

COSC 101C Homework 3: Fall 2024

The due date for this homework is **Friday, September 27th, 6:00 pm EDT**

Introduction

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

- Procedural decomposition and composition
- Function mechanisms: passing arguments and return
- `for` loops with `range` function
- The accumulator pattern
- Using built-in and module functions

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_chant.py`, and `hw3_forest.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 **Task 1**.
3. Complete `hw3_chant.py`. This file is used in **Task 2**.
4. Complete `hw3_forest.py`. This file is used in **Task 3**.
5. Review the grading criteria below.
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.

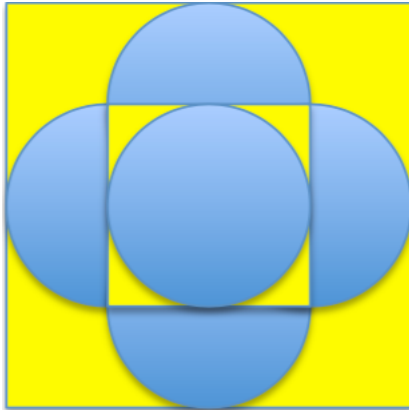
Grading

When we are assessing your code, higher levels of achievement are demonstrated, in part, by - Starter code left unmodified - All lines of output are present - Lines of output have proper formatting, including spacing and blank lines - Use spaces and blank lines in code for readability - Variables are appropriately named and used - Functions are appropriately named and definitions include type annotations - Functions are used to capture patterns and minimize repeated code - Functions are used to break down problems and employ abstraction - Loops are appropriately used to abstract repeated patterns

Task 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
```

Task 2: Chant

In `hw3_chant.py` write a program that asks a user for a word and prints a pep band chant in a particular format. If the user types `HOME`, the output should be:

```
Give me a word: HOME
Leader: Give me an H!
Band: H!
Leader: Give me an O!
Band: HOO!
Leader: Give me an M!
Band: HOOMMM!
Leader: Give me an E!
Band: HOOMMMEEEE!
```

As always you should carefully study the pattern before coding to plan its implementation. Make sure to observe the format of each line and understand the patterns in the score. You must exactly respect the punctuation and spaces. For example, notice that each line ends with the exclamation point—*there should be no space between the exclamation point and the text.*

Below is another example of input and output.

```
Give me a word: goalie
Leader: Give me an g!
Band: g!
Leader: Give me an o!
Band: goo!
```

```

Leader: Give me an a!
Band: gooaaa!
Leader: Give me an l!
Band: gooaaallll!
Leader: Give me an i!
Band: gooaaallllliiii!
Leader: Give me an e!
Band: gooaaallllliiiiieeeeeee

```

Task 3: Forest

Your task is to design and implement (in `hw3_forest.py`) a program that creates a drawing of a tree forest as text art.

Task A: Basic Tree

Write code to draw the particular text-art pattern that represents a decorated evergreen tree:

```

      \ | /
     --0--
      / | \
     // | \\
    /// |  \\
   //// |   \\
  ///// |    \\
 0 0  |||  0 0
  _|||_

```

The drawing is made using only seven distinct characters: spaces (), underscores (`_`), vertical bars (`|`), common slash (`/`), backslash (`\`), hyphen-minus (`-`) and (`0`).

You **must** use separate functions for different features and to increase reusability while reducing repetition in your code. Note that there are unique segments to the figure as well as *repeated elements* which lend themselves naturally to decomposition in functions as well as the use of one `for` loop for the tree cone.

Text Art Hints

Printing the backslash character `\` is tricky because python uses the backslash to produce special characters. For example, the string `"\t"` is a tab and `"\n"` is a newline. A single backslash character is represented by a string of two backslash characters (`"\\"`). For example:

```
print("/\\//\\")
```

will print: `/\\//\\`

Examining the picture carefully is essential, taking note of spacing and patterns. You should always work incrementally, testing each decomposition before composing them together. For example, you should work on one feature at a time, testing it before moving to the next one. You may work on the pattern omitting the spaces at first, getting the left-aligned version, before centering the tree.

Task B: Trees

Modify your code to allow the following user input: * the number of trees in a row and * the distance between each tree. This number defines the number of spaces separating their closest characters, which are the right most ball (`0`) and the left most one of the next tree on its left.

Example output 1

Enter the number of trees in a row: 2
Enter the distance between each tree: 10

```

      \|\ /
    --0--
      /\ \
     //  \ \
    ///   \ \ \
   ///    \ \ \ \
  ///     \ \ \ \ \
 ///      \ \ \ \ \ \
0 0 | | | 0 0
  _|_|_|_

```

Enter the number of trees in a row: 5
Enter the distance between each tree: 3

```

      \|\ /      \|\ /      \|\ /      \|\ /      \|\ /
    --0--      --0--      --0--      --0--      --0--
      /\ \      /\ \      /\ \      /\ \      /\ \
     //  \ \    //  \ \    //  \ \    //  \ \    //  \ \
    ///   \ \ \  ///   \ \ \  ///   \ \ \  ///   \ \ \  ///   \ \ \
   ///    \ \ \ \  ///    \ \ \ \  ///    \ \ \ \  ///    \ \ \ \  ///    \ \ \ \
  ///     \ \ \ \ \  ///     \ \ \ \ \  ///     \ \ \ \ \  ///     \ \ \ \ \  ///     \ \ \ \ \
 ///      \ \ \ \ \ \  ///      \ \ \ \ \ \  ///      \ \ \ \ \ \  ///      \ \ \ \ \ \  ///      \ \ \ \ \ \
0 0 | | | 0 0 0 0 0 0 0 0 | | | 0 0 0 0 0 0 0 0 | | | 0 0 0 0 0 0 0 0 | | | 0 0 0 0 0 0 0 0
  _|_|_|_      _|_|_|_      _|_|_|_      _|_|_|_      _|_|_|_

```

Task C: A forest

Modify your code to make the output a forest. Your code should now ask first a user input to provide the number of rows in the forest.

Example output 1

Enter the number of rows: 2
Enter the number of trees in a row: 4
Enter the distance between each tree: 2

```

      \|\ /      \|\ /      \|\ /      \|\ /
    --0--      --0--      --0--      --0--
      /\ \      /\ \      /\ \      /\ \
     //  \ \    //  \ \    //  \ \    //  \ \
    ///   \ \ \  ///   \ \ \  ///   \ \ \  ///   \ \ \
   ///    \ \ \ \  ///    \ \ \ \  ///    \ \ \ \  ///    \ \ \ \
  ///     \ \ \ \ \  ///     \ \ \ \ \  ///     \ \ \ \ \  ///     \ \ \ \ \
 ///      \ \ \ \ \ \  ///      \ \ \ \ \ \  ///      \ \ \ \ \ \  ///      \ \ \ \ \ \
0 0 | | | 0 0 0 0 | | | 0 0 0 0 | | | 0 0 0 0 | | | 0 0 0 0
  _|_|_|_      _|_|_|_      _|_|_|_      _|_|_|_

      \|\ /      \|\ /      \|\ /      \|\ /
    --0--      --0--      --0--      --0--
      /\ \      /\ \      /\ \      /\ \
     //  \ \    //  \ \    //  \ \    //  \ \
    ///   \ \ \  ///   \ \ \  ///   \ \ \  ///   \ \ \
   ///    \ \ \ \  ///    \ \ \ \  ///    \ \ \ \  ///    \ \ \ \
  ///     \ \ \ \ \  ///     \ \ \ \ \  ///     \ \ \ \ \  ///     \ \ \ \ \
 ///      \ \ \ \ \ \  ///      \ \ \ \ \ \  ///      \ \ \ \ \ \  ///      \ \ \ \ \ \
0 0 | | | 0 0 0 0 | | | 0 0 0 0 | | | 0 0 0 0 | | | 0 0 0 0
  _|_|_|_      _|_|_|_      _|_|_|_      _|_|_|_

```

Example output 2

Enter the number of rows: 4

Enter the number of trees in a row: 2
Enter the distance between each tree: 7

```

      \  /
     --0--
      /  \
    /  |  \
   /  |  \
  /  |  \
 /  |  \
/  |  \
0 0 | 0 0
  _|_|_

```

```

      \  /
     --0--
      /  \
    /  |  \
   /  |  \
  /  |  \
 /  |  \
/  |  \
0 0 | 0 0
  _|_|_

```

```

      \  /
     --0--
      /  \
    /  |  \
   /  |  \
  /  |  \
 /  |  \
/  |  \
0 0 | 0 0
  _|_|_

```

```

      \  /
     --0--
      /  \
    /  |  \
   /  |  \
  /  |  \
 /  |  \
/  |  \
0 0 | 0 0
  _|_|_

```

```

      \  /
     --0--
      /  \
    /  |  \
   /  |  \
  /  |  \
 /  |  \
/  |  \
0 0 | 0 0
  _|_|_

```

```

      \  /
     --0--
      /  \
    /  |  \
   /  |  \
  /  |  \
 /  |  \
/  |  \
0 0 | 0 0
  _|_|_

```

```

      \  /
     --0--
      /  \
    /  |  \
   /  |  \
  /  |  \
 /  |  \
/  |  \
0 0 | 0 0
  _|_|_

```

```

      \  /
     --0--
      /  \
    /  |  \
   /  |  \
  /  |  \
 /  |  \
/  |  \
0 0 | 0 0
  _|_|_

```

Submission Instructions

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

- hw3_garden.py
- hw3_chant.py
- hw3_forest.py

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