CSCI 1100 — Computer Science 1
Fall 2016
Homework 1: Calculations and Strings

Overview

This homework, worth **75 points** total toward your overall homework grade, is due Thursday, September 15, 2016 at 11:59:59 pm. The three equally-weight parts should each be submitted separately. All parts should be submitted by the deadline or your program will be considered late.

General Comments: Read This Before Starting the Homework

Welcome to CS-1 homeworks. We are starting this assignment by outlining the course homework policies. These will not be repeated so please read this carefully and refer back to it for future assignments.

Submission of homeworks: test first, but you can submit many times

As you learned in Lab 1, all homework will be submitted electronically through *Submitty*, the Department of Computer Science homework submission and auto-grading server. This link is also available on the course website. The homework submission server for a particular assignment will typically be available no later than the Monday of the week the homework is due. So, you can expect that for this homework, you should be able to submit **by Monday, September 12, 2016**, perhaps sooner.

Make a habit of testing your program by running it using the Wing IDE. First, make sure that it can run. Then, go through the logic to make sure that it is correct. Learning to test your code is a big part of learning to program. Use Submitty, which can get slow near the deadline, for submitting and final changes rather than for testing. You will not be penalized for submitting a program multiple times, but you are generally graded on the last solution you submit. Note that the time you submit the last version determines whether a homework is late or not.

How will you be graded?

- Program correctness will be the most important determinant of your grade. In some cases, especially later in the semester, we will test your code with many different inputs, not just the ones we provide as sample test cases in the homework description.
- Small differences in output spacing will cause small losses of points. While we care mostly about correct logic, making sure the output matches the sample output exactly teaches you to use the `print` function and strings effectively.
- In addition to looking at output, we will also read your code. Make sure your program is well-written and has clear, correct program logic. Test it yourself with additional test cases, not only the ones we give you. You may lose points even if you match all test cases but your logic is unclear or incorrect.

- We will look at programming style including the organization of your code, the written comments describing the purpose of your code, the names of variables, the use of functions, etc. This will be emphasized increasingly as we progress through the semester. It is important to develop good habits even when writing relatively short programs.
- Remember that generally the last version you submitted is the one that will be graded. You can use Submitty to make a past submission the active one if you want us to grade that one instead, but you need to specify this explicitly on the server. This must be done by the submission deadline. It is your responsibility to manage it.
- We often receive the question, **can I use a programming construct that we have not learned yet?** While you may not lose points for doing so — unless we give you an explicit warning — we strongly urge you not to. We design homeworks to specifically target the concepts we are learning right now. Hence, while there may be more advanced methods to solve a problem, if you feel you need to use them you are not properly learning and applying the concepts we are teaching.

Late homework policy

When a homework has multiple parts, the submission time is the time that the last part is submitted. For example, if you submit parts 1 and 2 three days before the due date and part 3 one hour after the submission deadline, you will have used up one late day. Reread the late homework policy in the syllabus. You have a total of three full or part days you can use for late homeworks in the whole semester and up to two days can be used for a single homewor, assuming you have them available. It is highly recommended that you save these for later homeworks that will be harder and longer.

Wing IDE vs. Homework Submission server

On *rare* occasions when the homework submission server runs your code it will produce different output than the Wing IDE does, even though both are using Python 3.5.2. In such cases pay careful attention to the server's output and try to figure out what happened. Usually you've done something wrong in the way you wrote, submitted or executed your program. If you can't fix the problem, check the Piazza site for a relevant discussion. Only after checking should you post a Piazza question (and remember do not post your code!). This is also a good topic for help during office hours.

Input format for homeworks in this class

Your program must read the same number of inputs as are required according to the given problem. For example, if we tell you to read a name first and an email address next, that means your program must have two `input` statements. If it does not, you will get an error like:

```
EOFError:  EOF when reading a line
```

This means either you are trying to read too many or too few inputs. Read the problem specification carefully.

In all homeworks, we will use a convention specific to CS-1. If we ask you to read an input, then after you read it you must immediately print that input. For example, the following is the correct method to input the name string and the email address string:

```
name = input('Enter a name ==> ')
print(name)
email = input('Enter an email ==> ')
print(email)
```

The above program will produce a different output in Wing IDE and on Submitty, as shown below:

```
Enter a name ==> Rensselaer
Rensselaer
Enter an email ==> rpiinfo@rpi.edu
rpiinfo@rpi.edu
```

Wing IDE output
(what you see on your computer)

```
Enter a name ==> Rensselaer
Enter an email ==> rpiinfo@rpi.edu
```

Homework submission output
(what you see on submission server)

If you forgot to add the print function calls, you would actually see something like this in the submission server which will be considered incorrect output:

```
Please enter a name==> Please enter your email==>
```

The difference — and this is not really important for actually completing Homework 1 — is that for each part of the homework, we place all the input into a file and do what's called "running from the command-line." In the above example, we are using an input file (let's assume it is called `input.txt`) that contains the two strings `Rensselaer` and `rpiinfo@rpi.edu` on two lines.

When a program is run from a command shell, we use a command-line of the form:

```
python part1.py < input.txt
```

Unfortunately, the free version of the WingIDE that we have been using in class does not allow us to specify this input. But, you can do the command-line form on Mac/Linux with a Terminal window, and on Windows using Linux-derived Cygwin tools. If you want to learn how, you can ask us during office hours. Anyway, here is the bottom line:

Anytime you read input, just print it immediately afterwards to match the expected output.

The Actual Homework Description

With all of that as a backdrop, here is the actual homework assignment, in three parts.

Part 1: Astronomical Calculations

Jupiter, the largest planet in our solar system, has a diameter of 88,846 miles and its average distance from the Sun is about 483,632,000 miles. These are big numbers and as such they are a

bit hard to understand intuitively. One way to appreciate them is to compare them to the values for Earth, which has a diameter of 7,926 miles and a distance from the Sun of about 92,957,100 miles. The size can also be compared to the Sun, which has an average diameter of about 864,938 miles. Use 186,000 miles per second as the speed of light.

Write a Python program to calculate and output

1. the ratio of the Sun's radius to Jupiter's,
2. the ratio of the Sun's radius to Earth's,
3. the ratio of Jupiter's radius to Earth's,
4. the ratio of Jupiter's distance from the Sun to the Earth's distance from the Sun,
5. the ratio of Jupiter's volume to that of the Earth (assuming both are spheres),
6. the time in minutes it takes light to travel from the Sun to Earth (use $186,000$ miles per second as the speed of light), and
7. the time in minutes it takes light to travel from the Sun to Jupiter.

Your output must match

```
Sun-to-Jupiter radius ratio: <number>
Sun-to-Earth radius ratio: <number>
Jupiter-to-Earth radius ratio: <number>

Jupiter-to-Earth Sun distance ratio: <number>
Sun-to-Jupiter volume ratio: <number>
Sun-to-Earth volume ratio: <number>
Jupiter-to-Earth volume ratio: <number>

Sun to Earth light travel time in minutes: <number>
Sun to Jupiter light travel time in minutes: <number>
```

where in each case `<number>` is replaced by the floating point value your program calculates. **You must** use the value 3.14159 for π or your output will differ from ours. (We will see very soon how to get more accurate values of π.)

Your program must have the five distances and diameter values typed into the code each **exactly once** and then use variables after that. This would make it easy to change if, for example, we decided to use the values for Saturn instead. It would be even better if a string variable stored the name for Jupiter and then this name were used in the `print` statements.

Upload your Python program file to Submitty as Part 1 of HW 1.

Part 2: Speed Calculations

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. 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 and my average speed is 7.07 miles per hour.

Your job in Part 2 of this homework is to write a program that asks the user for the minutes, seconds and miles from an exercise event and outputs both the average pace and the average speed. Here is an example showing the expected output of your program:

```
Minutes ==> 53
Seconds ==> 30
Miles ==> 6.3

Pace is 8 minutes and 29 seconds per mile.
Speed is 7.065420560747664 miles per hour.
```

You can expect minutes and seconds to both be integers, but miles will be a float. The two outputs for the Pace must both be integers, so please use integer division and remainder 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. (Very soon we will learn methods for shortening the speed output, but we will not do so for this assignment.) Notice that our solution generates the blank line before the output of the calculations. When you have tested your code and are sure that it works, please submit it as Part 2 of Homework 1.

Part 3: Madlibs

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

```
Hello <proper name>,
  Good morning!  Are you looking forward to a/an <adjective> <noun>?
  You will <verb> a lot of <noun> and feel <emotion> when you do.
  If you do not, you will <verb> this <noun>.

  Did you watch the <noun> this <season>?
  Were you <emotion> when the <team-name> won?

  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 Madlib. (Be sure to reread the *Input format for homeworks in this class* discussion in the instructions above.) Make sure your output looks like the above paragraph, except that the missing information is filled in with the user input. Here is an example run of the program (how it will look at the homework submission server):

```
Let's play MadLibs for Homework 1
Type one word responses to the following:
```

```
proper_name ==> Alex
adjective ==> interesting
noun ==> semester
verb ==> write
noun ==> code
emotion ==> proud
verb ==> struggle
noun ==> semester
noun ==> Olympics
season ==> summer
emotion ==> excited
team-name ==> USA
adjective ==> nice

Here is your Mad Lib...

Hello Alex,
  Good morning!  Are you looking forward to a/an interesting semester?
  You will write a lot of code and feel proud when you do.
  If you do not, you will struggle this semester.

  Did you watch the Olympics this summer?
  Were you excited when the USA won?
  Have a/an nice day!
```

We've provided reasonable inputs, but the idea of MadLibs 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 MadLib 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** MadLib, figures out what to ask the user, asks the user, reads the input, and generates the final MadLib.

When you have tested your code and you are sure that it works, please submit it as the solution to Part 3 of the homework.