

CSCI 1100 — Computer Science 1

Fall 2023

Homework 1: Calculations and Strings

Overview

This homework, worth **100 points** total toward your overall homework grade, is due **Thursday, September 14, 2023** at 11:59:59 pm. The three parts should each be submitted separately. All parts should be submitted by the deadline or your program will be considered late.

Please refer to the **Submission Guidelines** document before starting this assignment. It will give you details on what we expect and will answer some of the more common questions, including that you need to submit your program through **Submitty** and that **Submitty** will open by **Monday, September 11th, 2023**.

Remember that your output must match **EXACTLY** the format shown in example runs from the `hw01_files.zip` file. The purpose of this is to make testing easier and at the same time teach you how to be precise in your programming using the tools we gave you. We love creativity, but not in HW output formatting!

Part 1: Mad Libs (40 pts)

In this part you will write a Python program to construct the Mad Lib given below:

```
Good morning <proper name>!
```

```
This will be a/an <adjective> <noun>! Are you <verb> forward to it?  
You will <verb> a lot of <noun> and feel <emotion> when you do.  
If you do not, you will <verb> this <noun>.
```

```
This <season> was <adjective>. Were you <emotion> when <team-name> won  
the <noun>?
```

```
Have a/an <adjective> day!
```

You will ask the user of the program for the missing words — those enclosed in `< >` — using the `input` function. You will then take all the user specified inputs and construct the above Mad Lib. (Be sure to reread the *Input format for homework assignments in this class* discussion in the **Submission Guidelines**.) Make sure your output looks like the above paragraph, except that the missing information is filled in with the user input.

An example of the program run (how it will look when you run it using **spyder**) is provided in file `hw1_part1_output.txt`. (In order to access this file, you will need to download file `hw01_files.zip` from the **Course Materials** section of **Submitty** and unzip it into your directory for HW 1.)

We've provided reasonable inputs, but the idea of Mad Libs is to input random words and see how silly the result looks. Try it!

Of course, the program you write will only work for the specific Mad Lib we've written above. A more challenging problem, which you will be capable of solving by the end of the semester, is to write a program that reads in **any** Mad Lib, figures out what to ask the user, asks the user, reads the input, and generates the final Mad Lib.

Test your code well and when you are sure that it works, please submit it as a file named **hw1_part1.py** to Submittity for Part 1 of the homework.

Part 2: Speed Calculations (40 pts)

Many exercise apps record both the time and the distance a user covers while walking, running, biking, or swimming. Some users of the apps want to know their average pace in minutes and seconds per mile, while others want to know their average speed in miles per hour. In many cases, we are interested in projected time over a specific distance. For example, if I run 6.3 miles in 53 minutes and 30 seconds, my average pace is 8 minutes and 29 seconds per mile, my average speed is 7.07 miles per hour and my projected time for 2.7 miles is 22 minutes and 55 seconds.

Your job in Part 2 of this homework is to write a program that asks the user for the minutes, seconds, miles run, and miles to target from an exercise event and outputs both the average pace and the average speed.

An example of the program run (how it will look when you run it using **spyder**) is provided in file `hw1_part2_output.txt`. (Can be found inside the `hw01_files.zip` file.)

You can expect minutes and seconds to both be integers, but miles and miles to target will be floats. All minutes and seconds must be maintained as integers so please use integer division and modulo operations. Note that if you have a float value then the function `int` gives you the integer value. For example:

```
>>> x = 29.52
>>> y = int(x)
>>> print(y)
29
```

The output for the speed will be a float and should be printed to 2 decimal places. Notice also that our solution generates a blank line before the output of calculations.

We will test your code for the values used in our examples as well as a range of different values. Test your code well and when you are sure that it works, please submit it as a file named **hw1_part2.py** to Submittity for Part 2 of the homework.

Part 3: Framed Box (20 pts)

Write a program that asks the user for a frame character, and then the height and width of a framed box. Then output a box of the given size, framed by the given character. Also, output the dimensions of the box centered horizontally and vertically inside the box. In case perfect vertical centering is not possible, dimensions should be output such that there is one less row above the text than below it. In case perfect horizontal centering is not possible, dimensions should be output such that there is one less space character to the left of the text than to the right.

Assume that the user inputs valid values for each input: width is a positive integer (7 or higher) and height is a positive integer (4 or higher), and a single character is given for the frame.

Two examples of the program run (how it will look when you run it using **spyder**) are provided in files `hw1_part3_output_01.txt` and `hw1_part3_output_02.txt`. (Can be found inside the `hw01_files.zip` file.)

You will need to put the box dimensions in a string first, and then use its length to figure out how long the line containing the dimensions should be. If you have prior programming experience you

might be tempted to look for how Python implements “loops” in order to generate the full frame, but Python provides string manipulation tools (Lecture 3) that make this unnecessary. You must not use any `if` statements or loops in your program. We have not learned them yet, they are not needed, and they will not make your solution better or more elegant.

We will test your code for the values used in our examples as well as a range of different values. Test your code well and when you are sure that it works, please submit it as a file named **hw1_part3.py** to Submittity for Part 3 of the homework.