Homework 3 — Baking Bonanza!

Due Tuesday, September 25th at 11:55pm

(50 points)

Objectives:

- Write a program that uses functions for organization and code reuse
- Learn how to incrementally develop and run programs
- Understand the scope of different variables
- Understand the structure and purpose of the `main` function
- Understand the use of parameters with functions

Turn In:

- `hw3_recipe.ipynb`

You should turn this files in on moodle under the “Homework 3 - Baking Bonanza” link.

Instructions:

Basil wants to make cupcakes for their entire lab, but their recipes are not for the correct number! We want to write a program to help them calculate how much of each ingredient they will need so that everyone in their lab (say, 37 students) can have a cupcake, even if the number of people in their lab changes from week to week.

Write a program that lets the user (like Basil) enter how many cupcakes they want to make, then reports how much of each ingredient they will need. Use **grams** to specify the amount of flour, sugar, and butter, **teaspoons** for vanilla and baking powder, and **number** of eggs for the eggs.

Recipe:

Makes 12 cupcakes

- 100 grams flour
- 100 grams sugar
- 100 grams butter
- 2 eggs
- 1 teaspoon vanilla extract
- 1 teaspoon baking powder

```
How many cupcakes do you want to make? 12
100.0 grams flour
100.0 grams sugar
100.0 grams butter
2.0 eggs
1.0 teaspoons vanilla
1.0 teaspoons baking powder
```

Next, display the same information, but this time convert the amounts in grams to cups so that Basil can still make the recipe even if they’ve lost their scale!

Conversion factors:

- 120 grams flour = 1 cup
- 200 grams sugar = 1 cup
- 240 grams butter = 1 cup

```
Now, in cups!
0.83 cups flour
0.5 cups sugar
0.42 cups butter
2.0 eggs
1.0 teaspoons vanilla
1.0 teaspoons baking powder
```

Full Output:

You should exactly reproduce our output. Notice that the numbers in the expected output are rounded to two decimal places. Use the `round(value, places)` function to help you.

Expected output (second example on next page, user input in **underlined bold**)

```
How many cupcakes do you want to make? 12
100.0 grams flour
100.0 grams sugar
100.0 grams butter
2.0 eggs
1.0 teaspoons vanilla
1.0 teaspoons baking powder

Now, in cups!
0.83 cups flour
0.5 cups sugar
0.42 cups butter
2.0 eggs
1.0 teaspoons vanilla
1.0 teaspoons baking powder
```

```
How many cupcakes do you want to make? 1
8.33 grams flour
8.33 grams sugar
8.33 grams butter
0.17 eggs
0.08 teaspoons vanilla
0.08 teaspoons baking powder

Now, in cups!
0.07 cups flour
0.04 cups sugar
0.03 cups butter
0.17 eggs
0.08 teaspoons vanilla
0.08 teaspoons baking powder
```

Program Structure

You should have at least **3** functions, including your `main` function (you must have at least two functions other than your `main`). Each function other than `main` should be called at least twice. Your non-main functions **must** use parameters and returns.

You should **not have any code outside of functions** other than your "`if __name__ == "__main__"`" statement.

(**Hint!** We recommend one function that adjusts the amount of an ingredient according to how many "full recipes" you are making and returns the adjusted amount and one function that converts an ingredient to cups according to an original weight and a conversion factor.)

Make sure that your `main` function is structured like a "table of contents" rather than a waterfall: Note that this is an example, we expect you to give your functions names that are meaningful to your program, not "chapter1", etc!

Table of contents (good! main has control, calls the functions, control returns to main)	Waterfall (bad! functions are strung together, the first calling the next, and so on)
`def chapter1():`	`def chapter1():`
`    # code`	`    # code`
	`    chapter2()`
`def chapter2():`	
`    # code`	`def chapter2():`
	`    # code`
`def chapter3():`	`    chapter3()`
`    # code`	
	`def chapter3():`
`def main():`	`    # code`
`    chapter1()`	
`    chapter2()`	`def main():`
`    chapter3()`	`    chapter1()`

As a reference, our solution, including empty lines and comments, is about 65 lines.

Development:
Your code should always be in a "runnable" state. Do not attempt to develop your entire program without running it. Make a small part first, make sure it runs correctly (i.e. it does what you expect it to do), then move forward to make the next part and so on.

One way to solve this homework could be:

1) Start by writing your main function. How will you get information from the user about how many cupcakes they want? Run your program. Does it do what you expect?
2) Next, how will you adjust your amounts according to how many cupcakes you will make? Write your code as if you only need to adjust the amounts. (Hint: you should have one function whose job it is to adjust a given amount according to a given adjustment amount, then return that answer.)
3) Once you have your code working for the amount-adjusted cupcakes recipe, now work on converting the amounts to cups. (Hint: do you need another function here to help you convert amounts?)

Comments (5 points)
Your code must be commented. You must include a file comment, inline comments, and function comments. An example function comment is shown below:

```
# This function converts a temperature in fahrenheit to celsius
# and prints the equivalence.
# Parameters: fahrenheit - int or float degrees fahrenheit
# Return: float equivalent temperature in celsius
def fahrenheit_to_celsius(fahrenheit):
     celsius = (fahrenheit - 32) * (5/9)
     return celsius
```

Bonus (5 points)
Add a second recipe to your program! After printing out the cupcakes information, do the same things (ask user for the amount, print the adjusted recipe, print the converted recipe) but with a different recipe.

Note! If your original recipe uses cups instead of grams, the adjusted recipe should use cups and the converted recipe should use grams! You may want to write a new function to help yourself out.

Consult a website such as: https://www.kingarthurflour.com/learn/ingredient-weight-chart.html to help you find conversions.

You should add your new recipe to the output of your original program (you don't need to copy and paste everything again—that's the whole benefit of functions!)