“I confirm that this assignment is my own work.

Where I/we have referred to academic sources, I have provided in-text citations and included the sources in the final reference list. “

* + - Module Code : FC723
    - Class/Group: Group
    - Module Title: Programming Theory
    - Assessment Title: Portfolio Project
    - Tutor Name:
    - Student GUID Number: 3075267
    - Date of Submission:

**Academic Resources:**

Bro Code. (2020, December 21). *Python exception handling ⚠️* [Video]. YouTube. https://www.youtube.com/watch?v=j\_q6NGOwDJo

I used this source to learn more of how I can implement try except keywords.

*W3Schools.com*. (n.d.). <https://www.w3schools.com/python/ref_func_map.asp>

I used this source to explore the map() function.

*Git - What is Git?* (n.d.). https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

I used this source to further explain my git commands.

**Link to my git repository:**

<https://github.com/lamawahab/Assesment_1_FC723>

**Explanation of git commands:**

**A diagram of a computer system

Description automatically generated**

I have tried to illustrate all of Git’s commands in a flowchart to show them clearly. As for keeping track of my commits, I will make use of the displayed history to check every new commit and check updates in previous ones. Additionally, I wrote extremely clear descriptions of each commit to help me identify any updates. GitHub also displays the timestamp, author, and file name which could help more when looking at the repository.  There is also a symbol panel that shows all functions and symbols in the Python file as well as tags that could be added to my commits for better navigation. Lastly, the most useful git command for me is the file tree that compares your python file with every other commit highlighting every added line with green and deleted lines with red.

**Pseudocode Design:**

Begin

#function to calculate the greatest common dividor

def GCD(a,b):

#if one of the numbers is zero then the GCD is the other number

if a==0: #if a is zero the result is b

result=b

elif b==0: #if b is zero the result is a

result=a

else: #if none of them is zero then it goes into the algorithm

#the algorithm continues until one of the two numbers is zero

while a!=0 and b!=0:

#locate which is the higher number and which is lower

if a>b:#if a is bigger then b is lower

big=a

small=b

else:#if b is higher then a is lower

big=b

small=a

reminder=big%small #find the reminder by using modulo

#assign new values to a and b to repeat process until one of the values is zero

a=small

b=reminder

#CHECK AGAIN: if one of the numbers is zero then the GCD is the other number

if a==0:#if a is zero the result is b

result=b

elif b==0:#if b is zero the result is a

result=a

return result#return the GCD

#a function that checks whether the number is a float or not

def Is\_float(num):

return num % 1 != 0

#a function that takes user input and check whether it is valid or not for GCD§

def User\_input():

print("Please enter two positive integers to continue: ")

#use float() function to change it from string and be able to operate on it

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

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

#check the validity of the inputs, if they are wrong asks user to input again until correct

# the euclidean algorithm does not work if both numbers are zero or if one or both numbers is a float or if one or both numbers is negative

while (num1==0 and num2==0) or (num1<0 or num2<0) or (Is\_float(num1)) or (Is\_float(num2)):

print("Your input is invalid please try again, enter two positive integers to continue")

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

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

result = GCD(num1,num2)#after ensuring validity, call the GCD function for the two numbers

print(f"the greatest common dividor of {int(num1)} and {int(num2)} is {int(result)} ")#print result

#function that takes more than one input for GCD

def multiple\_input():

print("Enter pairs of positive integers separated by spaces (e.g., '12 15 9 27').")

try: #if entered values are valid

input\_string = input("Enter the numbers: ").strip()#collect the user input and takeout extra white spaces

numbers = list(map(float, input\_string.split()))#spearate numbers and put them in a list as float

while len(numbers) % 2 != 0:#check if we can form even pairs with input

print("You must enter an even number of values to form pairs. Please try again.")#print error message if numbers are not even

input\_string = input("Enter the numbers: ").strip()#collect the user input and takeout extra white spaces

numbers = list(map(float, input\_string.split()))#spearate numbers and put them in a list as float

results = []#new list for results

for i in range(0, len(numbers), 2):#for loop that makes i the first number of pairs since we jump 2

num1, num2 = numbers[i], numbers[i+1]#i as in the first number in the pair, i+1 makes it the second

#check the validity of the inputs, if they are wrong asks user to input again until correct

# the euclidean algorithm does not work if both numbers are zero or if one or both numbers is a float or if one or both numbers is negative

while (num1==0 and num2==0) or (num1<0 or num2<0) or (Is\_float(num1)) or (Is\_float(num2)):

print(f"Your input {num1} and {num2} is invalid please try again, enter two positive integers to continue")

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

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

result = GCD(num1, num2)#after ensuring validity, call the GCD function for the two numbers

results.append(f"the greatest common dividor of {int(num1)} and {int(num2)} is {int(result)} ")#add the result as a string in the list

#print all results

print("\nResults:")#print title

for result in results:#loop goes through each string in the result list

print(result)

except ValueError:#invalid input exception

print("Invalid input. Please enter only numbers.")#print error message

User\_input()#call function to start program

multiple\_input()#call function to start program

End