

## **PART1 & 2: Addition, Subtraction, Multiplication and Division Program**

Make a file ***criticalthinking\_module1.py***

Copy script in ***criticalthinking\_module1.py***

To run file use command: \$ **python(3) criticalthinking\_module1.py**

**# Beginning of Python Code, copy this line and everything below till End into python file**

#-----

# Name: Aditya Sandhu

# Course: CSC500-1 [Principles of Programming]

# Module: 1

#-----

# Part 1:

# Addition and Subtraction Program

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

sum\_result = num1 + num2

sub\_result = num1 - num2

print(f"The sum of {num1} and {num2} is: {sum\_result}")

print(f"The difference of {num1} and {num2} is: {sub\_result}")

# Part 2:

# Multiplication and Division Program

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

mul\_result = num1 \* num2

if num2 != 0:

div\_result = num1 / num2

else:

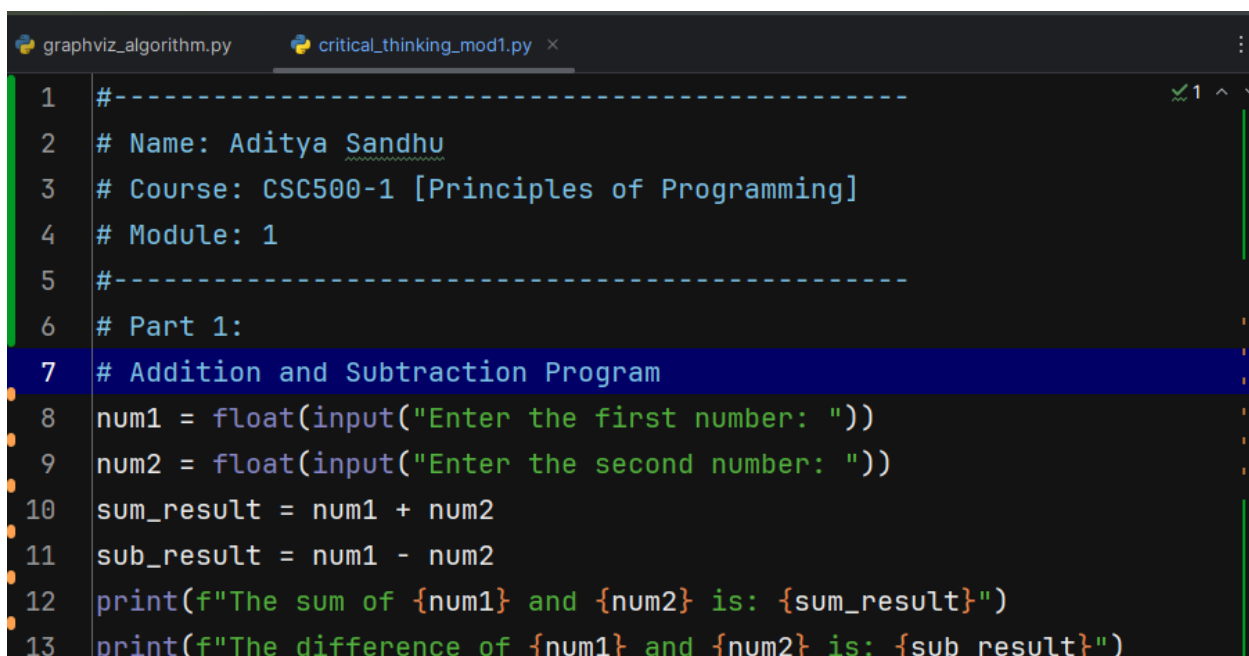
```
div_result = "undefined (cannot divide by zero)"
```

```
print(f"The product of {num1} and {num2} is: {mul_result}")
```

```
print(f"The division of {num1} by {num2} is: {div_result}")
```

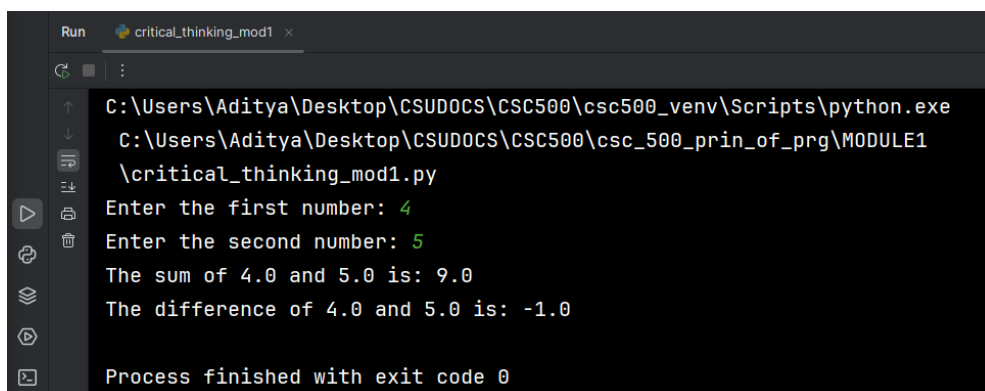
**# End of Python Code, copy this line and everything above into the file**

Figure 1 – Part 1, Python Code for Addition and Subtraction. The code prompts the user to input two numbers and then calculates both the sum and the difference of the numbers. The code performs operations based on user input in Python

A screenshot of a code editor with two tabs: 'graphviz\_algorithm.py' and 'critical\_thinking\_mod1.py'. The 'critical\_thinking\_mod1.py' tab is active, showing a Python script. The script starts with a multi-line comment block containing the author's name 'Aditya Sandhu', course 'CSC500-1 [Principles of Programming]', module '1', and a section header '# Part 1:'. Below the comment is the title '# Addition and Subtraction Program'. The code then prompts the user for two numbers, calculates their sum and difference, and prints the results using f-strings. Line numbers 1 through 13 are visible on the left margin.

```
1 #-----  
2 # Name: Aditya Sandhu  
3 # Course: CSC500-1 [Principles of Programming]  
4 # Module: 1  
5 #-----  
6 # Part 1:  
7 # Addition and Subtraction Program  
8 num1 = float(input("Enter the first number: "))  
9 num2 = float(input("Enter the second number: "))  
10 sum_result = num1 + num2  
11 sub_result = num1 - num2  
12 print(f"The sum of {num1} and {num2} is: {sum_result}")  
13 print(f"The difference of {num1} and {num2} is: {sub_result}")
```

Figure 2 - Execution of Part 1, Python Code for Addition and Subtraction. The output shows the result of the addition and subtraction of the numbers 4 and 5 entered by the user. The sum is 9.0, and the difference is -1.0.

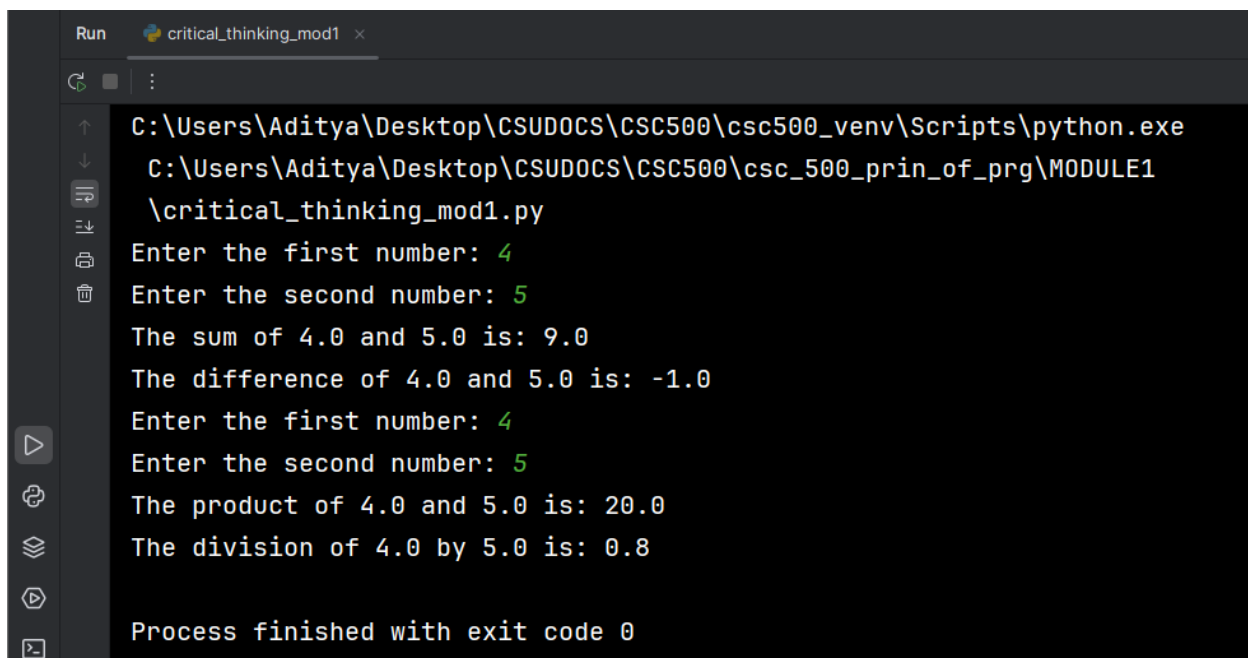
A screenshot of a terminal window titled 'Run critical\_thinking\_mod1'. It shows the execution path of the Python script, the user input for two numbers (4 and 5), the calculated sum (9.0) and difference (-1.0), and the final exit code (0). The terminal output is as follows:

```
C:\Users\Aditya\Desktop\CSUD0CS\CSC500\csc500_venv\Scripts\python.exe  
C:\Users\Aditya\Desktop\CSUD0CS\CSC500\csc_500_prin_of_prg\MODULE1  
\critical_thinking_mod1.py  
Enter the first number: 4  
Enter the second number: 5  
The sum of 4.0 and 5.0 is: 9.0  
The difference of 4.0 and 5.0 is: -1.0  
  
Process finished with exit code 0
```

Figure 3 Python Code for Multiplication and Division. The code prompts the user to input two numbers and calculates both the product and the quotient. It includes error handling for division by zero, providing a user-friendly message when the second number is zero.

```
14 # Part 2:
15 # Multiplication and Division Program
16 num1 = float(input("Enter the first number: "))
17 num2 = float(input("Enter the second number: "))
18 mul_result = num1 * num2
19 if num2 != 0:
20     div_result = num1 / num2
21 else:
22     div_result = "undefined (cannot divide by zero)"
23 print(f"The product of {num1} and {num2} is: {mul_result}")
24 print(f"The division of {num1} by {num2} is: {div_result}")
```

Figure 4 Execution of Python Code for Multiplication, and Division. The output shows the results of multiplication, and division of the numbers 4 and 5 entered by the user. The product is 20.0, and the quotient is 0.8.



```
Run critical_thinking_mod1 x
C:\Users\Aditya\Desktop\CSUD0CS\CSC500\csc500_venv\Scripts\python.exe
C:\Users\Aditya\Desktop\CSUD0CS\CSC500\csc_500_prin_of_prg\MODULE1
\critical_thinking_mod1.py
Enter the first number: 4
Enter the second number: 5
The sum of 4.0 and 5.0 is: 9.0
The difference of 4.0 and 5.0 is: -1.0
Enter the first number: 4
Enter the second number: 5
The product of 4.0 and 5.0 is: 20.0
The division of 4.0 by 5.0 is: 0.8
Process finished with exit code 0
```

The first part of the program focuses on the addition and subtraction of two user-provided numbers. This section is critical as it lays the groundwork for understanding how Python handles user input and performs basic arithmetic operations. The program begins by prompting the user to enter two numbers. These prompts are executed using the 'input()',

function, which reads input from the user as a string. This user interaction is crucial for making the program dynamic and responsive.

After converting the input strings to floating-point numbers, the program proceeds to perform the addition and subtraction operations. The addition operation is carried out using the '+' operator, which sums the two numbers and stores the result in the variable 'sum\_result'. Similarly, the subtraction operation is performed using the '-' operator, and the result is stored in the variable 'sub\_result'. These operations demonstrate Python's capability to handle floating-point arithmetic seamlessly. The choice of floating-point numbers is deliberate, as it allows the program to manage a broader range of numeric values, including decimals, which are often encountered in real-world scenarios.

The results of the addition and subtraction operations are then displayed to the user using formatted string literals, also known as f-strings. F-strings provide a concise and readable way to include variable values directly within a string. By using f-strings, the program constructs output messages that clearly convey the results of the arithmetic operations to the user. This part of the program exemplifies how to present information to the user in a clear and informative manner. The use of f-strings not only enhances readability but also ensures that the output is easily interpretable by incorporating the input values and results in a single coherent message.

The second part of the program extends the arithmetic operations to include multiplication and division. This section introduces additional complexity by incorporating conditional logic to handle potential division by zero errors. Similar to the first part, the program begins by prompting the user to enter two numbers, which are then converted to floating-point numbers. This consistency in input handling ensures that the user experience remains uniform across different arithmetic operations.

The multiplication operation is straightforward and is performed using the '\*' operator. The result of multiplying the two numbers is stored in the variable 'mul\_result'. This operation highlights Python's ability to handle basic arithmetic operations efficiently.

The division operation, however, requires special attention. Division by zero is undefined in mathematics and would cause a runtime error if attempted in the program. To prevent this, the program uses an 'if-else' statement to check whether the second number (the divisor) is zero. If the divisor is not zero, the division operation is performed using the '/' operator, and the result is stored in the variable 'div\_result'. If the divisor is zero, the program sets 'div\_result' to a descriptive string indicating that division by zero is undefined. This conditional logic is crucial for ensuring the robustness and reliability of the program. By

handling potential errors gracefully, the program enhances user experience and prevents unexpected crashes. As with the first part, the results of the multiplication and division operations are displayed to the user using f-strings. The output messages clearly convey the results of the operations, ensuring that the user can easily interpret the information.

The inclusion of a user-friendly message for division by zero demonstrates good programming practice. It shows how to anticipate and handle potential errors, providing a more resilient and user-friendly application. The output of the program, as observed in the execution, demonstrates the successful implementation of both parts. The program accurately performs the specified arithmetic operations and displays the results in a clear and informative manner.

The output messages for the addition and subtraction operations are:

Enter the first number: 4

Enter the second number: 5

The sum of 4.0 and 5.0 is: 9.0

The difference of 4.0 and 5.0 is: -1.0

The output messages for the multiplication and division operations are:

Enter the first number: 4

Enter the second number: 5

The product of 4.0 and 5.0 is: 20.0

The division of 4.0 by 5.0 is: 0.8