Homework 03: Function decomposition and loops

Introduction

This assignment is designed to give you practice with the following new topics:

- Interpret provided functions and compose them to solve some task.
- Given some output, write code that can generate this output while adhering to some constraints on the code strcuture.
- Implement simple definite loops.

Important Tips

- Remember, computer science is a science. Always write code with a prediction in mind. While you can sparingly code to try and see output, you should focus on thinking through what you want your code to do, what you expect as a result, and compare what your code actually does to your result.
- Before you type, trace the execution of the starter code already provided
- Sometimes sub-tasks go together, like computation followed by printing. You should consider them simultaneously if that helps you follow the correctness of your work.
- Use extra steps if needed for your thought process, printing results to test your code. Just remember to remove these extra prints once you know your code is correct.
- Match your output to the output given. Precision is important in computer science.
- Don't change file names. You're generally given `.py` files to work in. Don't change the filenames of those files.

Your assignment

Your task is to complete following steps:

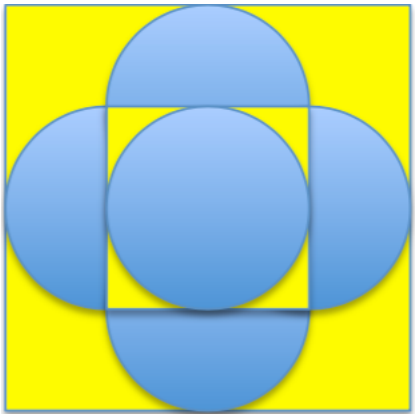
1. Download the `hw3.zip` file and open it. You will see three python files, `hw3_garden.py`, `hw3_deposit.py`, and `hw3_banner.py` in the unzipped folder. You are expected to write your programs in these files.
2. Complete `hw3_garden.py`. This file is used in Part 1.
3. Complete `hw3_deposit.py`. This file is used in Part 2.
4. Complete `hw3_banner.py`. This file is used in Part 3.
5. Review the grading criteria at the end of this assignment.
6. Submit your completed programs.

Notice that each starter `.py` file has a header with some information for you to fill in. Please do so. Your feedback helps the instructors better understand your experiences doing the homeworks and where we can provide better assistance.

Part 1: Garden

For this problem, you are given a partial implementation to calculate and report the supplies needed for a flower garden according to the design shown below. In this geometric pattern, the blue areas represent flowerbeds and the yellow areas use fill materials such as stone and mulch.

The garden is a square within which a central circle and four congruent semicircles form the flowerbeds. This congruency simplifies the calculations between the flowerbeds as the central circle reappears in the outer shapes. Examine this geometry to understand fully the computations of the provided functions.



Your task is to complete the implementation (in `hw3_garden.py`). The program should prompt the user for the following information:

1. The side length (in feet) of the square garden.
2. The recommended spacing (in feet) between plants.
3. The depth (in feet) of the flowerbeds (blue areas).
4. The depth (in feet) of the filled areas (yellow areas).

Then the program calculates the number of plants for the central circle and each of the semicircle, before giving the total number of plants.

Specifically, to estimate the number of plants for a flowerbed, its area is divided by the area needed per plant, which is the square of the recommended distance between plants. This result is truncated to be the number of plants per flowerbed.

Thereafter the amount of soil and fill material are computed and displayed. Specifically, the program should report four cubic yards values (all rounded to one decimal place):

- The amount of soil for the central circle flowerbed
- The amount of soil for each semicircle flowerbed
- The total amount of cubic soil for the garden
- The amount of material needed to fill the rest of the garden

Note that there are 3 (linear) feet in 1 (linear) yard.

Task A: Add docstrings

Read and understand the provided functions to fill in their docstrings.

Task B: Write `main`

Write the `main` function, effectively using the provided functions.

Example output 1

```
Enter length of side of garden (feet): 10
Enter spacing between plants (feet): .5
Enter depth of garden soil (feet): .8333
Enter depth of fill (feet): .8333

Plants for the circle garden: 78
Plants for each semicircle garden: 39
Total plants for garden: 234

Soil for the circle garden: 0.6 cubic yards
Soil for each semicircle garden: 0.3 cubic yards
```

```
Soil for all the beds of the garden: 1.8 cubic yards
Fill for the rest of the garden: 1.3 cubic yards
```

Example output 2

```
Enter length of side of garden (feet): 13
Enter spacing between plants (feet): 0.25
Enter depth of garden soil (feet): 0.5
Enter depth of fill (feet): 0.25

Plants for the circle garden: 530
Plants for each semicircle garden: 265
Total plants for garden: 1590

Soil for the circle garden: 0.6 cubic yards
Soil for each semicircle garden: 0.3 cubic yards
Soil for all the beds of the garden: 1.8 cubic yards
Fill for the rest of the garden: 0.6 cubic yards
```

Part 2: Bank deposits

In `hw3_deposit.py` write a program that asks a user for their current bank balance, and the number of deposits they want to make. Then, for each deposit that the user makes, it prints the current balance after the deposit.

Example output 1

```
Welcome to the River Bank!

What is the current balance in your account? 100.5
How many deposits would you like to make? 3

################################################

Amount for deposit 1: 50.5
Thank you for the deposit! Your balance is now 151.0

################################################

Amount for deposit 2: 2000
Thank you for the deposit! Your balance is now 2151.0

################################################

Amount for deposit 2: 140.75
Thank you for the deposit! Your balance is now 2291.75

################################################

Thank you for your 3 deposit(s).
Hope to see you again at River Bank!
```

Example output 2

```
Welcome to the River Bank!

What is the current balance in your account? 25
How many deposits would you like to make? 1

################################################
```

```
Amount for deposit 1: 2000
Thank you for the deposit! Your balance is now 2025.0

################################################

Thank you for your 1 deposit(s).
Hope to see you again at River Bank!
```

Example output 3

```
Welcome to the River Bank!

What is the current balance in your account? 25
How many deposits would you like to make? 0

################################################

Thank you for your 0 deposit(s).
Hope to see you again at River Bank!
```

Task A: Implement functions

We've given you four function headers with docstrings. Implement those functions.

Task B: Compose the functions in `main()`

Use the functions you've written to perform the task.

Part 3: Course banners

In `hw3_banner.py`, write a program that will create two course specific banners. Specifically, your task is to prompt the user for information about two courses and display a well-formatted banner for each of them. Specifically, for each course, the user is asked to enter

1. The course's name,
2. The course's room,
3. The duration in minutes of one lecture/lab session, and
4. The number of times the course meets per week.

Here is an example for how to prompt the user for the information.

```
Enter the first course: cosc101
Enter the location: Bernstein Hall 204
Enter the duration of one lecture/lab (mins): 50
Enter how many times we met a week: 3
```

Based on this information, your task is to create the following banner with the text centered in a box.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                 !!Welcome to cosc101!!                   +
+                     We will meet in                      +
+                   Bernstein Hall 204                     +
+     for a total of 2 hour(s) and 30 min(s) per week      +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Example output

Here is an example output for the two banners combined. **Note: the program prompts the user once at the start for a symbol to use in the borders of the banners**

```
Enter the one sign to use to make your banners: *
```

```
Enter the first course: COSC101
Enter the location: 214 Bernstein Hall
Enter the duration of one lecture/lab (mins): 50
Enter how many times we met a week: 3

*********************************************************
*                  !!Welcome to COSC101!!                *
*                      We will meet in                   *
*                    214 Bernstein Hall                  *
*     for a total of 2 hour(s) and 30 min(s) per week     *
*********************************************************


Enter the second course: COSC101L
Enter the location: 303 Bernstein
Enter the duration of one lecture/lab (mins): 113
Enter how many times we met a week: 1

*********************************************************
*                  !!Welcome to COSC101L!!               *
*                      We will meet in                   *
*                       303 Bernstein                    *
*     for a total of 1 hour(s) and 53 min(s) per week     *
*********************************************************
```

Constraints

- You should use exactly the words shown that are not part of the input the user enters.
- You can assume the last line of text (`for a total of ... per week`)
  - is the longest, and
  - include 5 spaces on either side.
- Your `main()` function should have fewer than 10 lines of code
- You need to include a minimum of three other functions, with at least one being a fruitful one.

Task A

For each line of the banner, describe the computation(s) required to be able print the line. Include the answer to this in the header of `hw3_banner.py`.

Task B

Implement your functions and put them together in a `main()` function. Remember to add docstrings!

Hint: Remember, `len(x)` gives the number of characters in string `x`

Submission Instructions

Submit three Python files to the platform indicated in your class section:

- `hw3_garden.py`
- `hw3_deposit.py`
- `hw3_banner.py`

Remember to complete the questions at the top of each file before submitting.