CS 2316

Individual Homework 1 – Python Practice

Due: Wednesday, January 18th, before 11:55 PM
Out of 100 points

Files to submit: **1. HW1.py**

This is an INDIVIDUAL assignment!

Collaboration at a reasonable level will not result in substantially similar code. Students may only collaborate with fellow students currently taking CS 2316, the TA's and the lecturer. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others.

For Help:
- TA Helpdesk – Schedule posted on class website.
- Email TA's or use T-Square Forums

Notes:
- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
- **Do not wait until the last minute** to do this assignment in case you run into problems.

Part 1 – Simple Functions

You will write a few python functions for practice with the language. In your HW1.py file, include a comment at the top with your name, section, GTID/Email, and your collaboration statement. Also include each of the following functions.

1. GnToFbf
2. coneVolume
3. footballScore
4. bodyMassIndex

Function Name: **GnToFbf**

Parameters:
None
Return Value:
None
Description:

1. Write a user-interactive function to convert Giraffe's Necks (Gn) to Foot Ball Fields (Fbf). The conversion equation is as follows: 1 Fbf is 3,600 inches, while 1 Gn is 70 inches.

2. Get the number of Gn from the user; make sure to use a descriptive prompt so the user knows what to enter (e.g. “Enter number of Giraffe's Necks”). You may assume that the user will enter a valid float (e.g. 34.4) and do not have to do error checking.
3. Convert the number entered by the user to Foot Ball Fields using the conversion information above.
4. Print the calculated number of Foot Bal Fields (including fractional Fbf); be sure to add a label to the display value so the user knows what the value means (e.g. display “35 Fbf” instead of just 35)
5. Error Check: 500 Gn's should be 9.722222 Fbf's.

Function Name: **coneVolume**

Parameters:

None

Return Value:

None

Description:

Write a user-interactive function to calculate the volume of a cone.

1. Get the height, and radius of the circle at the base of the cone, and the unit of measurement of the cone from the user; make sure to use a descriptive prompt so the user knows what to enter.
2. Calculate the volume of the cone using the following formula: Volume = 1/3 * pi * radius^2 * height.

3. Print the calculated volume; be sure to add a label to display value so the user knows what the value means with the units entered by the user (e.g. display “Volume of the cone is 35.322 units-cubed” instead of just 35.322, where units would be replaced with whatever unit of measurement the user entered).

Function Name: **footballScore**

Parameters:

None

Return Value:

None

Description:

Write a user-interactive function to make a guess at how a team earned the points they did for a particular score in an American football game.

1. Get the team’s score as an integer from the user; make sure to use a descriptive prompt so the user knows what to enter.
2. Compute the number of touchdowns, field goals, two-point conversions, and extra points that could make up the score the user entered (We are assuming no safeties). You should compute each of these numbers in the above order using the following information:
   - There are 6 points in a touchdown
   - There are 3 points in a field goal

- There are 2 points in a 2-point conversion
- There is 1 point in an extra point.
- The modulo (a.k.a. remainder) operator in python is % and will show the remainder left after an integer division. It IS useful for this problem!

3. Print the calculated number of touchdowns, field goals, two-point conversions, and extra points on **one line;** be sure to add appropriate labels to the display values so the user knows what the value means (e.g. display "3 touchdowns, 1 field goals, 1 two-point conversions, 0 extra points"). Also note that for low scores (below 6), you may get a score that is not legal according to the rules of American football; this is OK.

Function Name: **bodyMassIndex**

Parameters:

None

Return Value:

None

Description:

Write a user-interactive program to calculate a person's BMI, body mass index.

1. Ask for the person's weight in pounds; make sure to use a descriptive prompt so the user knows what to enter.
2. Ask for the person's height in inches; again, use a descriptive prompt.
3. Calculate the person's BMI using the information they provided and the formula below (you will need to convert the formula into a legal python equation):

$$\text{BMI} = \frac{weight(\text{lb})}{height^2(\text{in}^2)} \times 703$$

4. Print the result. The value must be formatted (both the wording and the number of decimal places) EXACTLY as follows: "Your BMI is 24.9." (where the value for BMI represents whatever you calculated based on the inputs). *You may have to use string formatting to remove extra decimal places from your calculation.*

Part 2 – Complex Functions

Now you will write a few python functions that return data for practice with the language.

Function Name: **calcDistance**

Parameters:

1. x_1 – a number representing the x coordinate of the first point as an integer
2. y_1 – a number representing the y coordinate of the first point as an integer
3. x_2 – a number representing the x coordinate of the second point as an integer
4. y_2 – a number representing the y coordinate of the second point as an integer

Return Value:

A floating point number representing the distance between the two points.

Test Cases:

1. calcDistance(5,10,15,20) --> 14.142135623730951
2. calcDistance(-1,15,7, -3) --> 19.697715603592208

Description:

Write a function calcDistance that will calculate and return the distance between the two given points as a floating point number. Use the distance formula given below:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Note that to successfully complete this problem, you will need to take the square root. To do this, you must import the math module using "import math" before you use the square root function, which is named sqrt. You can call functions from modules you import by saying modulename.functionname(params) instead of just saying functionname(params).

Function Name: **makeBikes**

Parameters:

1. wheels - an integer representing the number of wheels
2. frames - an integer representing the number of frames
3. links - an integer representing the number of links of chain

Return Value:

An integer representing the amount of bicycles you can make given the inputted stock of material.

Test Cases:

1. makeBikes(10, 5, 150) --> 3
2. makeBikes(1, 5, 51) --> 0
3. makeBikes11, 3, 200) --> 3

Description:

Write a function that returns the number of bicycles you can make based on the number of wheels, frames, and chain links specified as inputs. Assume that it takes the following to make 1 bicycle:

- 2 wheels
- 1 frame
- 50 links of chain

Note that the number of bikes must be a whole number; you cannot have 1 1/2 of a bike. For example, if you have 2 wheels, 3 frames, and 5000 links of chain, you can only make 1 bike (despite the fact that you have enough frames and chain links to make 3 bikes).

Hint: Python has a built-in function called min that takes in a comma separate list of numbers and returns the minimum value. E.g. min(5, 3, 7) returns the number 3. This may be useful in coming up with your solution.

You can solve this problem without using conditionals!

Part 3 – Turtles

Function Name: **drawSkyline**
Parameters:
None
Return Value:
None

Description:
Write a function that uses the turtle module to create a window and draw a skyline in the window. Your skyline must have at least ten buildings, with at least three different styles of building. You may create helper functions that you call to assist you with drawing the various buildings. You may use a for or while loop to draw the buildings, but the buildings can not be all the same height. You may add extra features such as windows, atmospheric effects, etc.

Grading

You will earn points as follows for each function that works correctly according to the specifications.

Simple Function

GnToFbf	**5**
coneVolume	**10**
footballScore	**20**
bodyMassIndex	**15**

Complex Functions

calcDistance	**15**
makeBikes	**20**

Turtles

drawSkyline	**15**

If your function works, but prints a value when it is supposed to return it, or returns it when you are supposed to print it, the TA will **deduct half of the points** for that function.