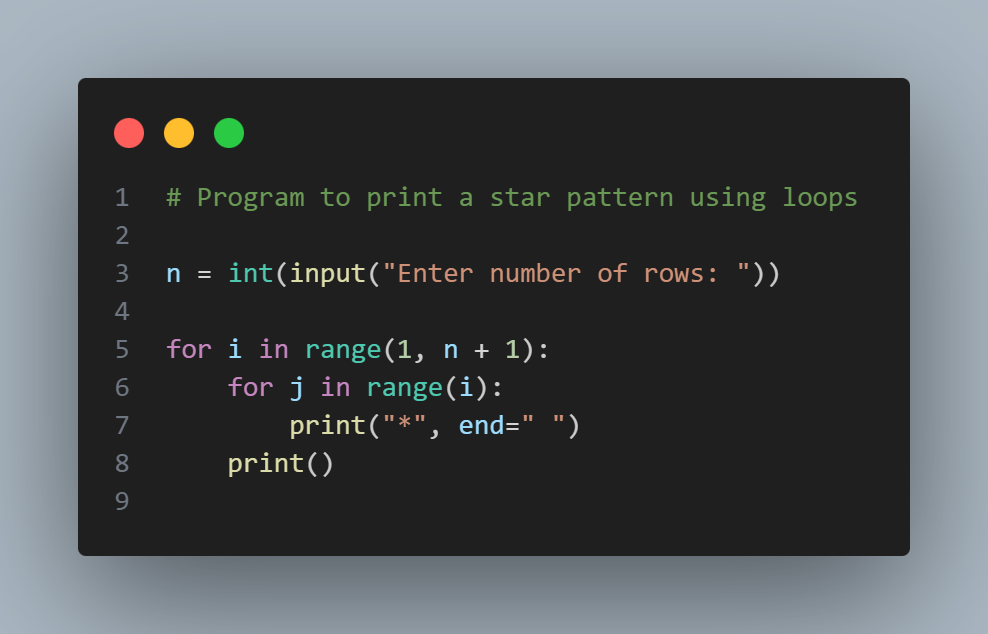
**Output:**

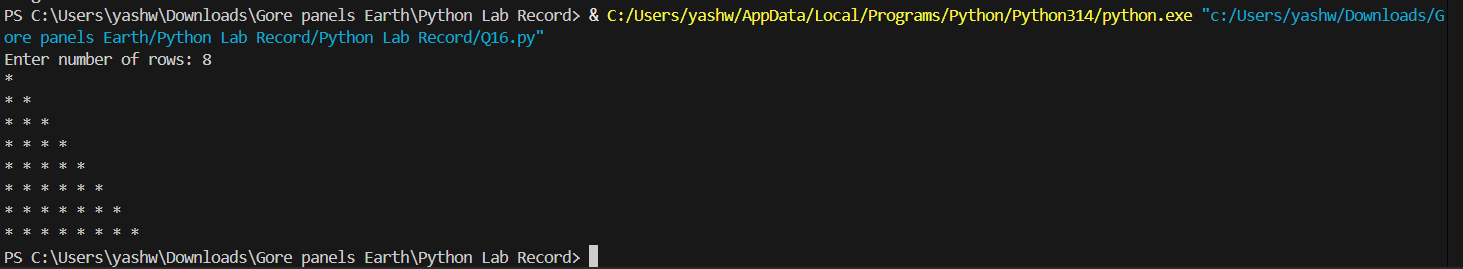
**Question 16:** Write a Python program to print different patterns using loop concepts (e.g., star patterns, number patterns).

**Algorithm:**

1. Start
2. Read the number of rows from the user
3. Use a nested loop to print stars row by row
4. Increment the number of stars in each row
5. Display the pattern
6. Stop

**Code:**

****

**Output:**

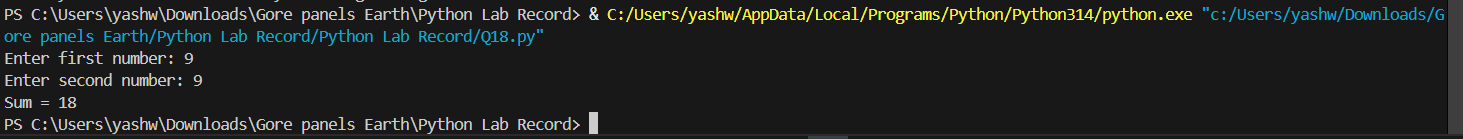
**Week 6:**

**Question 17:** Write a Python function that takes a user’s name and prints a greeting message.

**Algorithm:**

1. Start
2. Define a function that accepts a name as a parameter
3. Inside the function, print a greeting message using the name
4. Read the user’s name
5. Call the function by passing the name
6. Stop

**Code:**

**Output:**

**Question 18:** Write a Python function that accepts two numbers and returns their sum.

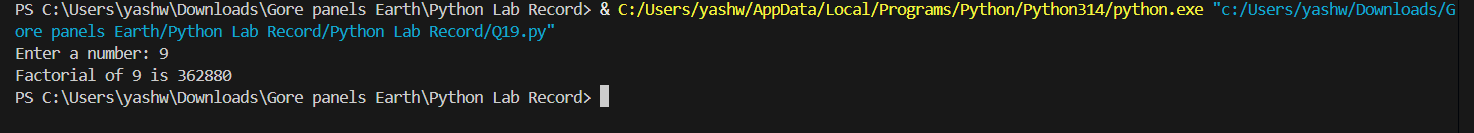
**Algorithm:**

1. Start
2. Define a function that accepts two numbers as parameters
3. Add the two numbers inside the function
4. Return the result
5. Read two numbers from the user
6. Call the function and store the returned value
7. Print the result
8. Stop

**Code:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Output:**

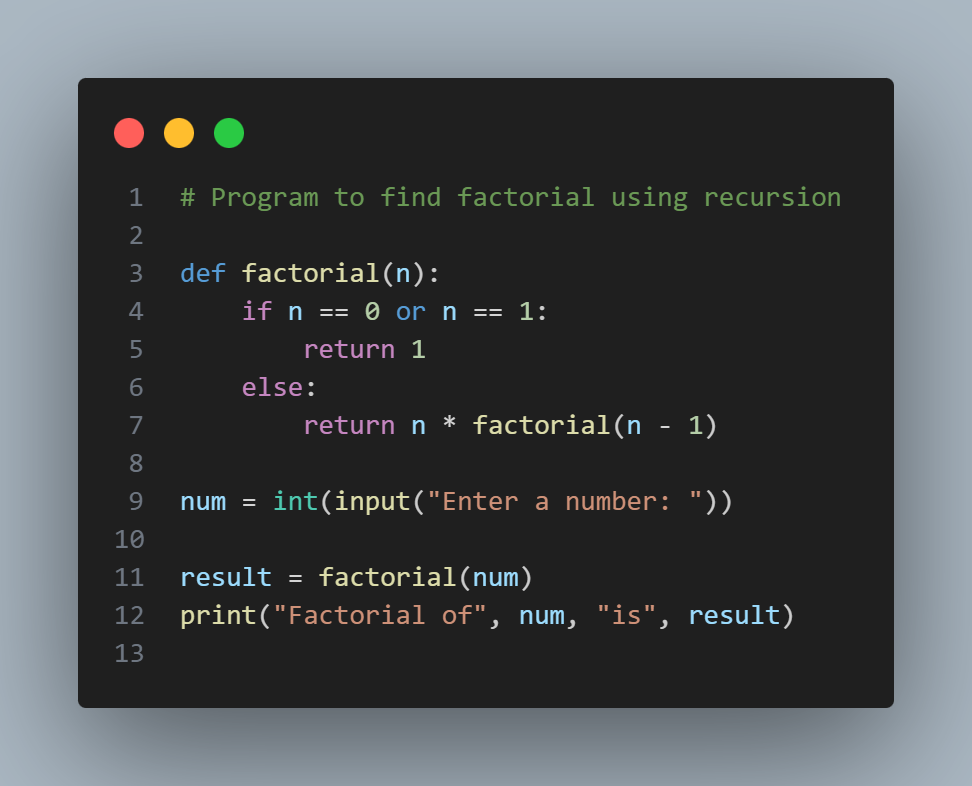
**Week 7:**

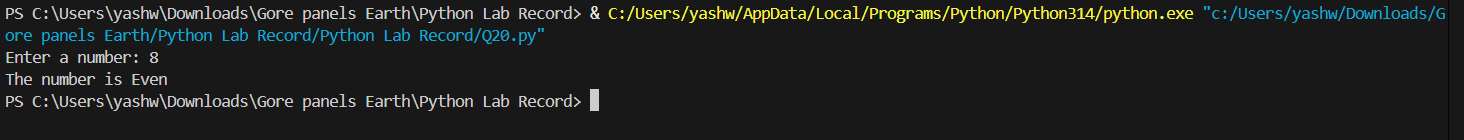
**Question 19:** Write a Python recursive function to find the factorial of a number.

**Algorithm:**

1. Start
2. Define a recursive function to calculate factorial
3. If the number is 0 or 1, return 1
4. Else, return the number multiplied by the factorial of (number − 1)
5. Read a number from the user
6. Call the recursive function
7. Print the result
8. Stop

**Code:**

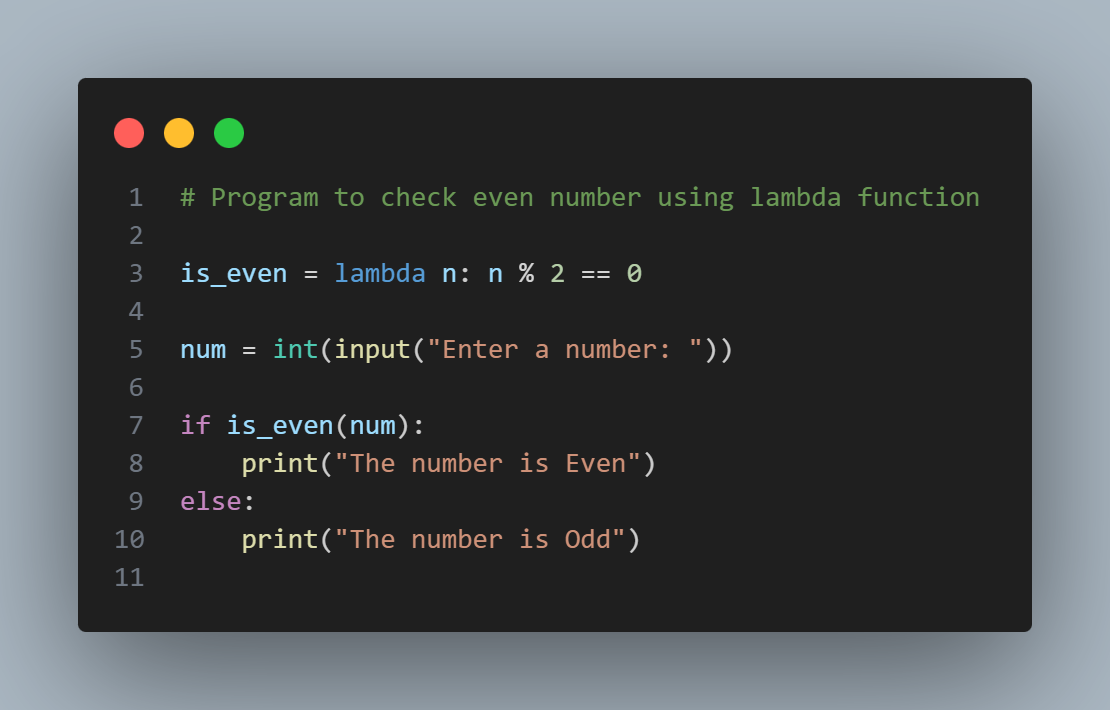
****

**Output:**

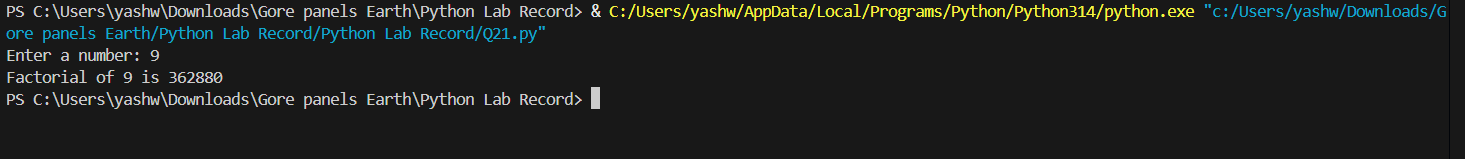
**Question 20:** Write a Python lambda function to check whether a number is even.

**Algorithm:**

1. Start
2. Define a lambda function to check even condition
3. Read a number from the user
4. Pass the number to the lambda function
5. Display whether the number is even or odd
6. Stop

**Code:**

**Output:**

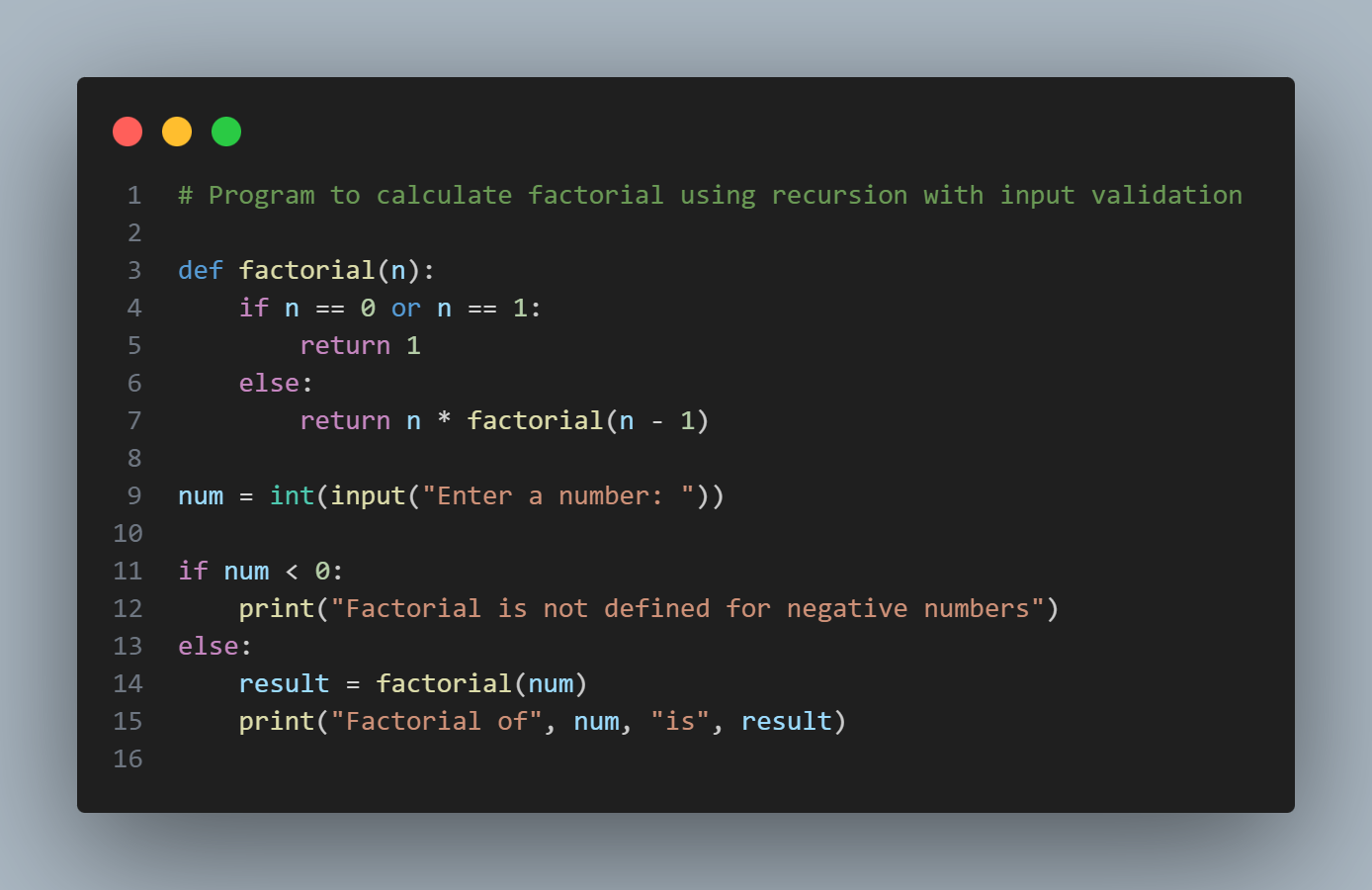


**Question 21:** Write a Python program to calculate factorial using recursion with input validation.

**Algorithm:**

1. Start
2. Define a recursive function to calculate factorial
3. If the number is 0 or 1, return 1
4. Else, return number × factorial (number − 1)
5. Read a number from the user
6. If the number is negative, display an error message
7. Else, call the recursive function and display the result
8. Stop

**Code:**

****

**Output:**

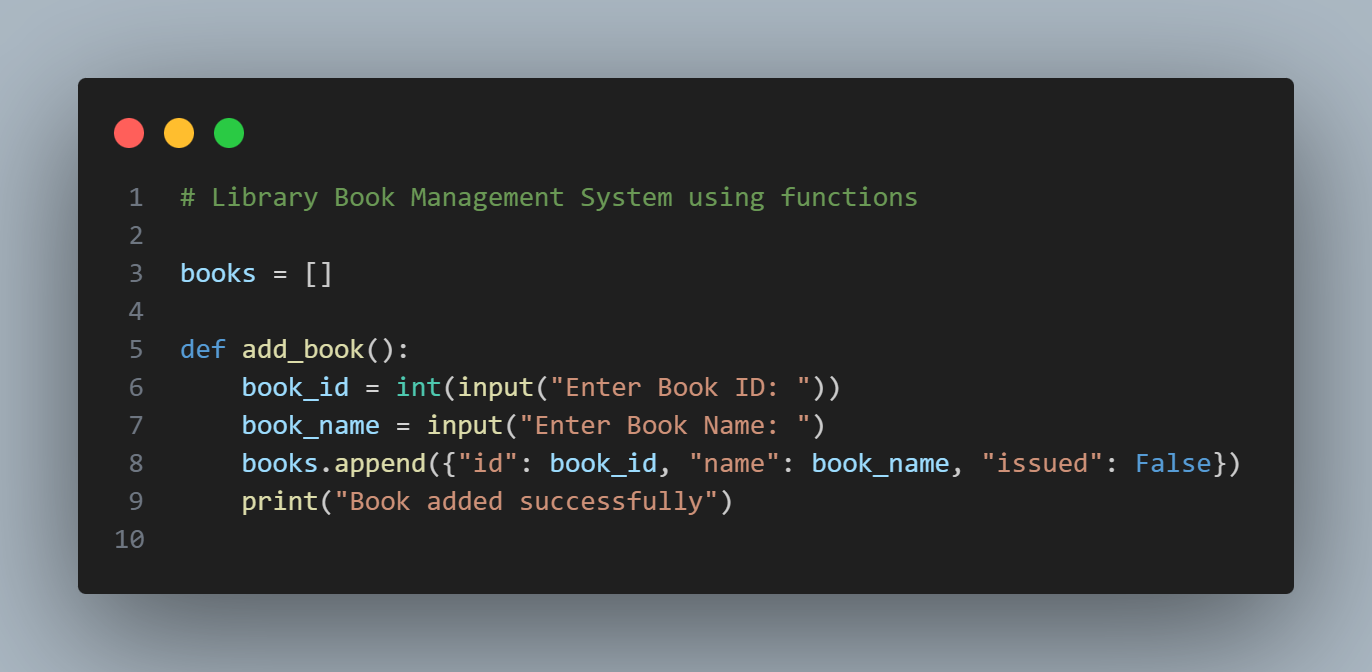
**PROJECT / ADVANCED QUESTIONS**

**Week 8:**

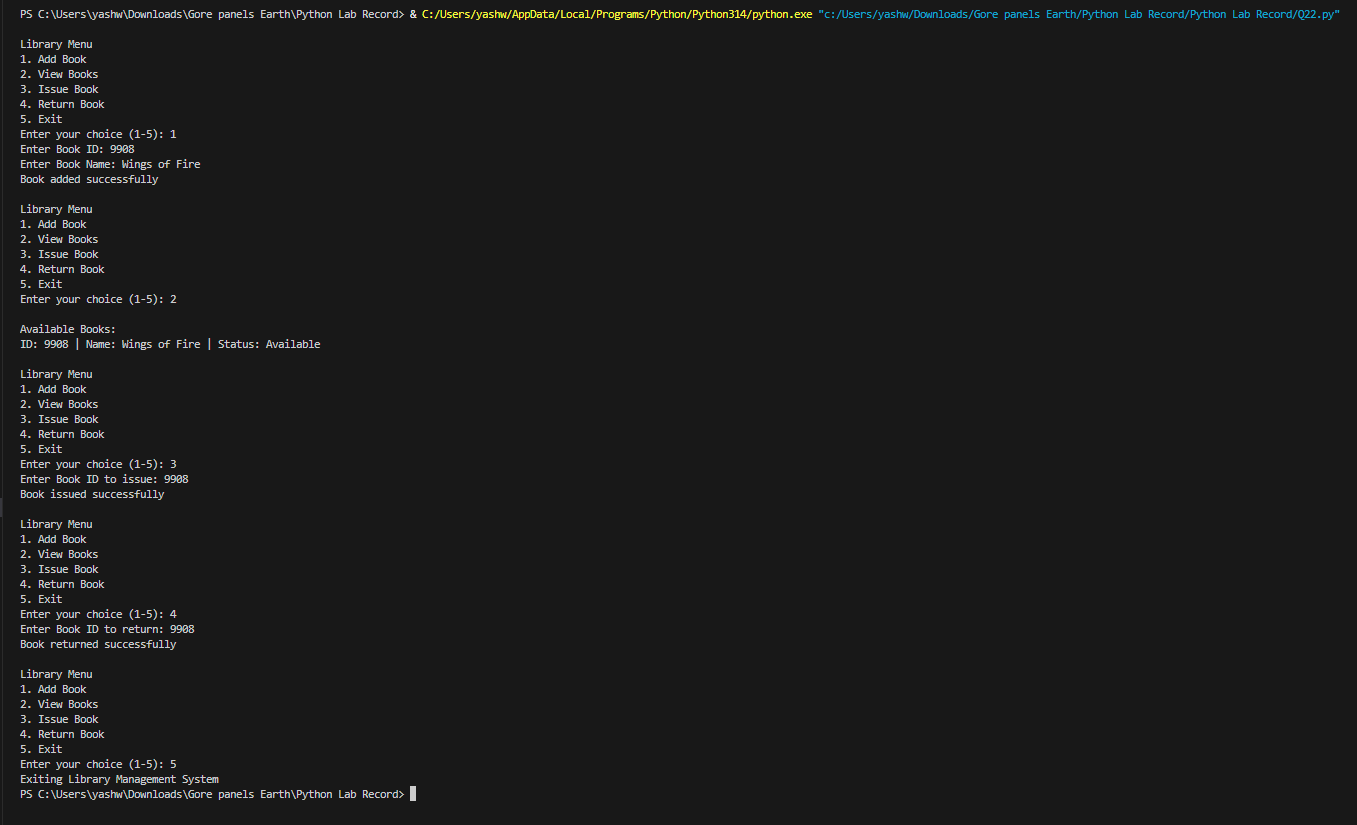
**Question 22:** Write a Python program to create a Library Book Management System using functions.

**Algorithm:**

1. Start
2. Initialize an empty list to store book details
3. Define a function to add books to the library
4. Define a function to display all books
5. Define a function to issue a book
6. Define a function to return a book
7. Display a menu with choices
8. Perform operations based on user choice using functions
9. Repeat the process until the user chooses to exit
10. Stop

**Code:**

****

**Output:**

**Week 9:**

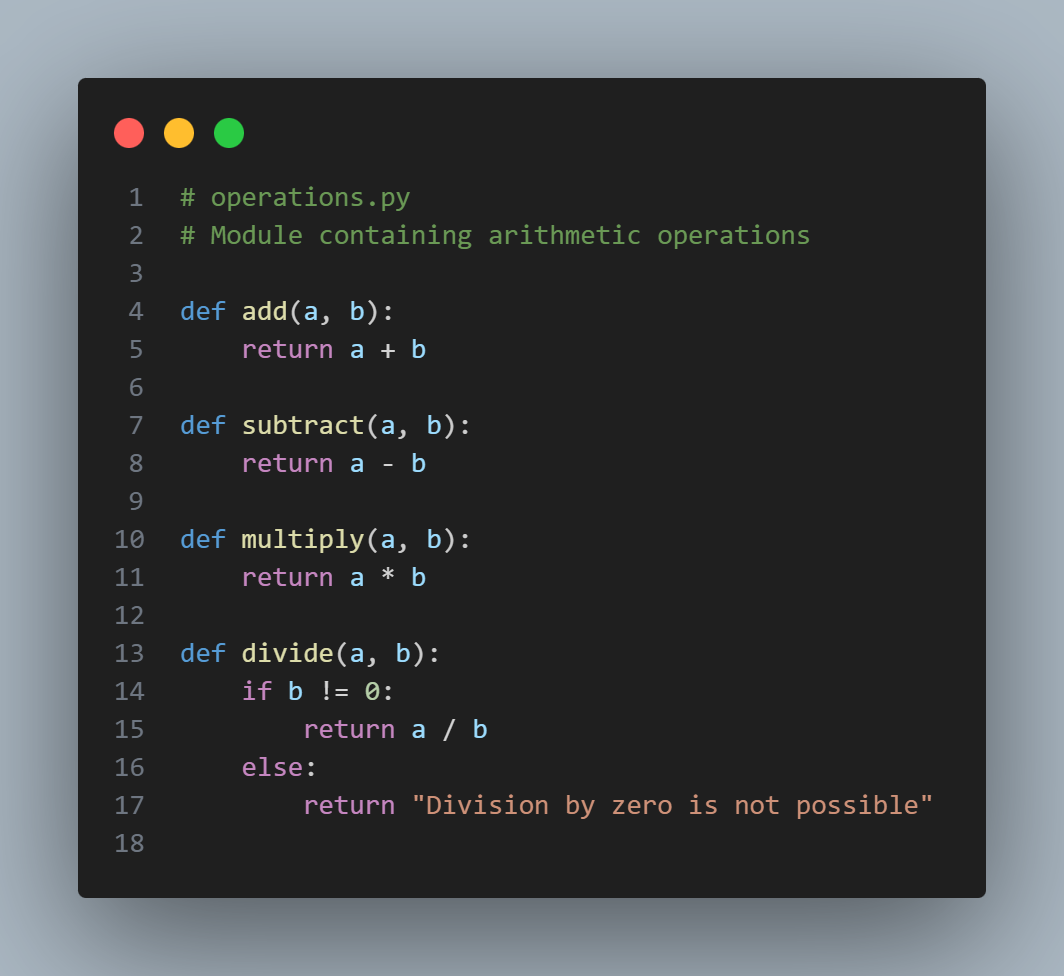
**Question 23:** Write a Python project to build a Calculator using modular programming (separate module for operations).

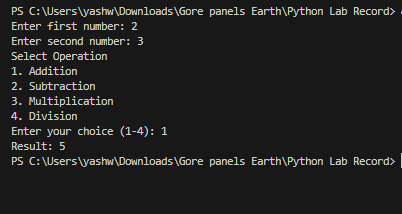
**Algorithm:**

1. Start
2. Create a module containing arithmetic functions
3. Import the module in the main program
4. Read two numbers from the user
5. Display calculator menu
6. Perform the selected operation using module functions
7. Display the result
8. Stop

**Code:**

****

**Output:**



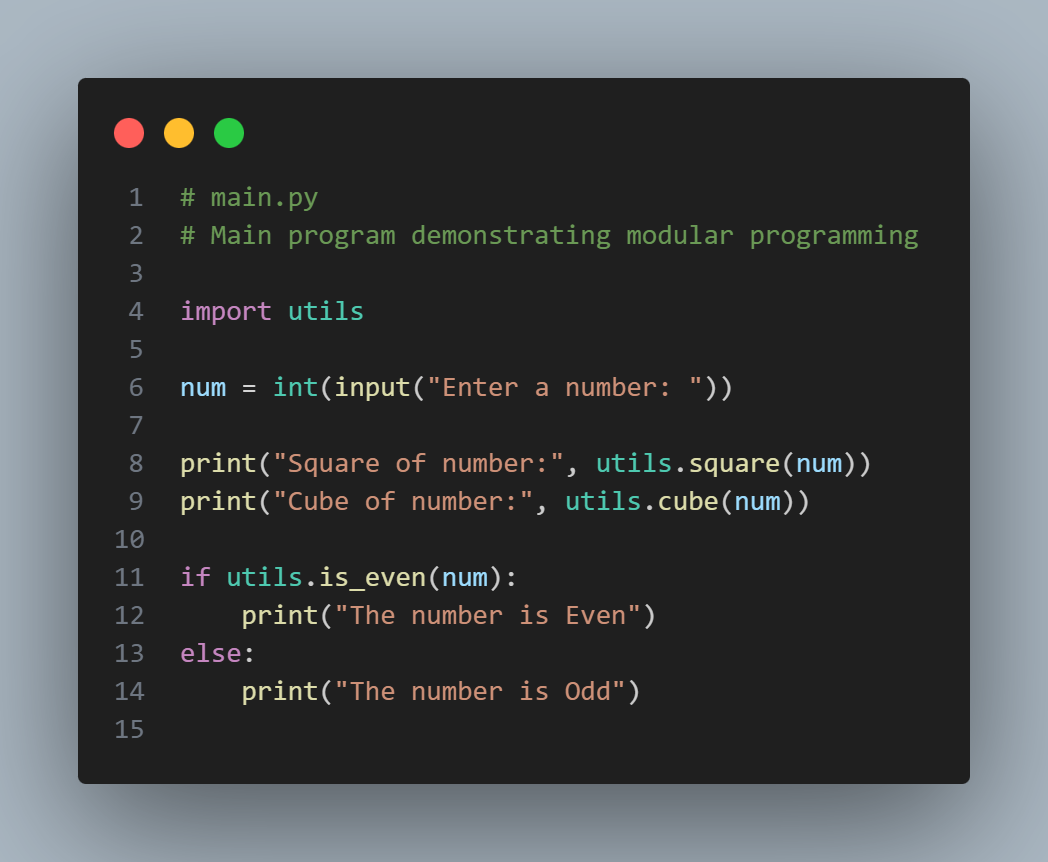
**Week 10:**

**Question 24:** Write a Python program that applies modular programming principles and defines multiple reusable functions.

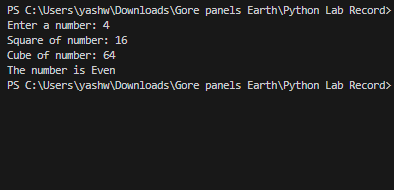
**Algorithm:**

1. Start
2. Create a module containing reusable functions
3. Import the module in the main program
4. Read input values from the user
5. Call the reusable functions as required
6. Display the results
7. Stop

**Code:**

****

**Output:**



**Week 11:**

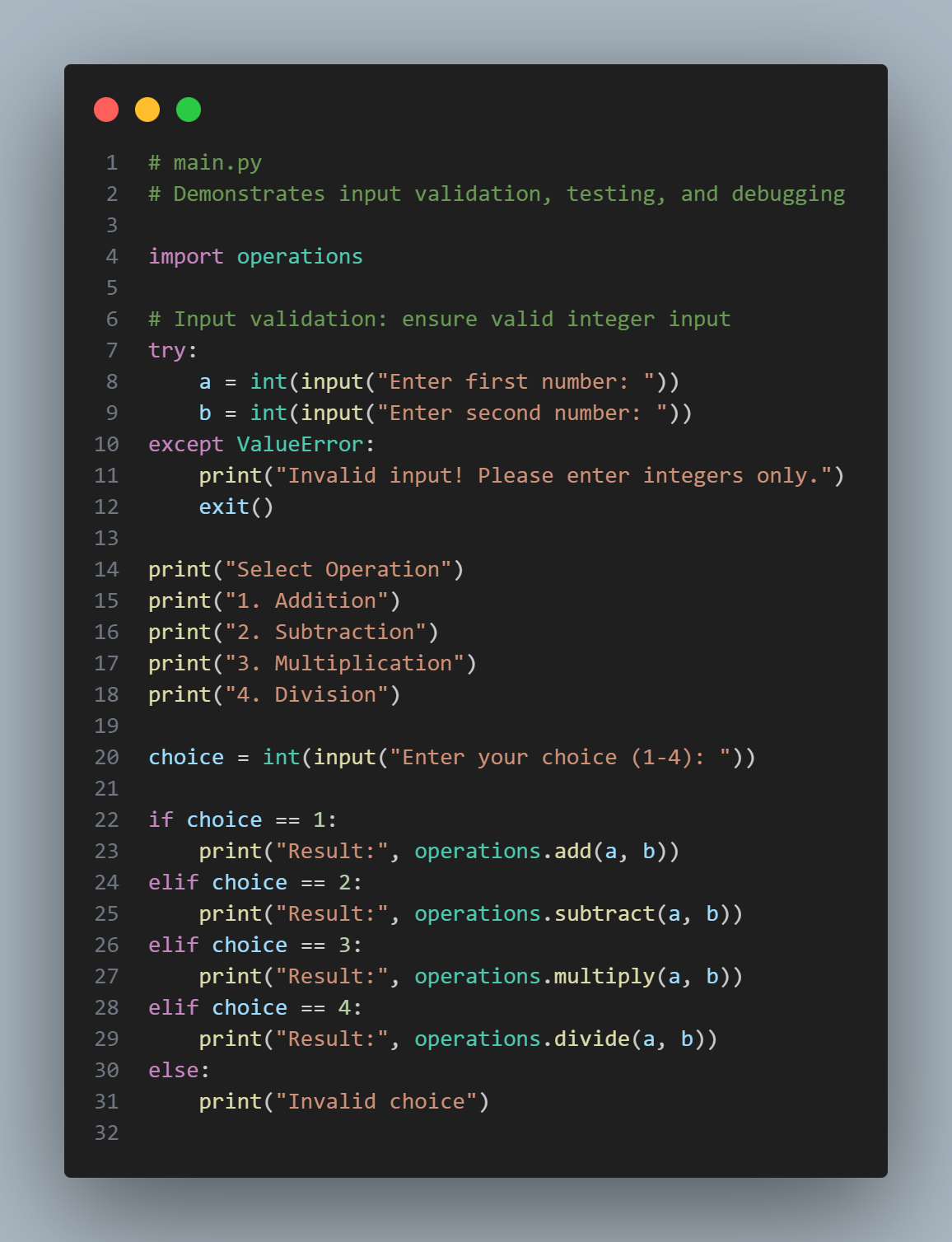
**Question 25:** Write a Python program using modular programming principles and demonstrate:

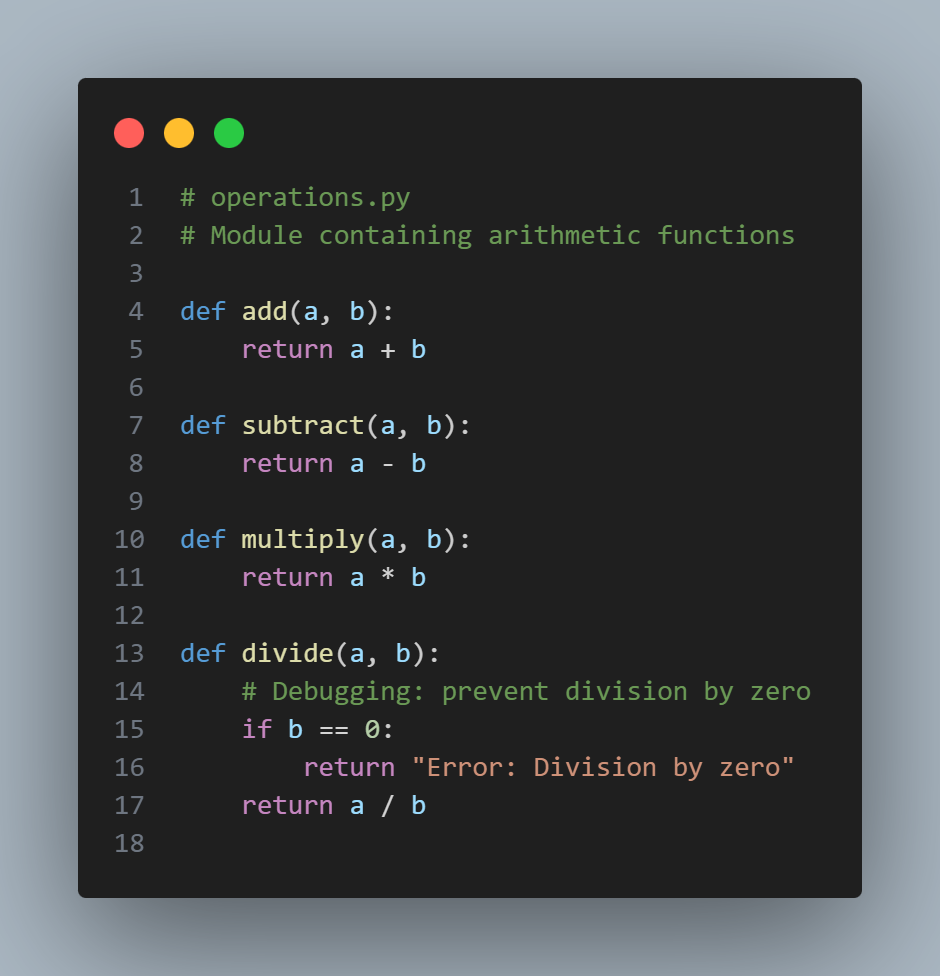
* Input validation
* Testing (minimum 3 test cases)
* Debugging practice with comments

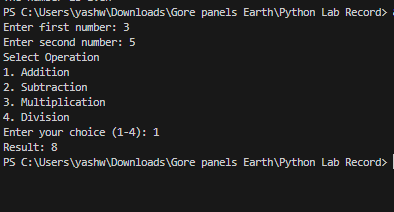
**Algorithm:**

1. Start
2. Create a module with arithmetic functions
3. Import the module into the main program
4. Accept user input and validate it
5. Perform the selected operation
6. Display the result
7. Test the program with multiple test cases
8. Stop

**Code:**

****

**Output:**



**Week 12:**

**Question 26:** Write a Python project for a User Registration System with input validation, testing, and debugging documentation.

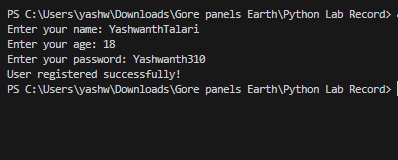
**Algorithm:**

1. Start
2. Create a module to validate user details
3. Import the module in the main program
4. Accept user name, age, and password
5. Validate user inputs
6. Register user if inputs are valid
7. Display success or error message
8. Stop

**Code:**

****

****

**Output:**

**Week 13:**

**Question 27:** Write a mini-project in Python incorporating various programming concepts (loops, functions, lists, modules, validation, testing).

**Algorithm:**

1. Start
2. Initialize an empty list to store student records
3. Define functions to add, view, and search student records
4. Display a menu to the user
5. Perform operations based on user choice
6. Repeat the process until the user exits
7. Stop

**Code:**

****

**Output:**

