

COSC 101 Homework 2: Spring 2016

The due date for this homework is **Thursday, Feb. 12, 5:00pm**.

Upload your solution to moodle by the deadline.

Introduction

This assignment is designed to give you a first introduction to writing programs in python and most importantly to practice for Exam 1. By completing this assignment, the warm-up part and the hand-in problems, you will demonstrate that you understand a number of important concepts:

- how to exchange input and output with the user using `raw_input` and `print`
- how to work with multiple data types (`str`, `int`, and `float` types) and how to change types (e.g., from `str` to `float` and `float` to `int`)
- how to use the turtle module to draw geometric shapes
- how to write loops to draw a text-art pattern.

I encourage you to start now and to complete the three warm-up problems this week. You will have next week to do the four problems to hand-in.

Warm-up

Open the `basicProgram_warmup.py` file in IDLE, which contains two problems.

The first problem starts with three `print` statements: observe the code and its output. Notice the special character that is backslash `\`: it is the escape character used to print out other special characters. Think about it we have for example double quotes `"`, which are part of a string definition. What if we wanted to use to have double quote in the output: we use a backslash to differentiate between the two meanings. We do the same if we want to print a backslash itself.

Once it makes sense apply what you learned to print out exactly the three lines given.

The second problem has a long description. Take notes to comprehend the problem. Make sure the three tests described in details (i.e. for a given input the output of the correct program) make sense. On paper sketch the steps of your solution first loosely, i.e. a mix of English and diagrams, and then refine your work to be more pseudo-code like and then close to code.

Code your answer incrementally, fix syntax as you go and then debug your completed code with the test examples, which where given and finally try your own tests.

The third warm-up problem is in a different file.

Turtle warm-up

Open the `turtle_warmup.py` file and read it carefully. Using the `turtle` module, which you discover in lab you are asked to code the drawing of a sock according to the input length L .

Master this problem so that you will be able to create a similar `turtle` program of your own design.

Do not submit any of this work but please come to Open Lab or to see me if you have difficulties. We are here to help and these problems are given to make sure you master concepts before you complete the next part, which is the assignment problems that will be graded.

Assignment: Problems to hand-in

Program 1: Mad libs

Open the file `hw2_madlibs.py` and write your answer to be submitted in it.

Write a short program that asks for a noun (string) and two verbs (both strings). (You can assume that both verbs are in third person form.) Using the three inputs, your program should print the sentence:

If it <verb 1> like a <noun> and <verb 2> like a <noun>, it probably is a <noun>.

Below is an example execution of the program where the user types “duck,” “walks”, and “talks.”:

```
What's the noun? duck
What's the first verb? walks
What's the second verb? talks
```

If it walks like a duck and talks like a duck, it probably is a duck.

A second example where the user types “loon,” “swims,” and “dives.”:

```
What's the noun? loon
What's the first verb? swims
What's the second verb? dives
```

If it swims like a loon and dives like a loon, it probably is a loon.

Program 2: Race

You have to write a personalized race-pace calculator!

Open the file `hw2_race.py` and write your answer (code and commented tests) to be submitted in it.

Write a program that asks for

- the runner name,
- the distance (in miles) of the race and
- the time (in minutes and in seconds) the runner desired to finish the race. Ask for minutes and seconds separately.

Your program should compute and print the average pace per mile required.

Your program’s output should look *exactly* like the example provided below.

```
What is your name? Joel
What is the race distance? 3.1
How fast do you want to finish? Give me the minutes: 20
And the seconds: 15
Joel, you should run each mile at a 6:31 pace.
```

Write as comments the input and output you tried at the Python console. Make sure you are fully testing your program.

Program 3: Creative Turtle

Inspired by the turtle program that draws a sock, which you completed in the warm-up part, you have now to create your own design on paper so as to then code it in Python.

I encourage you to be creative to make your own art work/object using the turtle module. The complexity of the object you are deciding to draw should be at least as complex than the sock and have similarly a dependence on a length L .

On paper draw the object your turtle program will draw. For example, it could be a house, a lamp, a flower... you decide. Make a careful diagram that will guide your program.

You have to bring this paper diagram in class Friday the 12th of February to show me what you planned before writing your program. Your diagram does not need to be color but should show dimensions relative to L . The sock is the example to follow in terms of the diagram.

Next in a new file named `myturtle.py` write a turtle program that asks the user for the value L and that then draw your object scaled according to this L value.

If you decided to work in pair you have to submit two designs and two corresponding program.

Program 4:

Open the file `hw2_volcano.py` to write your program for this last problem.

Your task is write a program for the volcano text-art pattern. Your text art must be **scalable**, meaning that the art is not a fixed sequence of characters, but a pattern that can be drawn at a variety of scales as shown on the following webpage: [Volcano](http://cs.colgate.edu/cosc101/a/hw/web/volcano.html), (<http://cs.colgate.edu/cosc101/a/hw/web/volcano.html>).

The program must ask the user for a positive number and then scale the text pattern accordingly. Note that the scale captures some notion of the size of the art, but it is not necessarily equal to the width or height.

Examine the pictures carefully, taking note of spacing and patterns.

Break the problem down into two major steps:

1. write code to draw the pattern and
2. get the pattern to scale with user input.

To complete step 1, choose a scale, say scale 3, and write a program that reproduces the scale 3 pattern exactly. At this point, the program should not take any user input. Once you have this working, figure out how the pattern changes with user input.

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 "\\ /\\" data-bbox="111 785 230 801" data-label="Text">

will print: /\


```

If you find it too challenging start by solving this easier [pattern](#).

If you want a more difficult challenge try [this one](#) or make your own. Submit your own in a separate file but do not submit the tree or rocket, which are for your own good!

Grading

Your assignment will be graded on two criteria:

1. Correctness[80%]: be sure that your programs produce output that is **identical** to the provided examples.
2. Program design and style [20%]: style, program design and tests become increasingly important the more complex your program becomes. For these programs, adhere to the following guidelines:
 - Variable names should be meaningful
 - Programs should contain at least a few descriptive comments. Do NOT comment every line of code with low level explanations of what each line does. Focus on high level ideas (e.g., **# now drawing the blue triangle**).
 - Tests should be included as comments for program 2 (maybe for program 1 a couple of them will show you are thinking about special cases).