CSI Lab 9

Tuesday, March 11, 2014

Once again, I will allow you the OPTION of working with a partner today. I STRONGLY encourage you to use a partner, but I will not require this option. If you work with a partner, you should be working on one computer and following the rules of paired programming. Overall, I think it will be better for you to work with a partner, but it is up to you.

In Monday's class we introduced the idea behind writing your own functions. You have been working with functions all semester (print(), range(), len(), etc). However, it is often helpful to be able to write your own functions. Now that you have seen how (lecture on Monday) and read how (reading your textbook), you can practice how.

To get started, download the following file:

- lab09.py

First, open the file and ***type your names*** in the Name field. Save the file.

Next, take a look at what is in the file. You will see that there are two sections of code. Section one defines the stubs for three functions that you will write this :

```
##Section One
"""This is the only section where you will write code"""

def hydrocarbon(hydrogen,carbon,oxygen):
    """Calculates the atomic mass of the given hydrocarbon"""
    # Add your code here


def bmi(weight,height):
    """Calculates the bmi of the person with the given weight (lbs)
        and height (inches)."""
    #Add your code here



def collatzLength(seed):
    """Calculates the number of steps required to move from the seed
            value to the base value of 1"""
    #Add your code here
```

Section Two contains code that will test the functions that you will write during this lab. You ***should not*** modify the code in the section. You MAY feel free to read what it says and does. This screenshot below is just a small part of the file:

```
##Section Two
"""Do NOT change the code below this section
    This is the test suite.  It helps you test if your code is correct
        and complete.
    If you get messages about incorrect values then you should
        look at the message to find the expected value
        and see what you can do to fix the code ABOVE this."""

print("Start checking for errors.")

result=hydrocarbon(4,3,0)
expected=40.0646
if not result==expected:
    print("hydrocarbon(4,3,0) produces",result,"expected",expected)
```

Once you get done reading the code load and run this file. You should see:

```
Start checking for errors.
hydrocarbon(4,3,0) produces None expected 40.0646
hydrocarbon(1,2,3) produces None expected 73.0281
hydrocarbon(0,0,0) produces None expected 0

bmi(150,66) produces None expected 24.210365225138165

collatzLength(1) produces None expected 0
collatzLength(2) produces None expected 1
collatzLength(3) produces None expected 7
collatzLength(4) produces None expected 2
collatzLength(5) produces None expected 5
collatzLength(6) produces None expected 8
collatzLength(7) produces None expected 16
collatzLength(8) produces None expected 3
collatzLength(9) produces None expected 19
Done checking for errors.
```

In other words, right now this code is full of errors. Well that isn't very surprising because the three functions don't have any code in them right now which means they return nothing (or more accurately, they return None). Your job is to write the body of the three methods so that when you run you program you see no errors at all.

```
Start checking for errors.


Done checking for errors.
```

[Q1] Where will you be writing your code in this lab?

[SIG1] Please show this to an instructor BEFORE moving on

Customer Request #1 (hydrocarbon from the midterm)

In your file locate the stub of the function called hydrocarbon(). This function should:

- take in three parameters assumed to be hydrogen, carbon, and oxygen IN THAT ORDER.
- validate that all three values are ***non-negative*** integers:
    - if not than you should return None
- if all three parameters are valid then you should calculate the mass of the hydrocarbon and return the mass where:
    - Each hydrogen has a molecular weight of 1.0079
    - Each carbon has a molecular weight of 12.011
    - Each oxygen has a molecular weight of 15.9994

For example:

- hydrocarbon(4,3,0) will return 40.0646
- hydrocarbon(1,2,3) will return 73.0281
- hydrocarbon(-1,-1,-1) will return None

When you think you have this working, load and run lab09.py. If you see ANY messages that talk about hydrocarbon then you know you don't yet fit the requirements listed above. See what is wrong and fix your code. Keep trying until all hydrocarbon messages disappear.

[SIG2] Please show this to an instructor BEFORE moving on

Customer Request #2 (bmi from PA02)

In your file locate the stub of the function called bmi(). This function should:

- take in two parameters assumed to be weight in pounds and height in inches IN THAT ORDER
- validate that both values are ***positive*** integers:
    - if not than you should return None
- if both parameters are valid than you should calculate the bmi as instructed in PA#2 and return this value.

So that everyone gets the EXACT same results you should use 0.0254 as your height conversion factor and 0.453592 as your weight conversion factor.

For example:

- bmi(150,66) will return 24.2104...
- bmi(150,0) will return None

Again, keep modifying and rerunning your code until you eliminate any messages that bmi does not work.

[SIG3] Please show this to an instructor BEFORE moving on

Customer Request #3 (New assignment)

Background:

The Collatz conjecture is an unsolved conjecture in mathematics named after Lothar Collatz, who first proposed it in 1937. The conjecture is also known as the 3*n* + 1 conjecture (because of it's formula) or the hailstone sequence (because of the way that the numbers bounce up and down like hail until they finally converge to 1).

The sequence works as follows:

- Take any integer *n*.
- If *n* is even, divide it by 2 to get *n* / 2.
- If *n* is odd multiply it by 3 and add 1 to obtain 3*n* + 1.
- Repeat the process until the number reaches 1, then quit

The conjecture is that no matter what number you start with, you will always eventually reach 1. This has been demonstrated true for integers of approximately 10**16 in size but has never formally been proven to be true.

For example, consider starting with a value of 3

Old Value	This is:	So the new value is:
3	odd	3*3 + 1 = 10
10	even	10/2 = 5
5	odd	3*5 + 1 = 16
16	even	16/2 = 8
8	even	8/2 = 4
4	even	4/2 = 2
2	even	2/2 = 1
1	THE END	

This has a length of 7. We must perform 7 operations before we finally reach 1.

The assignment:

In your file locate the stub of the function called collatzLength(). This function should:

- take in one parameter assumed to be a positive integer
- validate that it is a positive integer:
    - if not than you should return None
- if it is a non-negative integer it should calculate the length of the sequence needed to reach a value of 1
- It should return this value.

For example:

- collatzLength(0) will return None
- collatzLength(3) returns 7
- collatzLength(8) returns 3

Again, keep modifying and rerunning your code until you eliminate any messages that collatzLength does not work.

[SIG4] Please show this to an instructor BEFORE moving on

Final Submission

You will know that you have written your functions correctly if you see the following when you run your code:

```
Start checking for errors.


Done checking for errors.
```

This week I will again ask you to submit your code for electronic grading, using the eLearning submission system.

Follow the directions on the system to select the appropriate course and assignment and submit

- lab09.py

Don't forget to hand in your answer sheet to the professor or TA before you leave!