CSCI 1100 — Computer Science 1
Spring 2017
Homework 2: Strings and Functions

Overview

This homework, worth **75 points** total toward your overall homework grade, is due Thursday, September 28, 2017 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.

Note on grading. Make sure you read the `Submission Guidelines` document. They apply to this and all future homeworks 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. Overall, autograding will account for 60 out of the 75 available points (20 points/part). The remaining 15 points (5 points / part) will be assigned by your TA based on program structure and understandability.

Part 1: A Penny for a Gum Ball Mickey (25 pts)

Thanks for the Gum Ball!

We are going to conduct an experiment in guessing gum balls, but we are going to make a few assumptions. Assume that the container of gum balls is a cube, and that the gum balls are spheres. 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_sphere_volume(radius)` and `find_cube_volume(side)` to calculate the volume of a sphere given a radius and a cube given a side length
2. Now ask the user for the side length of the cube and the radius of the gum balls.
3. Calculate the total number of gum balls that can be contained in the cube, remembering that the number of gum balls along each dimension must be an integer.
4. Calculate the volume of the cube, the total volume of all the gum balls, and the ratio of the total volume of gum balls to the volume of the cube to get a percent. We will refer to this ratio of volumes as the density.
5. Now lets see what would happen if we change the shape of the container and keep the density of gum balls the same. Calculate the volume of a sphere with a diameter the same as the side length of the cube.
6. Assume the density of gum balls remains the same, how many gum balls can fit in the sphere? **Hint:** Calculate the volume of the sphere, multiply by the density, and divide by the volume

of a single gum ball. The number of gum balls must be an integer. A run of the program is below.

```
Enter the side length of the cube (in.) => 12
12
Enter the radius of the gum ball (in.) => 1
1
A box of side length 12.0 will hold 216 gum balls of radius 1.0.
The gum balls will take up 904.78 in^3
of the total volume of 1728.00 in^3 or 52.36%
A sphere with a diameter of 12.0 would have volume 904.78 in^3
It would hold 113 gum balls at the same density.
```

We will test your code for the above values as well as range of different values. You should do the same. Pay attention to the density calculated by your trial runs. What do you notice? For runs where the radius of the gum balls are small with respect to the side of the cube what is the density?

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: Are days getting longer or is this lecture really boring? (25 pts)

We learn that an Earth Day is 24 hours long, but to be precise it is 23 hours 56 minutes and 4 seconds long. The length of a day has not always been the same either, it has been getting longer. How can you tell this? Apparently, you can count bands on some shells which grow following a lunar cycle to find how many days there were in a year historically.

Write a program that assumes we are in 2017 (crazy I know), a day is 23 hours 56 minutes and 4 seconds long and days are getting longer at a rate of 6 hours every 900 million years. These are the only values you can hard code into your program (but not their second equivalents). Of course you can use constants like 60 minutes per hour and 60 seconds per minute.

The program asks the user for a future year and then finds and prints the length of the day in that given year.

Here is an example run of this program:

```
A day in 2017 is 23 hours 56 minutes and 4 seconds long.
Which is equivalent to 86164 seconds
Enter a future year => 123456789
123456789
A day in year 123456789 will be 89126 seconds long
which is equivalent to 24 hours 45 mins 26 secs
```

To implement your program, you must write and call two separate functions.

The first function `time_to_seconds` takes as input three values representing a length of time in terms of number of hours, minutes and seconds, and returns total number of seconds equivalent to the input length of time.

The second function `seconds_to_str` should take as input number of seconds and returns a string representing hours, minutes and seconds for the given time. Example runs of these functions are given below.

```
>>> time_to_seconds(21,45,18)
78318
>>> seconds_to_str(78318)
'21 hours 45 mins 18 secs'
```

Test your code well and when you are sure that it works, please submit it as a file named **hw2_part2.py** to Submitty for Part 2 of the homework.

Part 3: How do you feel about homeworks? (25 pts)

In this part of the homework, you will implement a very rough sentiment analysis tool. While the real tools use natural language processing, they all use word counts similar to the one we use here. Understanding the sentiment in messages is a crucial part of a lot of artificial intelligence tools.

Write a program that will ask the user for a string containing a sentence. The program will then compute the happiness and sadness level of the sentence is using two functions described below. If the happiness level is higher than sadness level, then the tone of the sentence is happy. If otherwise the sadness level is higher, then the tone of the sentence is sad. Otherwise, it is neutral. Find and print the tone of the sentence.

Here is an example run of this program:

```
Enter a sentence => Dr. Horrible's Sing-Along Blog is an excellent show.
Dr. Horrible's Sing-Along Blog is an excellent show.
Percentages. happy: 0.125 sad: 0.125
This is a neutral sentence
```

To accomplish this you will write a function called `percentage_happy(sentence)` which returns the percentage of words in a given string called `sentence` that are happy. To do this, find the total count of the following 5 words: `laugh happiness love excellent good` and divide this by the total number of words in the sentence. Here is an example run of this function:

```
>>> percentage_happy("I laughed and laughed at her excellent joke.")
0.375
```

This is because the count of happy words is 3 (laugh is repeated twice) and total number of words is 8. So, $3/8 = 0.375$. How do you find the number of words? You can count spaces and assume there is only one space between words for now. Your code should work even if there are upper and lower case words and extra spaces in the beginning and end of the sentence.

```
>>> percentage_happy("   Happiness is the state of a student who started homework early.
      ")
0.09090909090909091
```

Next, write a second function called `percentage_sad(sentence)` that works the same way but instead counts the percentage of the following 5 sad words in English: `bad sad terrible horrible problem` (there are sadder words for sure, but no reason to depress ourselves).

```
>>> percentage_sad("Dr. Horrible's Sing-Along Blog is an excellent show.")
0.125
>>> percentage_sad("Alexander and the Terrible, Horrible, No Good, Very Bad Day")
0.3
```

Of courses, there are more than 5 words of each category. We will see how to feed them using a file and use lists to process them in future classes.

Test your code well and when you are sure that it works, please submit it as a file named **hw2_part3.py** to Submitty for Part 3 of the homework.