Module: FC723

Groub: C

Module Title: Programming Theory

Assessment Title: Portfolio Project 1

Tutor Name: Sophie Norman

Student GUID Number: 3075614

P Number: P461989

Date of Submission: 26th January 2025 at 23:59

I confirm that this assignment is my own work. Where I have referred to academic sources, I have provided in-text citations and included the sources in the final reference list.

The Git Repository:

<https://github.com/Hydro-XD10/FC723.git>

3075614, FC723, Sophie Norman

* **The Pseudocode of the GCD function:**

This was the initial pseudocode.

1-define function take input The first number, the second number to find the GCD

2-check if the first number or the second number equal to zero. if yes we return the other

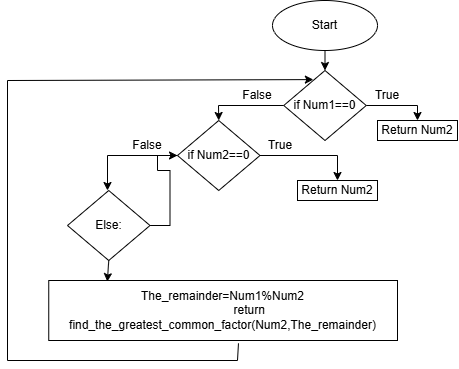
3-if not we use the function again using recursion concept.

The pseudocode was not very accurate, but it gave me a path to work on to know in what I should work on next. The pseudocode in the git repository is edited one after spotting the order of the inputs problem.

3075614, FC723, Sophie Norman

* **How does the code works:**

**The GCD function:**



The basic idea of the code is to divide the reminder by the second number until one of them is zero so the answer will be the other number following the Euclidean Algorithm.

I applied the idea using the recursion concept which I learned in the week 5 in FC724 when we were asked to convert a nested list to flatten list.

After using the pseudocode to write the code I had some issues with the way that I implement the Euclidean Algorithm like the order of putting the numbers.

After that I refactored the code to put the function in an appropriate class.

3075614, FC723, Sophie Norman

if isinstance(Num1 , str) or isinstance(Num2, str):

raise ValueError("The Input Should Be Intger Or Float")

if Num1==0 and Num2==0:

raise ValueError("The GCD between 0 and 0 is mathematically undefined")

This part of code above is to show a error message when invalid input is given like string or GCD of 0 and 0 which is not possible.

* **The Pseudocode of the LCM function:**

To find the Least common multiple(LCM) between two number

1- Get the first number and the second number a and b

2- Check if one of them is zero (invalid input) if yes raise a valuerorr with a message

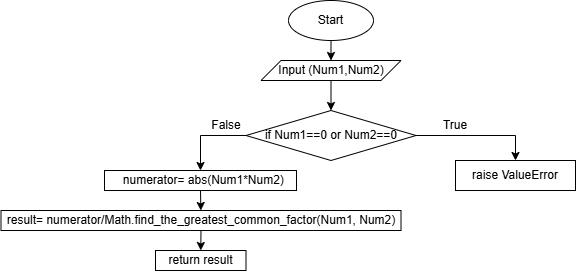
3- Get the absolute value of the multiplication of the a and b

4- Divide it by the GCD of a and B

5- Print result

3075614, FC723, Sophie Norman

**The LCM function(The extension):**



To find the Least common multiple we have to get the absolute value of the multiplication of the two numbers then divide it on by the GCD of the two numbers. If one of the number is zero it will be not defined because you can not multiple a zero.

Error message will be given when one of the numbers is zero because it will be not defined.

3075614, FC723, Sophie Norman

* Unit Test for the function:

I used unit test to test the function which we learned in FC723 Week 10. I write the following code to test the function.

import the\_euclidean\_algorithm\_code\_refactored as ss

def test\_find\_the\_greatest\_common\_factor():

GCD= ss.Math.find\_the\_greatest\_common\_factor(0, 3)

assert GCD== 3, "The greatest common factor"

This code shows the expected output of the find\_the\_greatest\_common\_factor function when we put 0 and 3 as parameters, which means finding the greatest common factor for 0 and 3. And the expected output is 3.

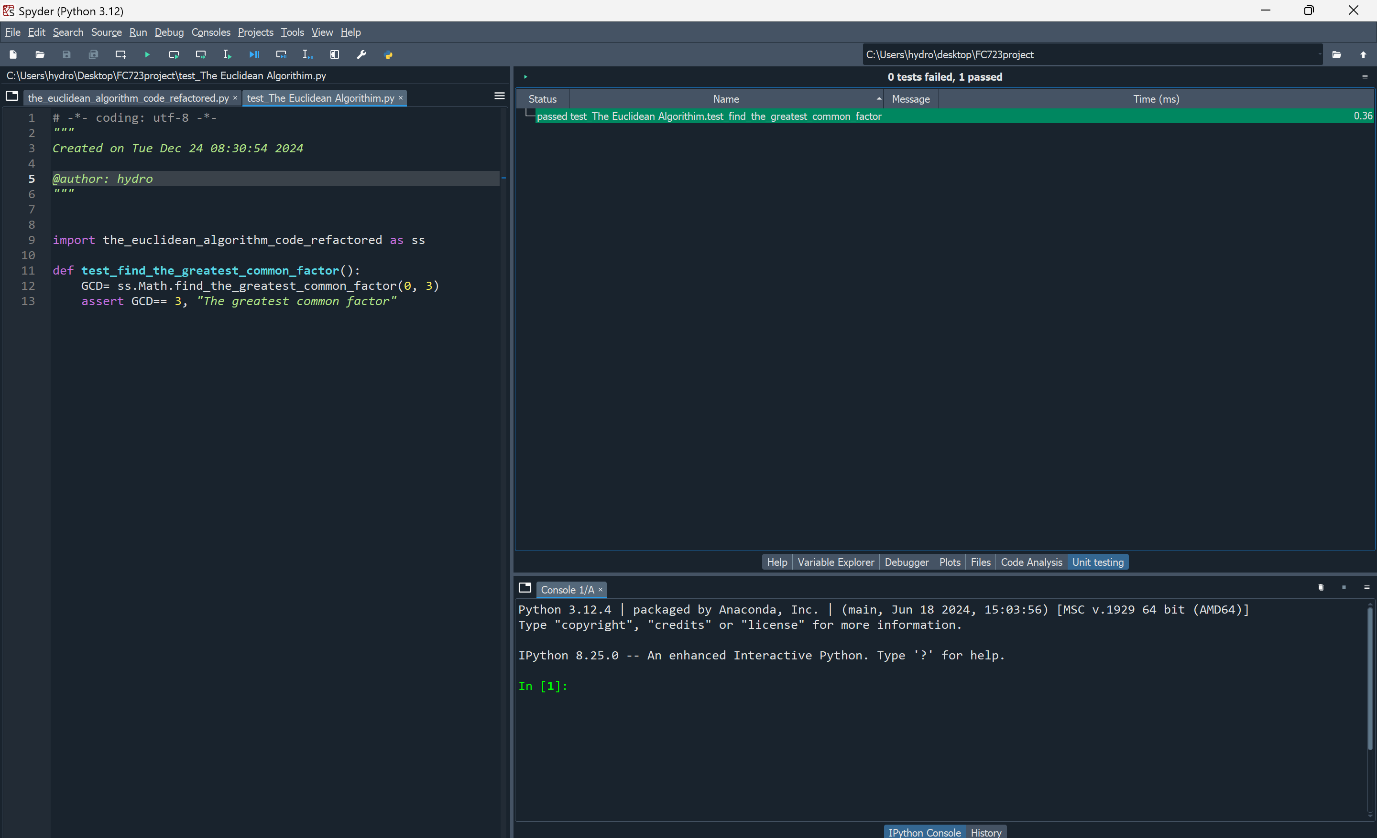
3075614, FC723, Sophie Norman

Running the test :

A screenshot of a computer test

Description automatically generated

Result:



3075614, FC723, Sophie Norman

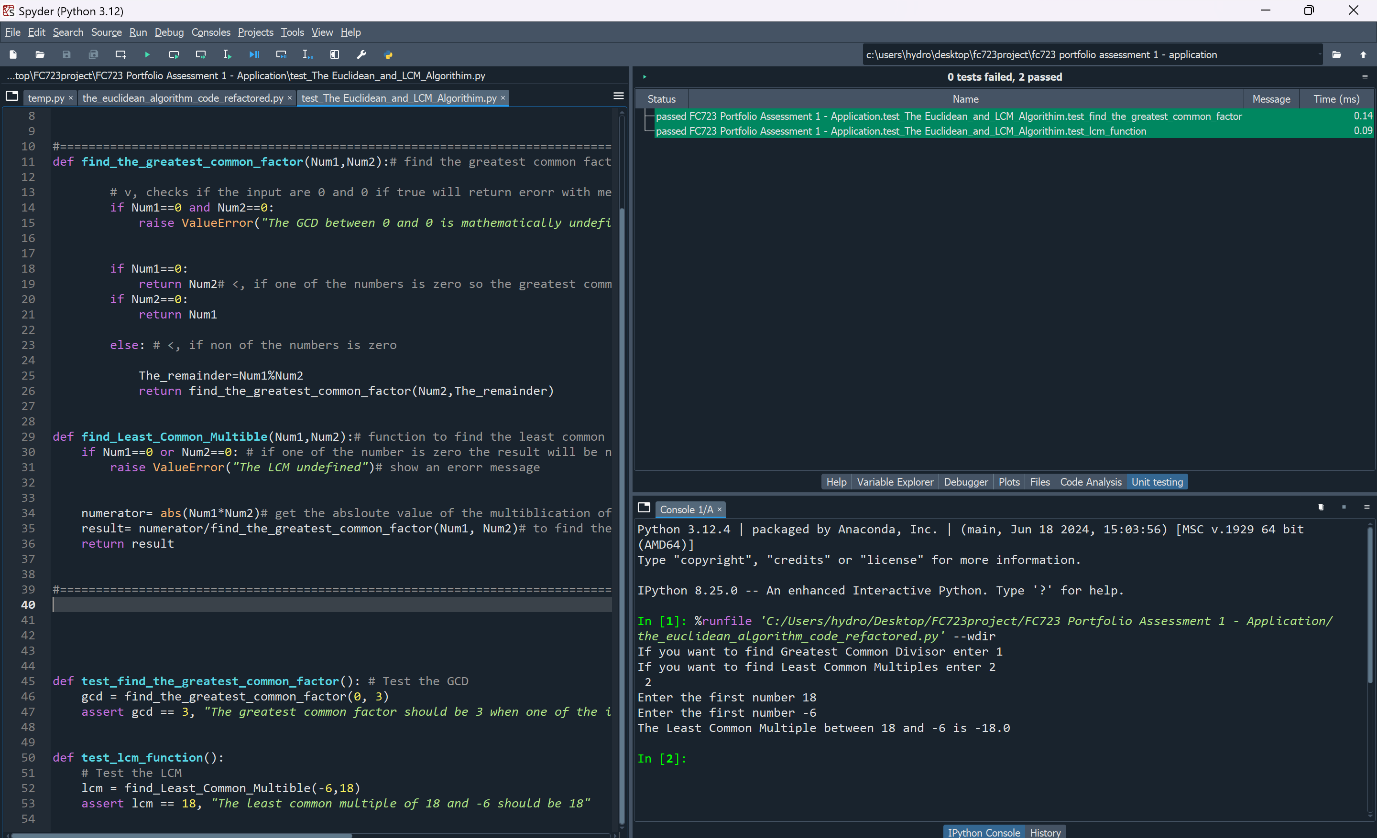
**Unit test for the LCM function:**

Running the test:

A screenshot of a computer test

Description automatically generated

The result:



both functions passed the tests.

3075614, FC723, Sophie Norman

* **Using Git:**

I keep track of my commits in git by writing comments using “git commit -m “The comment”. This way I can distinguish the difference between each commit by writing what have I done in the comment.

I often use commands like git status to check the tracked files and git add file name to add the file to tracked file to commit them later or git add . to add all the files at once. Then after the commit I use git push -f origin main to force push the files to my git repository. I used force push to make sure that I can push the files without any issues.