Functions

Kelly Rivers and Stephanie Rosenthal

15-110 Fall 2019

Announcements

- Add Deadline
- Homework 1 is due Monday at 12-noon!

  Start Early! Use office hours!

Recap – Unit 1

Monday – Algorithms

Wednesday – Programming Basics

Friday – Computer Organization

Today – Data Representation

Friday – Programming Functions

Learning Objectives

To distinguish code that could be written in functions

To read and trace code with functions including return statements and scope

To write new functions

Example

Consider the code:

```
print("Hello, my name is Stephanie")
print("Hello, my name is Kelly")
print("Hello, my name is Rebecca")
print("Hello, my name is Ellie")
print("Hello, my name is Gayatri")
print("Hello, my name is Rishab")
```

Tiring to write and error prone (we could easily forget parentheses)

Functions

Functions encapsulate meaningful code or repetitious code

Think: section headers in long papers

We can write the code one time and use it many times

Calling Functions

To use or call a function, write

```
name_of_function(arg1, arg2)
```

Notes:
Arguments can be variables or literals
Parentheses denote the set of arguments
Some functions do not require any arguments

Examples of Functions Already Used

```
str(2)       #function name: str, argument: 2
type(2)      #function name: type, argument: 2
int('5')     #function name: int, argument: '5'
print()      #function name: print, argument: (none)
pow(2, 5)    #function name: pow, arguments: 2,5
pow(3,2)     #function name: pow, arguments: 3,2
help()       #function name: help, arguments: (none)

#function name: print, arguments "here","to","there"
print("here","to","there")

x = 2        #no functions here
str(x)       #function name: str, argument: x
```

Return

The rest of the program doesn’t have access to any memory or variables that were used in the function

Return - If you want the rest of the program to remember or use a value that was computed in a function, you need to **return** it. A program that doesn’t return anything, returns None

Example: `str(5)`

I am calling str because I want a string.

The function `str` must return that string for me to use it

Return

The rest of the program doesn’t have access to any memory or variables that were used in the function

Return - If you want the rest of the program to remember or use a value that was computed in a function, you need to **return** it. A program that doesn’t return anything, returns None

Example: `print("HERE")`

I am calling print because I want something on the screen.
There is no other information for me. Print returns None.

Return Values as Variables

You can (and often do) assign a return value to a variable

```
s = str(2)
#str(2) returns "2", s holds the value "2"

t = str("2")
# str(2) returns "2", t holds the value "2"

i = int('5')
#int('5') returns 5, i holds the value 5
```

Return Values as Arguments

You can use a return value as the argument of another function

```
type(str(2))
#str(2) returns "2" type("2") returns str

type(int('5'))
#str('5') returns 5, type(5) returns int

print(int("4"))
#int("4") returns 4, print(4) returns None
```

Side Effects

Some functions produce side effects that change state outside of the function or program

Printing to the screen
Outputting data to a file
Displaying graphics

The computer modifies the look of my screen when I use print
The computer creates a file when I save data

Examples of Side Effects and Returns

```
print("15-110 is great")
```

Side effect: display "15-110 is great" on screen

Return: None

```
log2(2)
```

No side effects

Return: "1"

Common Errors

Wrong number of arguments:
`pow(2)` -> TypeError: pow expected at least 2 arguments, got 1

Error trying to print or use a return value that doesn't exist:
`print(print(5))` print(5) returns None

Importing Libraries

Other people write functions for you to use. Those functions are stored in code libraries. You must **import** a library to use it.

```
import math #a library of math functions
#Any function you want to call in math, you use math.funcname
math.sqrt(4) = 2.0
math.ceil(5.5) = 6
math.floor(5.5) = 5
math.factorial(5) = 120
math.log2(8) = 3.0
math.ceil(5.5,7.5) -> TypeError: ceil() takes one argument
math.sqrt(4,2) -> TypeError: sqrt() takes one argument
```

Writing Functions

```
def f(x):
    x = x+2
    return 2*x
    print("Done") #doesn't run
```

Start with def
Name the function
Parentheses list the arguments (can have any number of arguments)
Indent all lines of the function
Return at the end

Writing Functions

```
def f(x):
    x = x+2
    return 2*x
    print("Done") #doesn't run
```

def is the keyword that starts functions

Function name: f
Argument name: x

Colon : after the closing parentheses around args

All lines in the function are indented 1 tab

The last line the function runs is the return
Anything after the return does not get run

Function names: similar to variables, can’t start with a number, can’t have spaces, can’t use non-alphanumeric characters other than _

Example Functions

```
def function_name():
    print('This is a function.')
    print('Isn\'t that great?')
function_name()
function_name()
```

What is the side effect and the return of this function?

Example Functions

```
def sing_birthday(name):
    print('Happy birthday to you')
    print('Happy birthday to you')
    print('Happy birthday dear ' + str(name))
    print('Happy birthday to you')
sing_birthday('Jim')
```

Example Running Function with Different Arguments

```
def f(x):
    return 2 * x + 1

#you can call the function with different arguments
z = f(4)
y = f(5) + 1
```

What is the side effect and the return of this function?

Beginning of Class Example

Before

```
print("Hello, my name is Stephanie")
print("Hello, my name is Kelly")
print("Hello, my name is Rebecca")
print("Hello, my name is Ellie")
print("Hello, my name is Gayatri")
print("Hello, my name is Rishab")
```

After

```
def hello(name):
     print("Hello, my name is "+name)

hello("Stephanie")
hello("Kelly")
hello("Rebecca")
hello("Ellie")
hello("Gayatri")
hello("Rishab")
```

Scope

```
def hello(name):
    print("Hello, my name is "+name)
```

Variable `name` is only available to use within the function and every time the function is called it has a new value.

If I want to use that string outside of the function, I have to return it

Beginning of Class Example

```
def hello(name):
    print("Hello, my name is "+name)

hello("Stephanie")
hello("Kelly")
hello("Rebecca")
hello("Ellie")
hello("Gayatri")
hello("Rishab")
```

- What if I want to use the "Hello, my name is Stephanie" string? Return it

Beginning of Class Example with Return

```
def hello(name):
    s = "Hello, my name is "+name
    return s

hello("Stephanie")
print(s)
```

Beginning of Class Example with Return

```
def hello(name):
    s = "Hello, my name is "+name
    return s

hello("Stephanie")
print(s)
```

The variable `s` is only usable in the function `hello`

Beginning of Class Example with Print and Return

```
def hello(name):
    s = "Hello, my name is "+name
    print(s)
    return s

hello("Stephanie")
hello("Kelly")
hello("Rebecca")
hello("Ellie")
hello("Gayatri")
hello("Rishab")
```

Scope

```
def hello(name):
    s = "Hello, my name is "+name
    print(s)
    return s
```

Don’t forget to assign a variable to the returned value!

Beginning of Class Example with Print, Return, Assign

```
def hello(name):
    s = "Hello, my name is "+name
    print(s)
    return s

n1 = hello("Stephanie")
n2 = hello("Kelly")
n3 = hello("Rebecca")
n4 = hello("Ellie")
n5 = hello("Gayatri")
n6 = hello("Rishab")
```

Making Name Badges – two functions

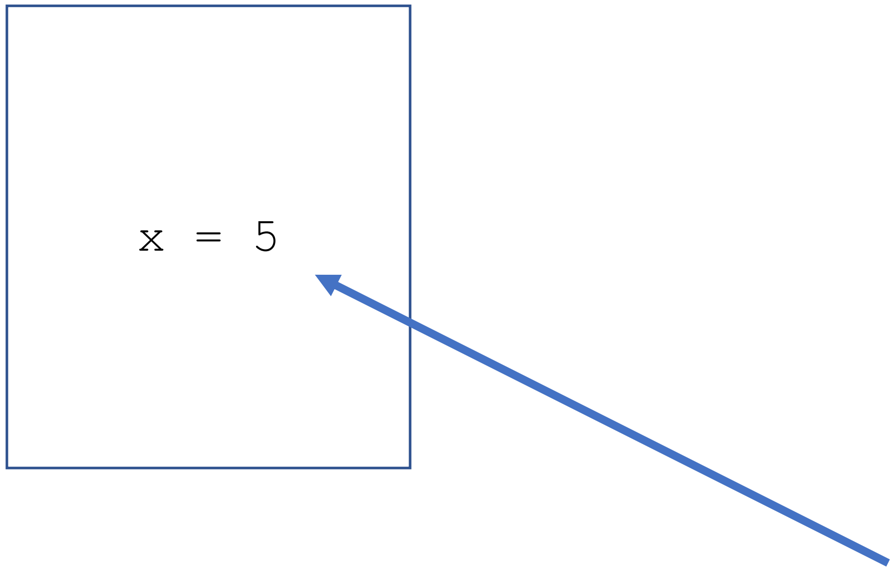
```
def hello(name):
    s = "Hello, my name is "+name
    print(s)
    return s

def make_name_badge(ns1, ns2, ns3, ns4, ns5, ns6):
    #organize and print 6 on a page
    return None

n1 = hello("Stephanie")
n2 = hello("Kelly")
n3 = hello("Rebecca")
n4 = hello("Ellie")
n5 = hello("Gayatri")
n6 = hello("Rishab")
make_name_badge(n1, n2, n3, n4, n5, n6)
```

Global Scope

You can use a variable in a function that was defined outside of the function (but not in a different function)

Valid Code:

```
x = 5
def f(c):
    print(x)
```



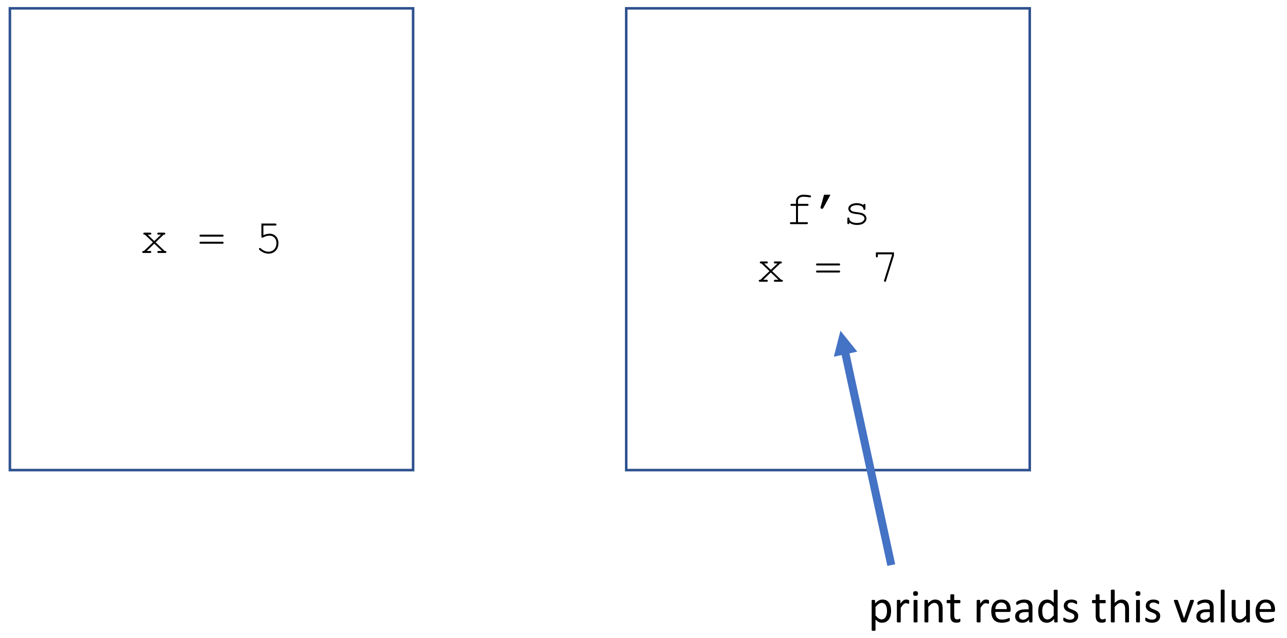
print reads this value

Global Scope

Attempting to set an *external* variable inside a function creates a function-specific variable instead

Invalid Code:

```
x = 5
def f(c):
    x = 7
    print(x)
```

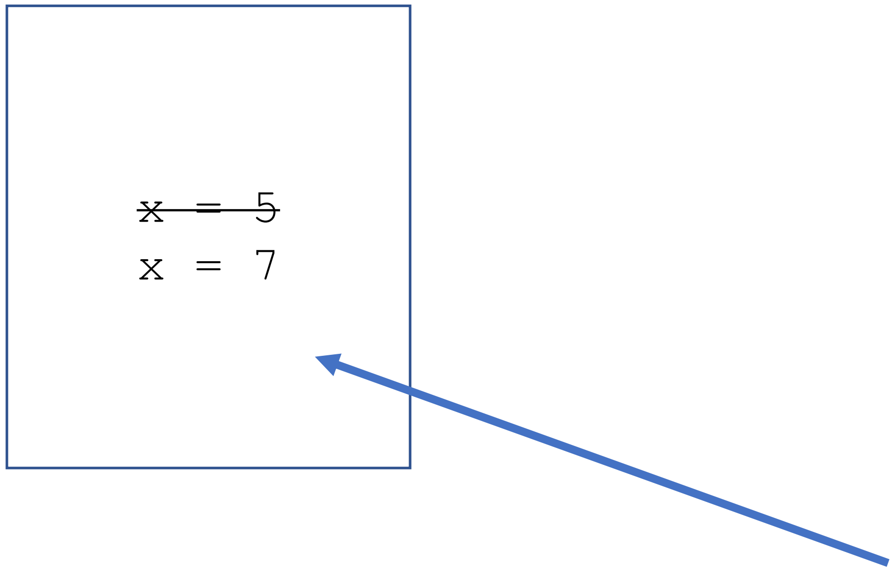


Global Scope

To set an *external* variable inside a function, use global (another side effect!)

Valid Code:

```
x = 5
def f(c):
    global x
    x = 7
    print(x)
```



print reads this value

More Scope Examples

```
def secretaddition(x,y):
    #no other z is defined
    return x+y+z

z=3 #this z can be used inside the function
secretaddition(1,2)         #returns 6
print(z)                    #prints 3
```

Scope with Variables Examples

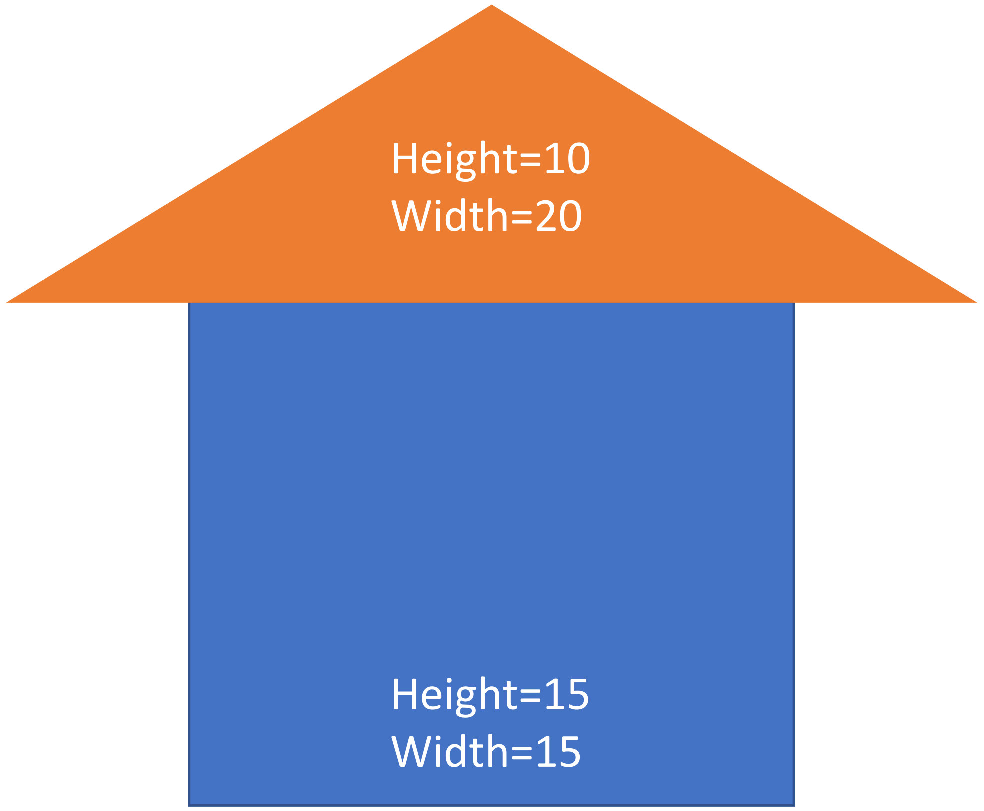
```
def secretaddition(x,y):
    z = 10 #this is a new z, it is only available
           #inside the function
    return x+y+z

z=3
secretaddition(1,2)  #returns 13
print(z)             #prints 3
```

Scope with Variables Examples

```
def secretaddition(x,y):
    global z #use the z defined outside the function
    z = 10    #the value of the outside z is changed
    return x+y+z

z=3 #this z will change value
secretaddition(1,2)  #return 13
print(z)             #return 10
```

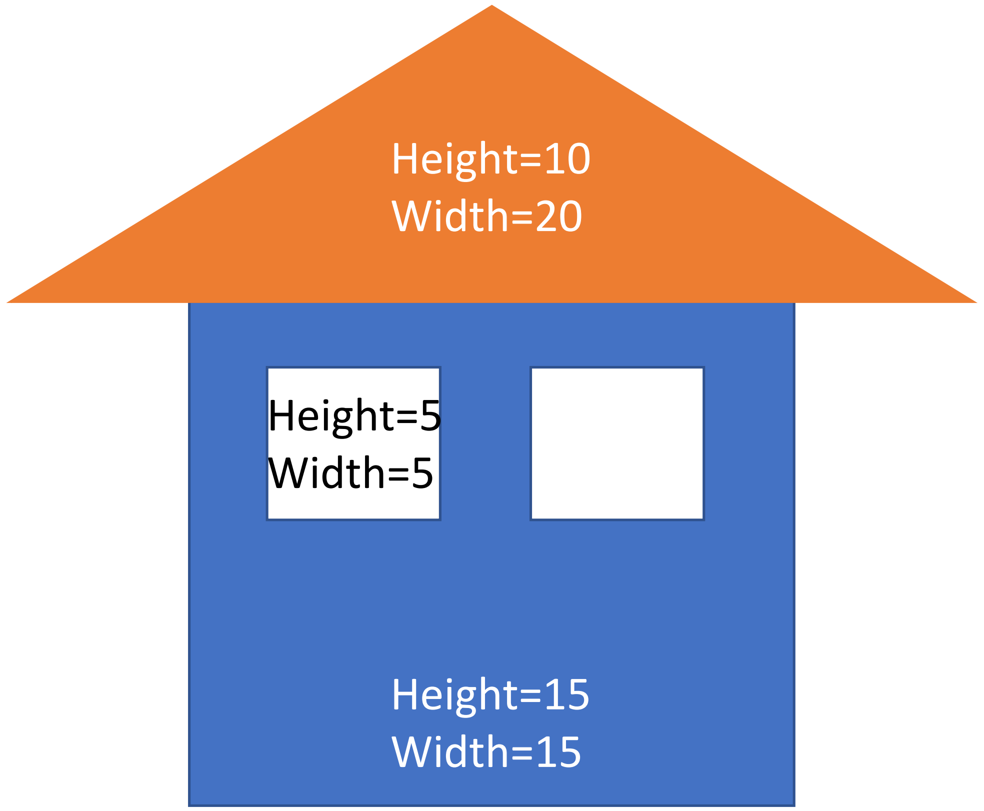
Geometry Example

Write a function to compute the area of a square and area of a triangle
Use those functions to compute the area of this house



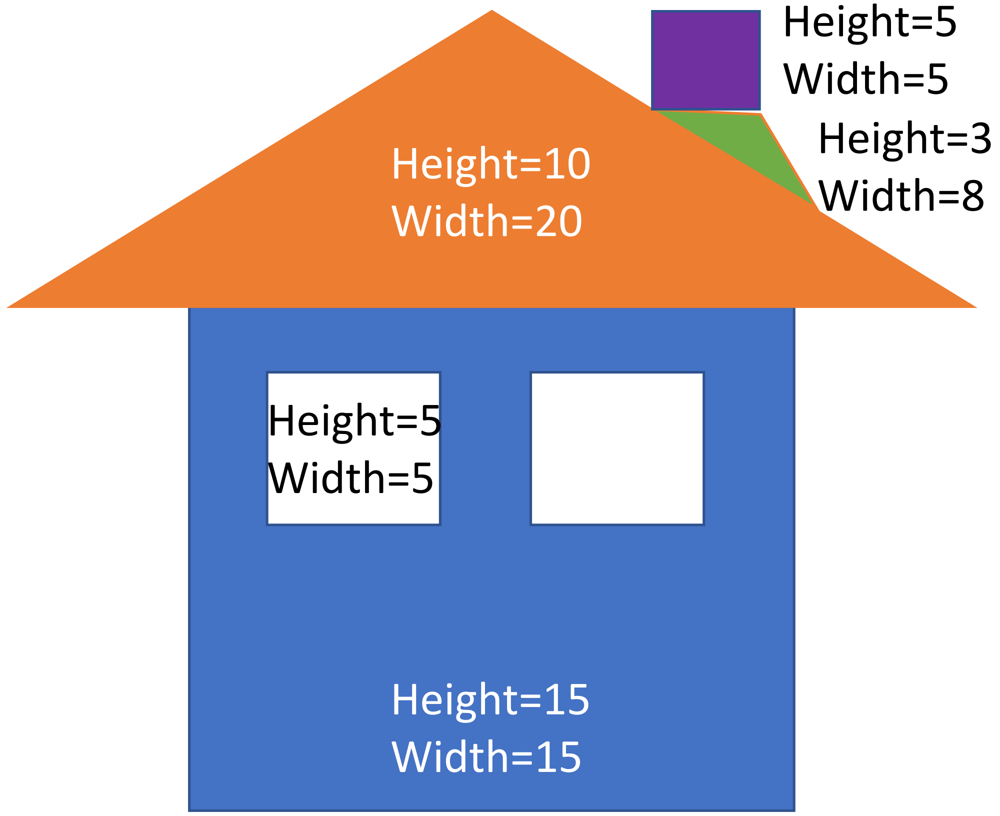
Geometry Example

Write a function to compute the area of a square and area of a triangle
Use those functions to compute the area of this house
Subtract the area of the windows



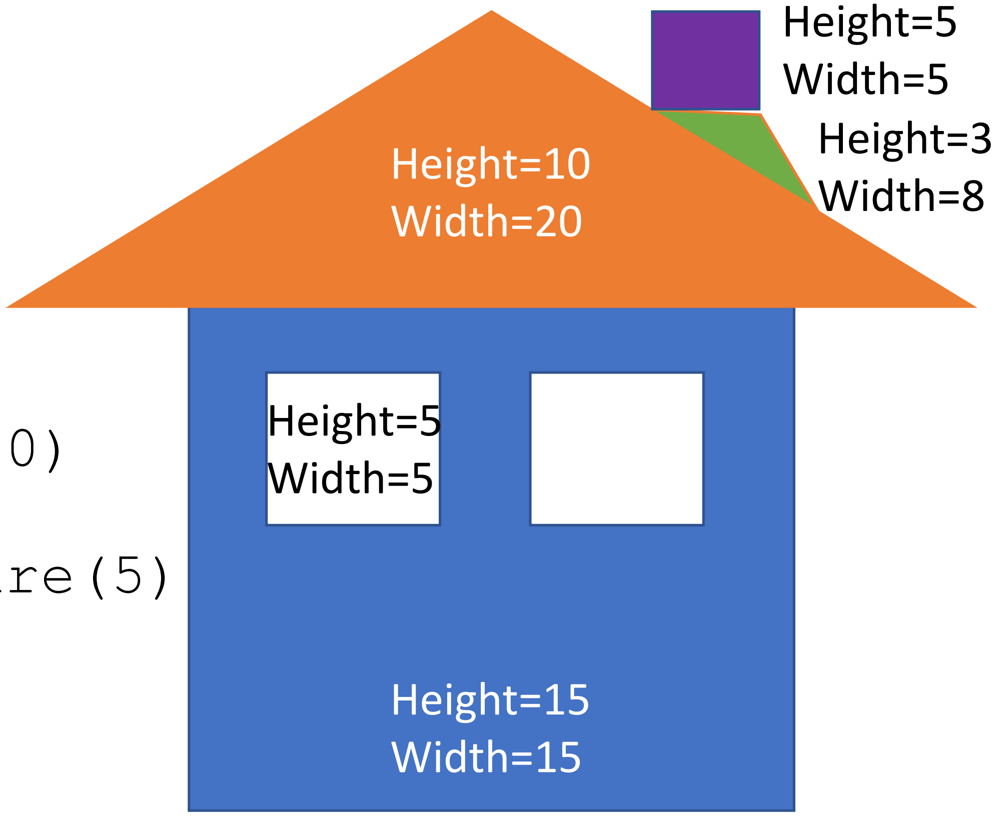
Geometry Example

Write a function to compute the area of a square and area of a triangle
Use those functions to compute the area of this house
Subtract the area of the windows
Add the area of the chimney



Geometry Example Answer

```
def areasquare(w):
    return w*w

def areatriangle(h,w):
    return (w*h)/2

part1 = areasquare(15)+areatriangle(10,20)
part2 = part1-(2*areasquare(5))
part3 = part2+areatriangle(3,8)+areasquare(5)
```


Height=5
Width=5
Height=3
Width=8
Height=10
Width=20
Height=5
Width=5
Height=15
Width=15