Project 13-1: Greatest Common Divisor

Create a program that finds the greatest common divisor of two numbers.

Console

```
Greatest Common Divisor

Number 1: 15
Number 2: 5
Greatest common divisor: 5

Continue? (y/n): y

Number 1: 15
Number 2: 6
Greatest common divisor: 3

Continue? (y/n): y

Number 1: 15
Number 2: 7
Greatest common divisor: 1

Continue? (y/n): n

Bye!
```

Specifications

- Use the following recursive algorithm to calculate the greatest common divisor (GCD):

  ```
  divide x by y and get the remainder
  if the remainder equals 0, GCD is y (end function)
  otherwise, calculate GCD again by dividing y by remainder
  ```

- If number 1 is less than number 2, the program should display a message that indicates that number 1 must be greater than number 2 and give the user another chance to enter the numbers.
- Assume the user will enter valid data.

Student Guide: Python Greatest Common Divisor Calculator

Introduction

In this guide, we will create a Python program that calculates the Greatest Common Divisor (GCD) of two numbers. We'll build this program step by step, testing each part to ensure understanding and correct functionality.

Step 1: Writing and Testing `main`

Task

Create the main function that will drive your program.

Explanation

This step involves:

1. Function Definition: def main(): defines the main function, which is the entry point of a Python program.
2. Printing: print("Greatest Common Divisor Calculator") displays a message to the user. This demonstrate s basic output in Python.

Instructions

- **Code**: Start by defining your `main` function.

  ```python
  def main():
      print("Greatest Common Divisor Calculator")
      # Further steps will add more code here.

  if __name__ == "__main__":
      main()
  ```

- **Test**: Run the program to ensure the `main` function is set up correctly.
- **Verify**: Confirm that the program prints the initial message.

Step 2: Implementing User Input

Task

Write code to accept two numbers from the user.

Explanation

This step demonstrates:

1. User Input: input() function is used to get input from the user.
2. Type Conversion: int(input()) converts the string input into an integer, as input() returns a string by defa ult.
3. Variable Assignment: num1 and num2 store the user inputs.

Instructions

- **Code**: Add the following code to your `main` function for user input.

  ```python
  num1 = int(input("Enter the first number: "))
  num2 = int(input("Enter the second number: "))
  ```

- **Test**: Run your program and enter two numbers to test the input functionality.
- **Verify**: Ensure that the program accepts two numbers without errors.

Step 3: Writing and Testing the `gcd` Function

Task

Create a function named `gcd` that calculates the Greatest Common Divisor of two numbers.

Explanation

This step involves:

1. Recursive Function: def gcd(a, b): is a function that calls itself to find the GCD.
2. Base Case in Recursion: if b == 0: is the base case that stops the recursion.
3. Recursive Case: return gcd(b, a % b) is the recursive call with updated parameters.

Instructions

- **Code**: Implement the `gcd` function based on the provided pseudocode.

```python
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)
```

- **Test**: Add a temporary test in `main` or use a separate script to test the `gcd` function.
- **Verify**: Confirm that the function returns the correct GCD for various number pairs.

Step 4: Integrating and Testing `gcd` in `main`

Task

Integrate the `gcd` function into the `main` function.

Explanation

This step shows:

1. Function Integration: Using gcd(num1, num2) to call the gcd function within main.
2. Formatted Output: print(f"The GCD of {num1} and {num2} is {result}") uses an f-string for output.

Instructions

- **Code**: Use the `gcd` function in `main` to calculate and display the GCD.

  ```python
  result = gcd(num1, num2)
  print(f"The GCD of {num1} and {num2} is {result}")
  ```

- **Test**: Run the entire program with different pairs of numbers.
- **Verify**: Check that the correct GCD is displayed for each pair.

Step 5: Adding Error Handling and Input Validation

Task

Improve your program by adding error handling and validating input.

Explanation

This step will involve:

1. Error Handling: Implementing try-except blocks to handle non-integer inputs.
2. Input Validation: Checking if num1 is greater than num2 and handling the case if it's not.

Instructions

- **Code**: Implement checks to ensure user inputs are numbers and the first number is greater than the second.
- **Test**: Test these features with various inputs.
- **Verify**: Make sure the program handles invalid inputs gracefully.

Conclusion

Congratulations! You've successfully created a Python program to calculate the Greatest Common Divisor using a step-by-step, test-as-you-go approach. This method aids in understanding each code segment and ensures your program functions correctly.

Source Code

```python
#!/usr/bin/env python3

def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)

def main():
    print("Greatest Common Divisor Calculator")
    print()

    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))

    result = gcd(num1, num2)
    print(f"The GCD of {num1} and {num2} is {result}")
    print()

    print("Bye!")

if __name__ == "__main__":
    main()
```