Learning Journal Unit 4

Godfrey Ouma

University of the People

CS 1101: Programming Fundamentals

Janice Block

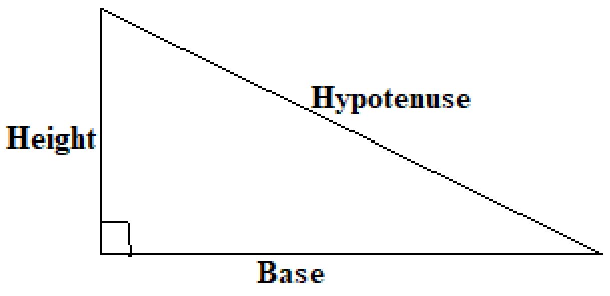
July 14, 2023

Part 1: Hypotenuse Function

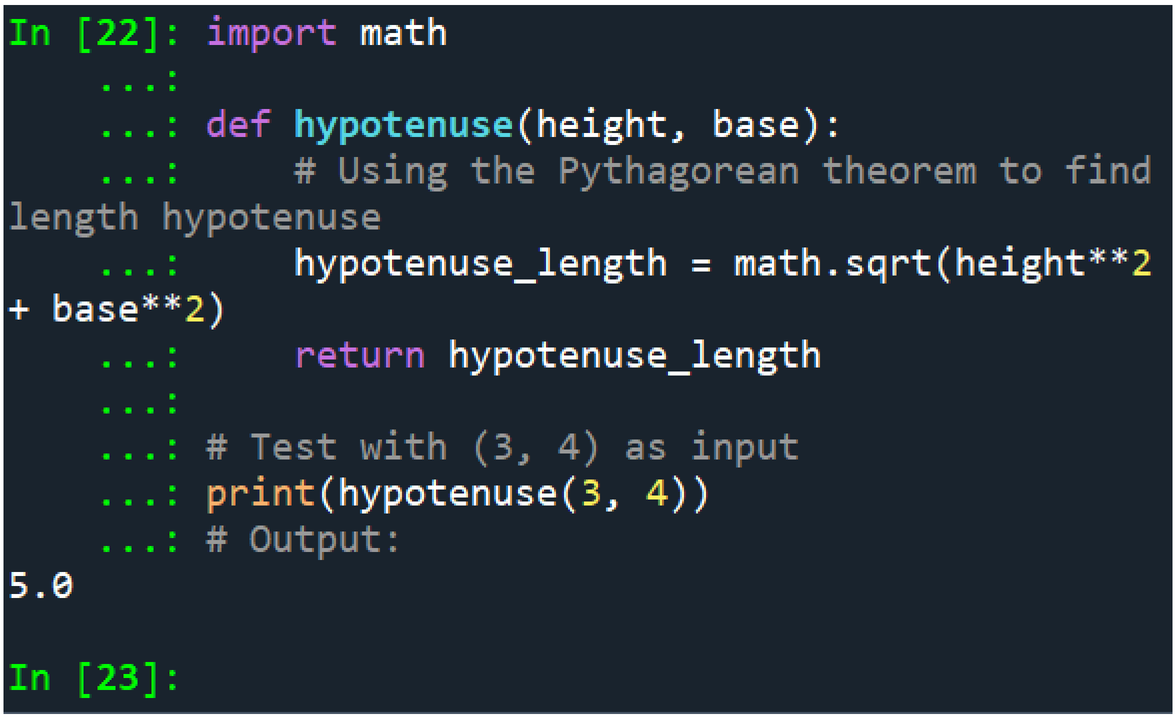
Stage 1: Initial Function Definition and Test

Figure 1

Parts of a right-angled triangle



The development process begins with defining the function "hypotenuse(height, base)", which takes two arguments "height" and "base" of a right-angled triangle, as shown in Figure 1. A simple case is also included to make sure the function is computable. See the codes below.

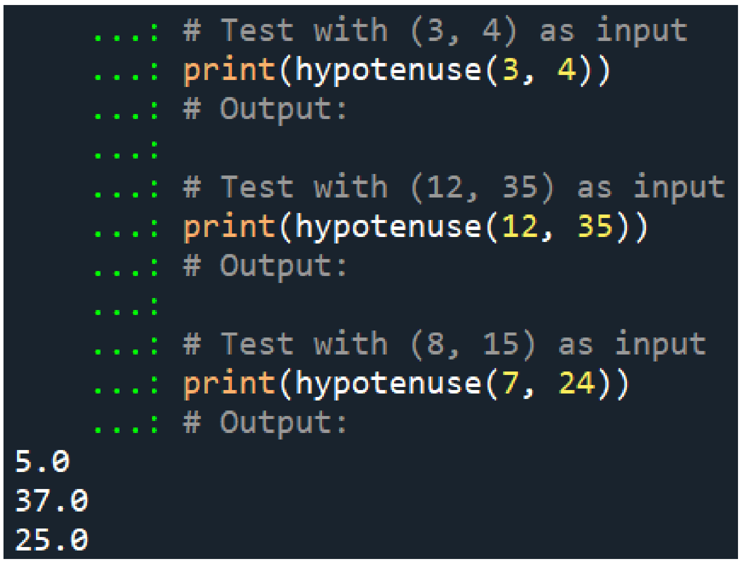


```
In [22]: import math
    ...:
    ...: def hypotenuse(height, base):
    ...:     # Using the Pythagorean theorem to find
length hypotenuse
    ...:     hypotenuse_length = math.sqrt(height**2
+ base**2)
    ...:     return hypotenuse_length
    ...:
    ...: # Test with (3, 4) as input
    ...: print(hypotenuse(3, 4))
    ...: # Output:
5.0

In [23]:
```

Stage 2: Performing Two More Test Cases

The function can undergo further validation by adding two more test cases with different arguments using other Pythagoras-associated triples, as shown below.



```
    ...: # Test with (3, 4) as input
    ...: print(hypotenuse(3, 4))
    ...: # Output:
    ...:
    ...: # Test with (12, 35) as input
    ...: print(hypotenuse(12, 35))
    ...: # Output:
    ...:
    ...: # Test with (8, 15) as input
    ...: print(hypotenuse(7, 24))
    ...: # Output:
5.0
37.0
25.0
```

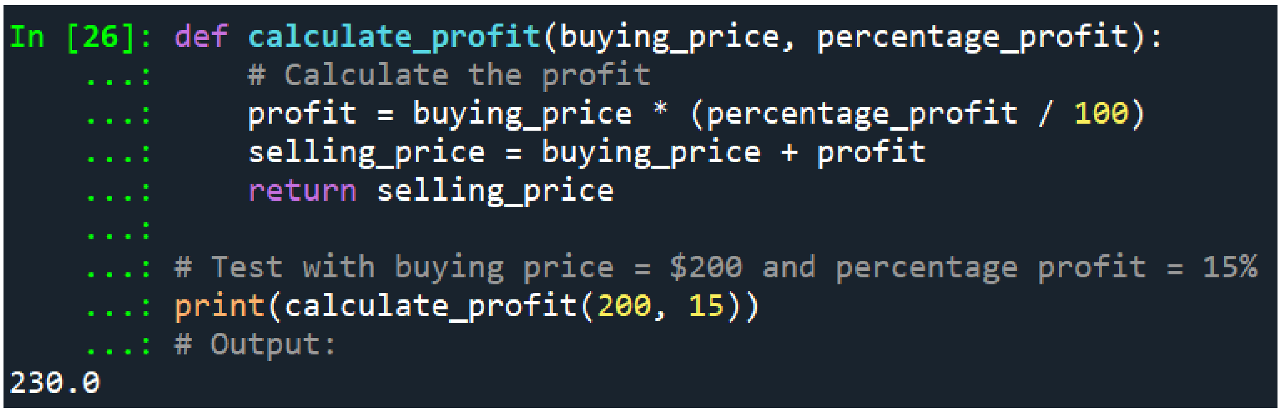
Explanation:

In the implementation above, the function "hypotenuse" takes two arguments: height and base, which represent two lengths of a right triangle shown in Figure 1. The function is set to apply the Pythagorean theorem ($hypotenuse = \left(\sqrt{height^2 + base^2}\right)^2$) to compute the length of the hypotenuse, which will be returned as the output. The first test case hypotenuse(3, 4) returns the length of length as 5.0 in the output, the second test case hypotenuse(12, 35) returns the length of length as 37.0 in the output, and the third test case hypotenuse(7, 24) returns the length of length as 25.0 in the output. All the test cases have successfully returned the length of the hypotenuse in the output to show that the function is working as anticipated. The three results prove that the function "hypotenuse(height, base)", can handle different input values and correctly compute the length of the hypotenuse for different right triangles. This supports Downey's (2015) claim that the purpose of incremental development is to add and test only a little amount of code at a time, hence avoiding lengthy debugging sessions.

Part 2: Custom Function

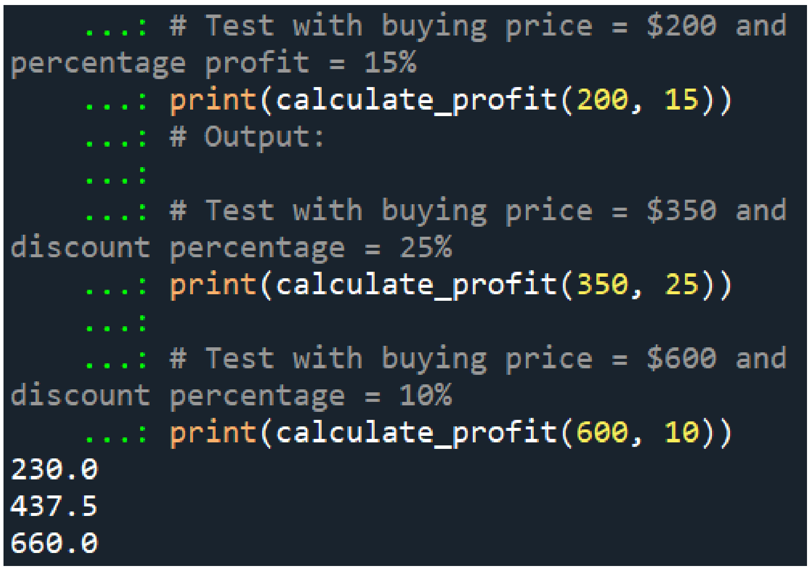
Part 2 entails creating a function called "calculate_profit" that calculates the selling price of an item given whose buying price and the percentage profit are indicated in the arguments. The development process of this code will be performed incrementally, and each stage recorded as shown below.

Stage 1: Initial Function Definition and Test



```
In [26]: def calculate_profit(buying_price, percentage_profit):
    ...:     # Calculate the profit
    ...:     profit = buying_price * (percentage_profit / 100)
    ...:     selling_price = buying_price + profit
    ...:     return selling_price
    ...:
    ...: # Test with buying price = $200 and percentage profit = 15%
    ...: print(calculate_profit(200, 15))
    ...: # Output:
230.0
```

Stage 2: Performing Two More Test Cases



```
    ...: # Test with buying price = $200 and
percentage profit = 15%
    ...: print(calculate_profit(200, 15))
    ...: # Output:
    ...:
    ...: # Test with buying price = $350 and
discount percentage = 25%
    ...: print(calculate_profit(350, 25))
    ...:
    ...: # Test with buying price = $600 and
discount percentage = 10%
    ...: print(calculate_profit(600, 10))
230.0
437.5
660.0
```

Explanation:

In the example above, the function "calculate_profit" takes two arguments: buying_price and percentage_profit. The function is then set to compute the selling price by finding the sum of profit and buying price. The profit is computed by taking the product of the buying price and the percentage profit divided by 100. The function then returns the selling price as the output. The first test case calculate_profit(200, 15) leads to the function correctly calculating the selling price as $230 when the original price is $200, while the percentage profit is 15%. The second and the third additional case tests, calculate_discount(350, 25) and calculate_discount(600, 10) return $437 and $660 respectively. Therefore, the three results confirm that the function “calculate_profit(buying_price, percentage_profit)” can handle different buying prices and percentage profits, correctly by computing the selling price accordingly. Similarly, just like in part 1, this supports Downey's (2015) claim that the purpose of incremental development is to add and test only a little amount of code at a time, hence avoiding lengthy debugging sessions.

Reference

Downey, A. (2015). *Think Python: How to think like a computer scientist*. Needham, Massachusetts: Green Tree