# **Programming Assignment Unit 4**

Computer Science, University of the People

CS 1101-01 Programming Fundamentals - AY2024-T1

Instructor, Jose Alvarado

September 28, 2023

## Part 1 - Incremental Development of the Hypotenuse Function

For this exercise, we were asked to write a function that takes 2 parameters and uses them to calculate and return the result based on the Pythagorean Theorem - a2 + b2 = c2. We are going to build the function in stages and explain each stage of the development and why we did what we did.

**Stage 1: the function stub**, in this stage we will build the basic outline of the function including its name, the parameters it will take and a return value. This step will help us make sure that the structure of the function is correct.



**Stage 2: the function body,** for this stage we are going to build the implementation of the Pythagorean Theorem in Python code. We will do this using the mathematical operator ‘\*\*’ which represents the power operator. The equation we are using in the function is a modified version of the Pythagorean Theorem since we want to return the length of the hypotenuse, so we need to use the following version: . To implement the square root calculation in Python we will user the power of 0.5 which result in the same value.



**Stage 3: Input validation**, for this stage we are going to add checks and validations to the parameters passed to us by the user of the function, this will help prevent misuse, and help users who do miss use the function to understand the issue and fix it faster. The Theorem declares that a and b must be positive values. So, we can add validations that make sure the user passes values that are correct.



**Final stage: Test the code**, for this stage we will test our function to verify it works using a few test cases making sure the expected result is returned in each case.



## Part 2 - Incremental Development of the ‘fizz\_buzz’ Function.

For this part of the assignment, I have chosen to build a function that takes in an integer value and depending on the value of the integer passed will return a string value or the original number passed in. The response of the function is as follows:

* "Fizz" if the number is divisible by 3 with no remainder.
* "Buzz" if the number is divisible by 5 with no remainder.
* "Fizz Buzz" if the number is divisible by both 3 and 5 with no remainder.
* The number itself if it's not divisible by 3 or 5.

**Stage 1 – the function stub**, as before, in this stage we will build the basic outline of the function including its name, the parameters it will take and a return value. This step will help us make sure that the structure of the function is correct.



**Stage 2 - Implementing "Fizz"**, for this stage we are going to add the logic that checks if the value passed in is divisible by 3 with no remainder and if so, will return the string “Fizz”.



**Stage 3 - Implementing "Buzz"**, next, for this stage, we are going to extend the logic that checks if the value passed in is divisible by 3 and if it doesn’t, we will check if instead, it is divisible by 5 with no remainder and if so, it will return the string “Buzz”.



**Stage 4 - Implementing "Fizz Buzz"**, this stage continues the extension of the conditional chain and adds a condition that checks if the parameter value is divisible by 3 and by 5 with no remainder. It's important to note that this condition needs to be added to the head of the condition chain, if not then the condition will never be true since the partial conditions will take precedence.



**Stage 5 - Input Validation**, for this stage we are going to age some input validations that will make sure the user passes a valid value into the function and if not, he is presented with a logical and helpful error message.



**Final stage: Test the code**, for this stage we will test our function to verify it works using a few test cases making sure the expected result is returned in each case.



**Conclusion**

In this assignment, we learned and applied the method of incremental development to construct two distinct functions: one to calculate the hypotenuse of a right triangle using the Pythagoras theorem and the other, a take on the classic “Fizz Buzz” problem. Each function was built step-by-step, starting with a basic stub, and progressively adding complexity and functionality. This approach allowed us to build each function in a way that would result in the least number of issues.

## References

Python 3 Documentation

<https://docs.python.org/3/>

Pythagorean theorem

Ancient Babylon and Egypt (beginning about 1900 B.C.)  
Publicized by Pythagoras 6th century B.C.

Think Python - How to Think Like a Computer Scientist (2nd Edition, Version 2.4.0)

Chapters 1-6

Source Scripts:

