# Programming Assignment Unit 4

## Department of Computer Science, University of the People

## **CS 1103 - Programming 2**

## **Student ID**: S409682

## Date: 28-10-2025

**Part 1:**

To develop the function according to client’s requirements, We will use incremental development.

**Initial code for validate input:**

def hypotenuse(a, b):

    if not isinstance(a, (int, float)) or not isinstance(b, (int, float)):

        return "Invalid input, Please provide numeric values for the sides of the triangle."

    if a <= 0 or b <= 0:

        return "Invalid input, Side lengths cannot be negative or 0."

    # Further code will go here for actual hypotenuse calculation

    return "input Ok"  # Placeholder for now

# Test Input Validation

print("hypotenuse for 3,4:", hypotenuse(3, 4))  # Test valid input

print("hypotenuse for -1,4:", hypotenuse(-1, 4))  # Test negative side length

**Output:**

hypotenuse for 3,4: input Ok

hypotenuse for -1,4: Invalid input, Side lengths cannot be negative or 0.

**Explanation**: This is initial code for the function. It will check if our inputs are valid in this part. It will first check the data type of input with **isinstance** function (Downey, 2015), If data type is other than integer or float it will return invalid input error message. Then it will check if input is 0 or negative number it will return invalid input error message. After all these check if input is valid then it will return “input ok” which will be replaced by math function later.

As we can see output for 3,4 was “input Ok” cause inputs are valid. But on second input -1 is negative so it returned validation error message.

**Code with Pythagorean Theorem:**

from math import pow, sqrt

def hypotenuse(a, b):

    if not isinstance(a, (int, float)) or not isinstance(b, (int, float)):

        return "Invalid input, Please provide numeric values for the sides of the triangle."

    if a <= 0 or b <= 0:

        return "Invalid input, Side lengths cannot be negative or 0."

    # Further code will go here for actual hypotenuse calculation

    # Calculation using Pythagorean theorem

    c = sqrt(pow(a,2) + pow(b,2))

    return c

#test function

print("hypotenuse for 5,6:", hypotenuse(5, 6))  # Test valid input

**Output:**

hypotenuse for 5,6: 7.810249675906654

**Explanation:** In this version of code everything is same as before just replaced the placeholder with actual math. It will import pow and sqrt function from math module and calculate hypotenuse with Pythagorean theorem.

Output of 5,6 is 7.810249675906654 as it first get the squire of both input number, then squire root the sum of them.

Final Code Explain: It maybe seems like all ok but there might be another issue. What happen if we input large numbers. It will give overflow error so We need to do error handling that’s why I added try-except on math formula line. So if there is any overflow error it will go to except block and print the error instead of breaking the program.

Final code:

from math import pow, sqrt

def hypotenuse(a, b):

    if not isinstance(a, (int, float)) or not isinstance(b, (int, float)):

        return "Invalid input, Please provide numeric values for the sides of the triangle."

    if a <= 0 or b <= 0:

        return "Invalid input, Side lengths cannot be negative or 0."

    # Further code will go here for actual hypotenuse calculation

    # Calculation using Pythagorean theorem

    try:

        c = sqrt(pow(a,2) + pow(b,2))

        return c

    except OverflowError as error:

        return error

#test function

print("hypotenuse for 99999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999,999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999:", hypotenuse(99999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999, 999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999))  # Test large input

print("hypotenuse for 3,4:", hypotenuse(3,4))

Output:

hypotenuse for 99999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999,999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999: math range error

hypotenuse for 3,4: 5.0

first output shows math range error as both inputs were so large. But when we input 3,4 output was 5.0 and there was not error.

**Part 2:**

I am developing a function to compute Fibonacci series for showcase in my portfolio.

Initial Code:

def fibonacci(n):

    # Placeholder function to compute Fibonacci series up to nth term

    if n <= 0:

        return "Invalid input. Please provide a positive integer."

    # Further code will go here

    return None  # Placeholder for now

# Test with a basic input

print(fibonacci(5))  # Test computing Fibonacci series up to 5th term

Output: None

Explanation: I have initially write the function that check for only positive inputs for our function. If input Is positive it will return None as a placeholder for further code. As we can see it output None for input 5

Updated code with initialize Fibonacci series:

def fibonacci(n):

    # Placeholder function to compute Fibonacci series up to nth term

    if n <= 0:

        return "Invalid input. Please provide a positive integer."

    # Further code will go here

    elif n == 1:

        return [0]

    elif n == 2:

        return [0, 1]

    fib\_series = [0, 1] # Initialize Fibonacci series with first two terms

    #fibonacci series computing formula will be added here

    return fib\_series

# Test with a basic input

print(fibonacci(5))  # Test computing Fibonacci series up to 5th term

Output: 0,1

Explanation: updated code will check the input if it is 1 then return [0] and for 2 return [0,1]. If none of the condition matched then it will initialize Fibonacci series with [0,1]

As we didn’t added the Fibonacci series compute formula output for 5 will be remain [0,1]

Final Code:

def fibonacci(n):

    # Placeholder function to compute Fibonacci series up to nth term

    if n <= 0:

        return "Invalid input. Please provide a positive integer."

    # Further code will go here

    elif n == 1:

        return [0]

    elif n == 2:

        return [0, 1]

    fib\_series = [0, 1] # Initialize Fibonacci series with first two terms

    while len(fib\_series) < n:

        fib\_series.append(fib\_series[-1] + fib\_series[-2])

    return fib\_series

# Test with a basic input

print(fibonacci(5))  # Test computing Fibonacci series up to 5th term

Output: [0, 1, 1, 2, 3]

Explanation:

Now code is completed and output will be Fibonacci series upto input length.