Computer Science 1 — CSCI 1100
Lab 1 — Variables, Assignments, Expressions and Submitty

Lab Overview

This lab has two goals: (1) to learn how to use *Submitty* the submission server we will use for submitting homeworks and checking grades, and (2) to practice writing short programs that involve variables, assignments, expressions and simple string operations. This will be accomplished by three small checkpoints. Throughout, we will also practice finding and fixing syntax and semantic errors, begin our study of good program structure and also learn what hard-coding is and how to avoid it.

Checkpoint 1: Introduction to Submitty

For each set of lecture exercises and for each homework assignment you will log into Submitty, upload your code/programs, run them, and view the results.

Go to the Class Materials section on Submitty and download the program `lab1_volume.py`. (Store this in a folder you create within your Dropbox folder structure for Lab 1.) Bring up the Spyder IDE and Open this file (click File and then Open and navigate to the desired file) so you can look at the program closely. Rather than making you write your own program for this first part, we will start with this "flawed" version just to demonstrate some important aspects of *Submitty*. Note that this file is intended to calculate the volume of a cone. If you are fuzzy about the correct formula, it is `1/3 * area_of_base * height`. I am going to assume you can calculate the area of the circular base!

This simple Python program contains **both** syntax and semantic errors. Don't fix these errors yet, even if you know what they are! If you don't remember the difference between a syntax error and a semantic error, look back at the Lecture 2 notes.

First, we want to familiarize you with the various errors messages you will get when you run your own program in the Spyder IDE and when you upload a program to Submitty that has errors. Please follow these steps:

(a) Look at `lab1_volume.py` in the IDE and, without making any changes, attempt to run it (click the green triangle/arrow). It will complain of syntax error(s). Do **NOT** fix these errors yet. Instead, upload this file as is to Submitty to see what happens.

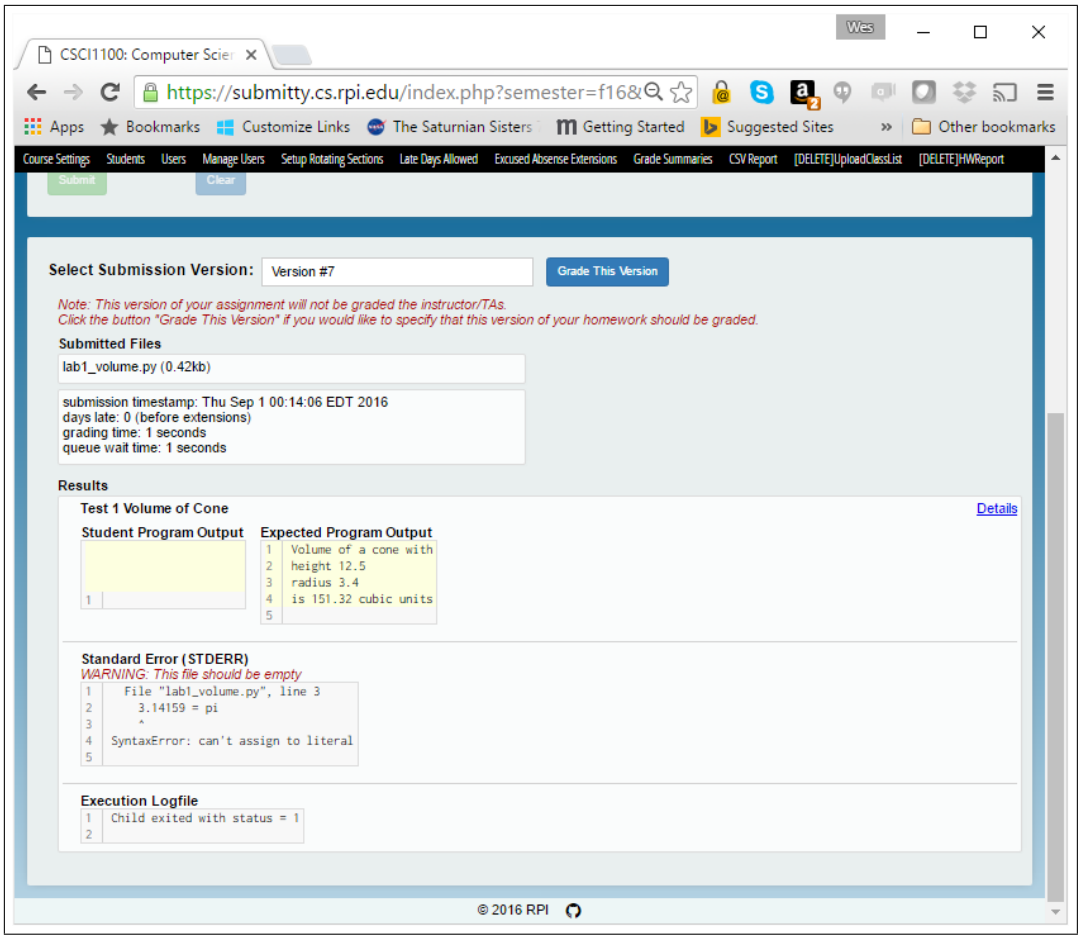
(b) To do this, cut-and-paste the following link to the submission page into a browser (or click below)

    `https://submitty.cs.rpi.edu/s25/csci1100`

    and login with your RCS id and password. Note, if your RCS id is abcde@rpi.edu, your login for Submitty will be abcde and your password will be your RCS password for that account. The submission server is also linked from the Course Website (`https://www.cs.rpi.edu/~gildem4/CS1100/index.html`) if you need to find it later.

(c) Once you are logged in, click on the `Submitty Practice` item. This will bring up a submission window. Either drag your `lab1_volume.py` file into the submission box, or click in the box and select your file from the file browser. Click **Submit** to start the grading process. Once the program runs, click on the **Details** text to the right of the

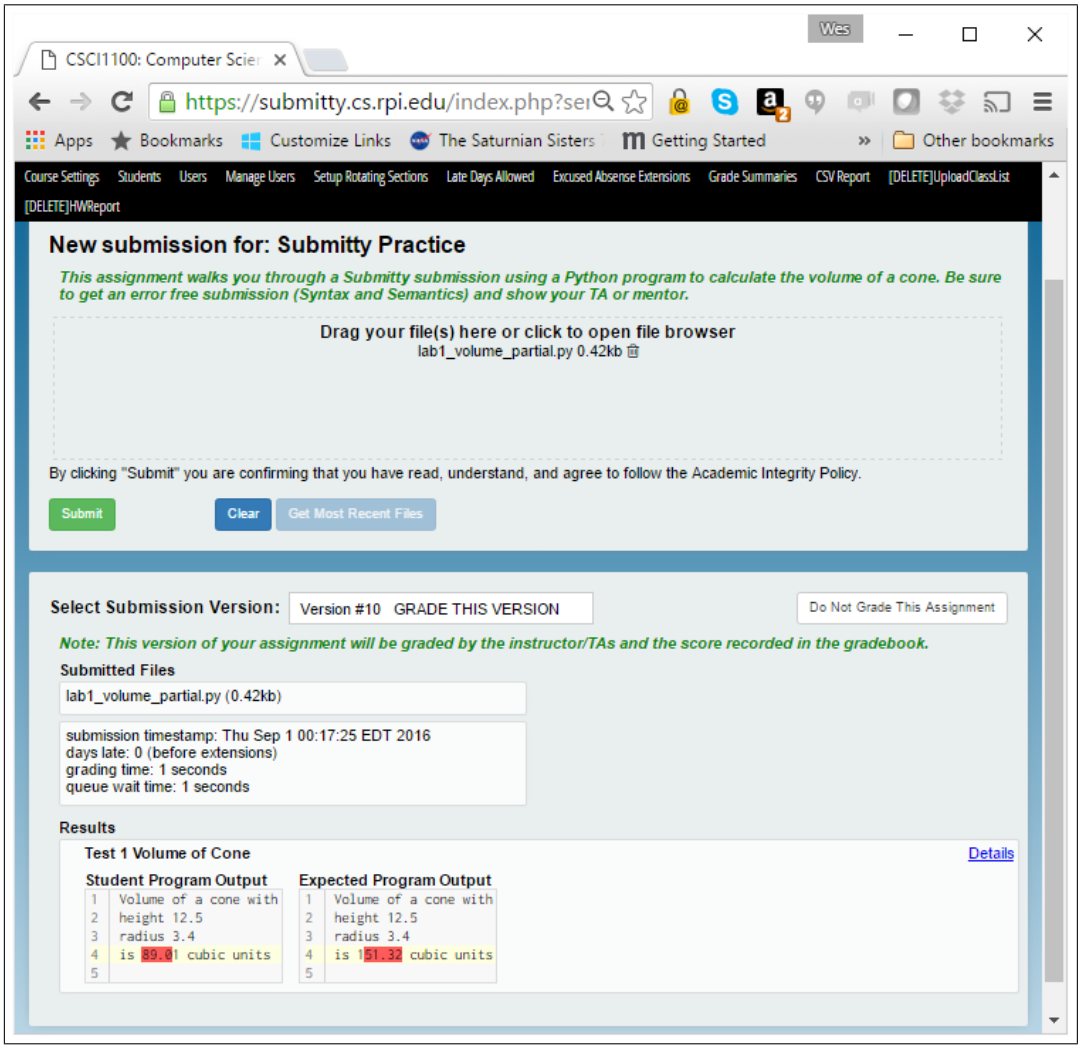
test to see the details of the `Submitty` run. Make sure that you do this every time you submit to make sure you catch any errors.



This is what the server shows you when your program has syntax errors. (Python only shows one syntax error at a time, so if you have multiple syntax errors in your program you will only see the first one.)

(d) Now it is time to find and fix the syntax error(s). It would be best if you do **not** fix any semantic error(s) you may find yet. We recommend that you edit the file in your Spyder IDE and use Spyder to get the program to run rather than using Submitty as a syntax checker. Keep correcting syntax errors and then checking the program by using the green arrow in Spyder until you have made sure that the program runs in the Spyder IDE.

(e) At this point, we want to demonstrate how Submitty shows semantic errors. Submit the syntactically-correct version of `lab1_volume.py` to the homework server by again loading your program into the submission block. Answer Y if asked if you want to replace the file. Once the file is loaded, hit **Submit**, and review the difference between your results and the expected results for any discrepancies. Here is an example



Places where your solution differs from the expected solution are shown in **red**

(f) Finally, fix the semantic errors in the program and resubmit. Repeat this as many times as you need to until your answer **exactly** matches ours.

To complete Checkpoint 1: Show a TA or mentor that you have successfully submitted both correct and incorrect versions of the program to Submitty.

Final note on the homework server: You have seen at this point that multiple submissions of the same assignment are accepted on the homework server. There is no limit to the number of times you may submit. (You may start getting a warning about the number of submissions after twenty tries, but this can be safely ignored.) The server keeps track of all submissions. Please be aware, however, that the TAs will only look at the **active version** when grading your homework, which is usually the latest version unless you designate an older version as the active version. You can explicitly change the active version by selecting the version from the dropdown next to "Select Submission Version" and clicking the "Grade this version" button.

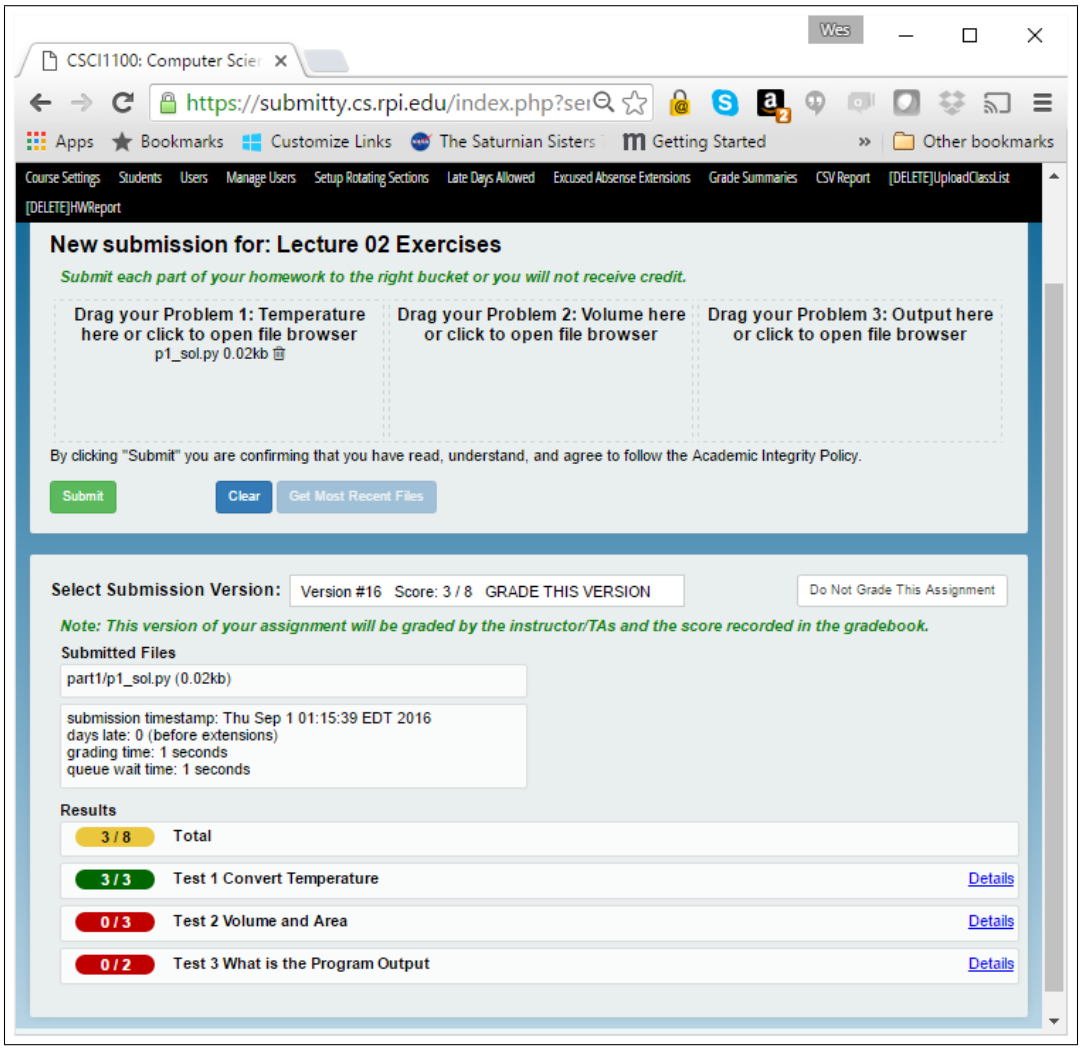
Also be aware that the server may run your program slower near deadlines, making you wait to see the results of your program. Make a habit of testing your program before you

submit to avoid unnecessary delays. Note that in terms of deadlines what matters is when you submitted the program, not when it showed you the output.

Checkpoint 2: Submitting Lecture Exercises

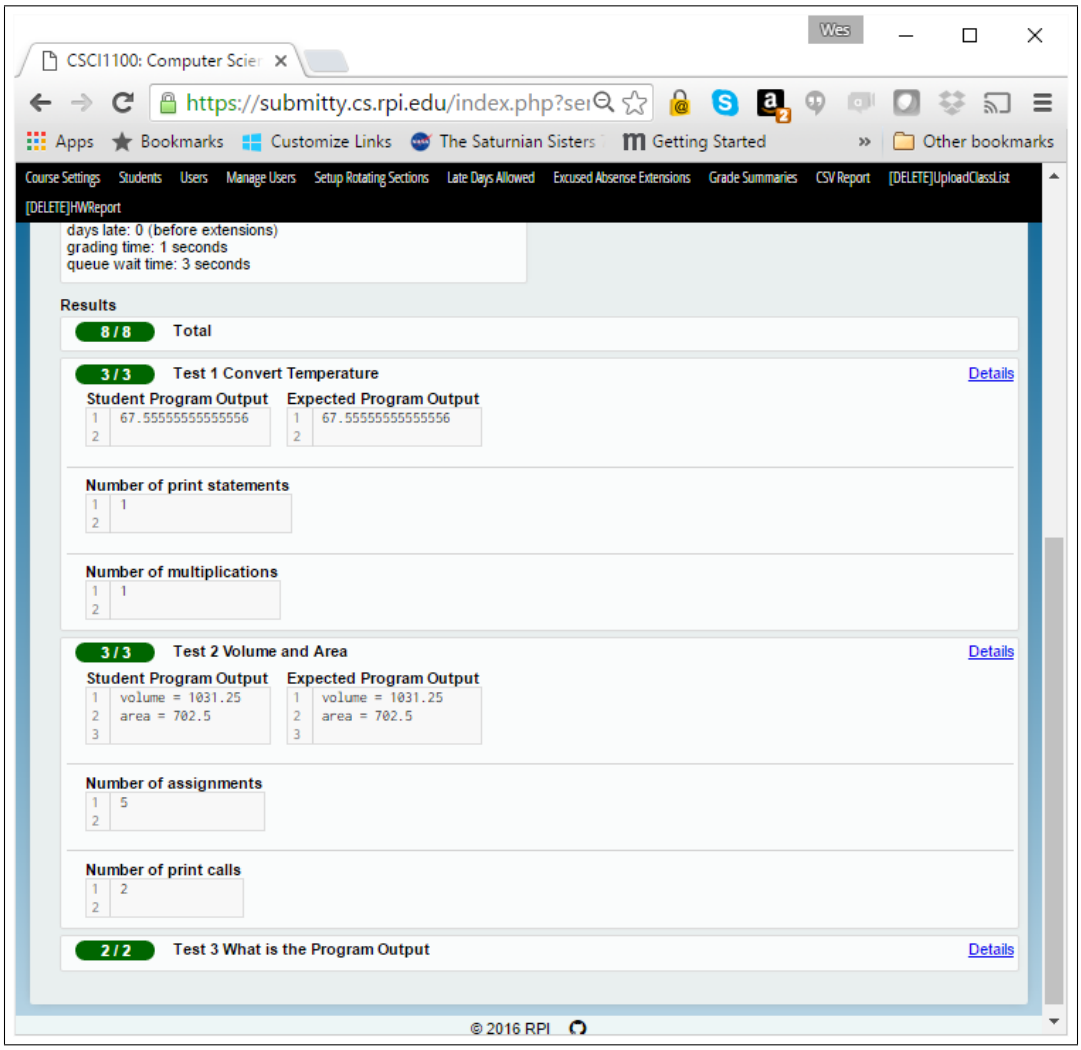
The Lecture 2 exercises asked you to write two short Python programs and to generate a file containing the output from a program. In this Checkpoint you will upload these three files to Submitty.

Start by navigating back to the CS 1 Submitty start page using the links at the top of the page. This time, select **Lecture 2**. Since Lecture 2 asked you to submit two Python programs and one text file, you will notice that there are three submission blocks on the page labeled for the three problems you were asked to solve. These three boxes work independently. You can upload solutions to any or all of the parts independently before running submit. For example, the following screen shows a submission with only the first question answered.



You can iteratively add the rest of the answers in, and you can change any or all of the solutions independently at any time. When you hit **Submit** the most recent versions of each file will be evaluated and scored. All lecture exercises will be automatically scored

without TA intervention. To make sure that you are following instructions, we have the ability to inspect the code. Clicking on the **Details** link shows what we are looking for in the answer as shown below.



For Lecture 2 we are counting print statements, multiplication operators, assignment statements and plus statements to be sure you are using the correct number based on the instructions. We are being very specific in our messages for this set of exercises; that may not always be the case. If you are having issues getting your tests to pass, **re-read the instructions** and make sure you are following all of the guidelines.

Additional details on using Submitty can be found at `http://submitty.org/student`. Expect them to be updated as the semester proceeds.

To complete Checkpoint 2: Show a TA or mentor that you have successfully submitted correct Lecture Exercise 2 to Submitty. You will get full credit for showing Lecture 2 submission only. However, it is recommended that you finish all lecture exercises upto and including Lecture 6 on the day of Lab 1.

Checkpoint 3: How big is your hard disk?

How much data can your hard drive store? This question is harder to answer than you think. The main measure of data in computers is a byte. For example, an integer in Python is 8 bytes.

In computer science, a gigabyte (GB) is a measure equivalent to $2^{10} * 2^{10} * 2^{10} = 2^{30}$ bytes. In fact, most operating systems use this convention. A terabyte (TB) is 1024 gigabytes, or 2^{40} bytes. We call this base 2 computation; everything is a power of 2.

However, computer manufacturers find this too complicated. So, instead they use base 10 computation. When they sell you a computer, a gigabyte is assumed to be $10^3 * 10^3 * 10^3 = 10^9$ bytes. A terabyte similarly is 10^3 gigabytes, i.e. $10^3 * 10^9 = 10^{12}$ bytes.

So, when you buy a 128 GB hard disk (manufacturer definition, base 10) and put it in your computer, your computer will record that it is actually 119 GB (computer definition, base 2). As a result, you have somehow lost 9 GB before even using your computer. Sad but true.

In this checkpoint, you will print out this size difference for an input value. To accomplish this, you are going to write a new Python program.

(a) Start by clicking `File -> New` in the IDE to create a new program file to type into.

(b) In this new program, prompt the user for a disk size in gigabytes and store the value in a a variable called `base10size`.

(c) Immediately print out the value entered. This is critical for properly showing the input value when your code is run in Submitty and we will be using this convention throughout the semester.

(d) Create a new variable called `base2size` and assign it the size of a disk with `base10size` "gigabytes" converted to base 2 "gigabytes". Do this by multiplying `base10size` by 10^9 (a base 10 gigabyte) and then dividing by 2^{30} (a base 2 gigabyte).

(e) Create a variable called `lost_size` that stores the difference between the two gigabyte sizes.

(f) Write a print statement to output the result in the example format using your variables. (See the examples below for expected input and output when using the Spyder IDE.)

(g) Finally, print two strings to show the difference. The first string will have `base10size` stars and the second line will have `base2size` stars. Be careful, there is an extra space after `Input:`. When checking your output, Submitty will look for the exact string.

Here are two potential expected runs of this program, one with input `32` and the other with input `64`.

```
Disk size in GB => 32
32
32 GB in base 10 is actually 29 GB in base 2, 3 GB less than advertised.
Input:  ********************************
Actual: ****************************
```

```
Disk size in GB => 64
64
64 GB in base 10 is actually 59 GB in base 2, 5 GB less than advertised.
Input:  ************************************************************
Actual: ************************************************************
```

Now, save this (very short) program to a file called `part3.py` in your Lab 1 Dropbox folder. Run it to make sure it is correct.

To complete Checkpoint 3: When you are convinced your program is correct, show it to a TA or a mentor. Your program should have only three variables. Optionally, you may also submit it to the **Lab 01 - Disk Size** gradeable on Submitty.