CS 2316

Individual Homework 1 – Python Practice

Due: Wednesday, January 15th, before 11:55 PM
Out of 100 points

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

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.

Learning Objectives Associated With This Homework:

- Variable Assignment
- Simple Expressions
- Calling Functions
- Defining Functions
- Printing and String Formatting

Part 1 – Simple Functions

You will write a few python functions for practice with the language. In part 1, all functions will use the **INPUT** function to get input data from the user and the **PRINT** function to display output to the user. All of the functions from part 1 and part 2 will appear in the same file, HW1.py. In your HW1.py file, also include a comment at the top with your name, section, GTID/Email, and your collaboration statement. After the comment, write each of the following functions. You must name each function exactly as specified in this document, and your input and output must exactly match the given examples.

1. mmHgToPSI
2. changeMaker

Function Name: mmHg**ToPSI**

Parameters:

None

Return Value:

None

Description:

1. Write a user-interactive function to convert mmHg to PSI. The conversion equation is as follows: **1** psi is equal to **51.71492** millimeter of mercury (mmHg)

2. Get the number of mmHg from the user; make sure to use a descriptive prompt so the user knows what to enter (e.g. "Enter mmHg"). 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 PSI using the conversion information above.
4. Print the calculated PSI (including fractional components); be sure to add a label to the display value so the user knows what the value means (e.g. display "35 PSI" instead of just 35)
5. Error Check: **1324** millimeter of mercury (mmHg) is equal to **25.6019** psi

Function Name: **changeMaker**

Parameters:

None

Return Value:

None

Description:

Write a user-interactive function that will tell the user the coins needed to make a given amount of money with the least coins used. For example, let's say we're trying to make 45 cents worth of change using quarters, dimes, nickels and pennies using the least number of coins possible. The solution would be to use the largest denomination of coin possible at each step to solve the problem, so we would first use a quarter, leaving 20 cents. Then, we would use a dime, leaving 10 cents. Then, we would use another dime leaving 0 cents and making a total of 45 cents. In this example, we created 45 cents using 3 coins, which is the optimum solution for the problem.

You should prompt the user for a **number of cents**, so 102 would be 1 dollar and 2 cents.

1. Get the amount of cents 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 quarters, dimes, nickels, and pennies that could make up the cents the user entered. You should compute each of these numbers in the above order using the following information:
   - Quarters are worth 25 cents
   - Dimes are worth 10 cents
   - Nickels are worth 5 cents
   - Pennies are worth 1 cent
   - 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 of quarters, dimes, nickels, and pennies 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 quarters, 1 dimes, 1 nickels, 0 pennies").

Part 2 – Complex Functions

Now you will write a few python functions that take in input as **PARAMETERS** and **RETURN** output data for practice with the language. These functions should also be put into the HW1.py file.

Function Name: **calcDistance**

Parameters:

1. x1 – a number representing the x coordinate of the first point as an integer
2. y1– a number representing the y coordinate of the first point as an integer
3. x2 – a number representing the x coordinate of the second point as an integer
4. y2 – 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: **makeBLT**

Parameters:

1. bacon - an integer representing the number of bacon strips
2. lettuce - an integer representing the number of lettuce leaves
3. tomatoes - an integer representing the number of tomato slices

Return Value:

An integer representing the number of BLT sandwiches you can make given the inputted stock of ingredients.

Test Cases:

1. makeBLT(150, 20, 40) --> 6
2. makeBLT(1, 5, 51) --> 0
3. makeBLT(11, 3, 200) --> 1

Description:

Write a function that returns the number of BLT sandwiches you can make based on the number of bacon strips, lettuce leaves, and tomato slices specified as inputs. Assume that

it takes the following to make 1 sandwich:

- 10 bacon strips
- 3 lettuce leaves
- 6 tomato slices

Note that the number of sandwiches must be a whole number; you cannot have 1 1/2 of a sandwich. For example, if you have 12 bacon slices, 20 lettuce leaves, and 5000 tomato slices, you can only make 1 sandwich (despite the fact that you have enough lettuce and tomato to make 6 sandwiches).

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!

Function Name: **overallGrade**

Parameters:

1. ex1 – a number between 0 and 100 representing the exam 1 score.
2. ex2– a number between 0 and 100 representing the exam 2 score
3. ex3 – a number between 0 and 100 representing the exam 4 score.
4. ex4 – a number between 0 and 100 representing the exam 4 score.
5. fial -- a number between 0 and 100 representing the final exam.

Return Value:

A floating point number representing the average of all exam scores AFTER the lowest score is replaced with the 2nd lowest score. .

Test Cases:

1. overallGrade(60,70,80,90,0) --> 72.0000
2. overallGrade(88,32.0,44,92,44.0) --> 62.400
3. overallGrade(60,60,70,80,90) --> 72.000

Description:

Write a function overall Grade that will calculate and return a student's 2316 exam average. It will accept five parameters, which correspond to exam1,2,3,4 and the Final grade. It should replace the lowest exam grade with the 2nd lowest exam grade, and then calculate the average exam grade. You may assume that all exams will be a value between 0 and 100.0, and that each exam has the same weight.

Part 3 – Turtles

The final part of the homework involves importing the turtle module and create a simple picture of a video game level. This code will also be put into HW1.py file.

Function Name: **drawLevel**

Parameters:
None

Return Value:
None

Description:
Write a function that uses the turtle module to create a window and draw a video game level in the window. Your level must have at least ten objects, with at least three different styles of objects. You may create helper functions that you call to assist you with drawing the various things. You may use a for or while loop to draw the buildings or objects, but they can not be all the same height/size. You may add extra features such as colors, atmospheric effects, characters, etc. When your turtles are finished drawing, you should use the **exitonclick()** method to display the window until the TA clicks on it.

Grading Breakdown

Simple Function

mmHGToPSI	**5**
changeMaker	**20**

Complex Functions

calcDistance	**15**
makeBLT	**20**
overallGrade	**25**

Turtles

drawLevel	**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.