CSCI 1100 — Computer Science 1
Fall 2018
Homework 2: Strings and Functions

Overview

This homework, worth **60 points** total toward your overall homework grade, is due Thursday, October 4, 2018 at 11:59:59 pm. The two parts should each be submitted separately. All parts should be submitted by the deadline or your program will be considered late.

Note on grading. Make sure you read the `Submission Guidelines` document. It applies to this and all future homework assignments and will be of increasing importance. In all parts of the homework, we will specify which functions you must provide. Make sure you write these functions, even if they are very simple. Otherwise, you will lose points. We will write more complex functions as the semester goes on. In addition, we will also look at program structure (see Lecture 4), and at the names of variables and functions in grading this homework.

You are not allowed to use any loops anywhere in this assignment.

Fair Warning About Excess Collaboration

For all homework assignments this semester we will be using software that compares **all** submitted programs, looking for inappropriate similarities. This handles a wide variety of differences between programs, so that if you either (a) took someone else's program, modified it (or not), and submitted it as your own, (b) wrote a single program with one or more colleagues and submitted modified versions separately as your own work, or (c) submitted (perhaps slightly modified) software submitted in a previous year as your software, this software will mark these submissions as very similar. All of (a), (b), and (c) are beyond what is acceptable in this course — they are violations of the academic integrity policy. Furthermore, this type of copying will prevent you from learning how to solve problems and will hurt you in the long run. The more you write your own code, the more you learn.

Please read the **Collaboration Policy** document for acceptable levels of collaboration and how you can protect yourself. The document was/will be distributed during class and can also be found on the course resources page on **Submitty** `https://bit.ly/2OZh6w7`. Penalties for excess collaboration can be as high as:

- 0 on the homework, and
- an additional overall 5% reduction on the semester grade.

Penalized students will also be prevented from dropping the course. More severe violations, such as stealing someone else's code, will lead to an automatic F in the course. A student caught in a second academic integrity violation will receive an automatic F.

By submitting your homework you are asserting that you both (a) understand the academic integrity policy and (b) have not violated it.

Finally, please note that this policy is in place for the small percentage of problems that will arise in this course. Students who follow the strategies outlined above and use common sense in doing so will not have any trouble with academic integrity.

Part 1: A Penny for a Gum Ball Mickey (30 pts)

Thanks for the Gum Ball!

We are going to conduct an experiment in selling gum balls, but we are going to make a few assumptions. Assume that you are selling gum balls from a vending machine. The machine is a cube, and the gum balls are spheres. You check the machine once a week. The goal is to size the machine so that it is completely full at the start of the week, you never run out of gum balls before you get back to check the machine, and you do not have too many gum balls left at the end of the week to go stale. We will make the assumption that all the gum balls line up so that the number of gum balls along any dimension of the cube is simply the side length divided by the diameter of the spheres. So if the side length is 9.0 and the gum balls have a radius of 0.5 exactly 9 gum balls would fit along each dimension, or 729 gum balls in total. This is known as a `cubic lattice`.

Do the following:

1. First write two functions, `find_volume_sphere(radius)` and `find_volume_cube(side)` that calculate the volume of a sphere given a radius, and that calculate the volume of a cube given a side, respectively.
2. Now ask the user for the radius of the gum balls and the weekly sales.
3. Calculate the total number of gum balls that need to fit in the machine as 1.25 times the weekly sales and use this to calculate the side length of the machine in terms of an integer number of gum balls. **Hint:** You know the total number of gum balls and in a cubic lattice, you can fit the same number along each dimension, so if you can fit `N` gum balls along each dimension, then the machine holds N^3 gum balls. The math module function `ceil` will always round up and may be of use here. (We won't be cutting gum balls to fit them in the machine.)
4. Calculate a few more values: how many gum balls will actually fit given the dimension you chose (remember that there must be an integer number of gum balls along each dimension of the cube); the volume of the cube; the volume of the gum balls, and the wasted space if we put in both the number of gum balls we need to hold and how many we can hold.
5. Print these values out using the `.2f` format for all floating point values. Sample output is below.

```
Enter the gum ball radius (in.) => 0.5
0.5
Enter the weekly sales => 1
1

The machine needs to hold 2 gum balls along each edge.
Total edge length is 2.00 inches.
Target sales were 2, but the machine will hold 6 extra gum balls.
Wasted space is 6.95 cubic inches with the target number of gum balls,
or 3.81 cubic inches if you fill up the machine.
```

or

```
Enter the gum ball radius (in.) => 0.4
0.4
Enter the weekly sales => 983
983

The machine needs to hold 11 gum balls along each edge.
Total edge length is 8.80 inches.
Target sales were 1229, but the machine will hold 102 extra gum balls.
Wasted space is 352.00 cubic inches with the target number of gum balls,
or 324.65 cubic inches if you fill up the machine.
```

We will test your code for the above values 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 **hw2_part1.py** to Submitty for Part 1 of the homework.

Part 2: A night at the races (30 pts)

In this part of the homework, we will play a word game based on the idea of a racetrack. We have provided you with a module, `track.py` in the Course Materials section of Submitty. The module provides two functions, (1) `get_number_of_tracks()` which takes no parameters and returns the number of tracks it can provide and (2) `get_track(track)` which takes an integer track number and returns the corresponding track as a string. Tracks are just strings with two different sets of letters. All of the letters in `'bend'` (`'b', 'e', 'n', 'd'`) indicate curve segments of the track and all of the letters in `'straight'` indicate straight segments. Each track segment represents one-quarter of a mile.

You will need to write two functions. `calculate_curve(track, speed)` and `calculate_straight(track, speed)` each of which takes a string representing the track and returns a 2-tuple of the total distance and total time for the car on the appropriate (curved or straight) segments.

In your main program, first find out how many tracks the module provides and then ask the user to pick a track between 1 and the number of tracks available. Also, ask for the car's speed on the curved and straight segments. Be careful here! We will be providing a **track** module on Submitty that provides a different number and configuration of tracks, so do not assume values. **Use the calls we provide.**

Call your functions to calculate the distance and time spent on the curved and straight segments repectively, and then calculate average speed as well.

Print out the track (it is just a string) and the information you calculated. Finally, print a critique of the car's performance as follows:

1. if the average speed is less than 60, print `"Kind of slow."`
2. if it is at least 60, but less than 120, print `"Getting there."`
3. If it is at least 120, print `"Wow, what a car!"`

Here are some examples:

```
Select a track between 1 and 3 => 1
```

```
1
Speed on curved segments (MPH) => 40
40
Speed on straight segments (MPH) => 120
120

Track:

 E
B n
 d

is 1.00 miles long. You raced it in 90.00 seconds at an average speed of 40.00 MPH
Kind of slow.
```

---

or

```
Select a track between 1 and 3 => 2
2
Speed on curved segments (MPH) => 40
40
Speed on straight segments (MPH) => 120
120

Track:

sTRaight

is 2.00 miles long. You raced it in 60.00 seconds at an average speed of 120.00 MPH
Wow, what a car!
```

---

or

```
Select a track between 1 and 3 => 3
3
Speed on curved segments (MPH) => 76
76
Speed on straight segments (MPH) => 131.3324
131.3324

Track:

  sTRaiiGht
 D         B
n           e
e          N
 b        d
  thGiartS

is 6.25 miles long. You raced it in 211.24 seconds at an average speed of 106.52 MPH
Getting there.
```

Test your code well and when you are sure that it works, please submit it as **hw2_part2.py**.