# Computer Science 1 — CSci 1100
# Lab 1 — Variables, Assignments, Expressions and Submitty
# Fall Semester 2016

## 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 and expresssions. 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. For checkpoint one we will concentrate on submitting a single python file in order to get a correctly running program with no syntactic or semantic errors.

Go to the resources page from our Piazza site 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 Wing 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 Wing IDE and when you upload a program to Submitty that has errors. Please follow these steps:

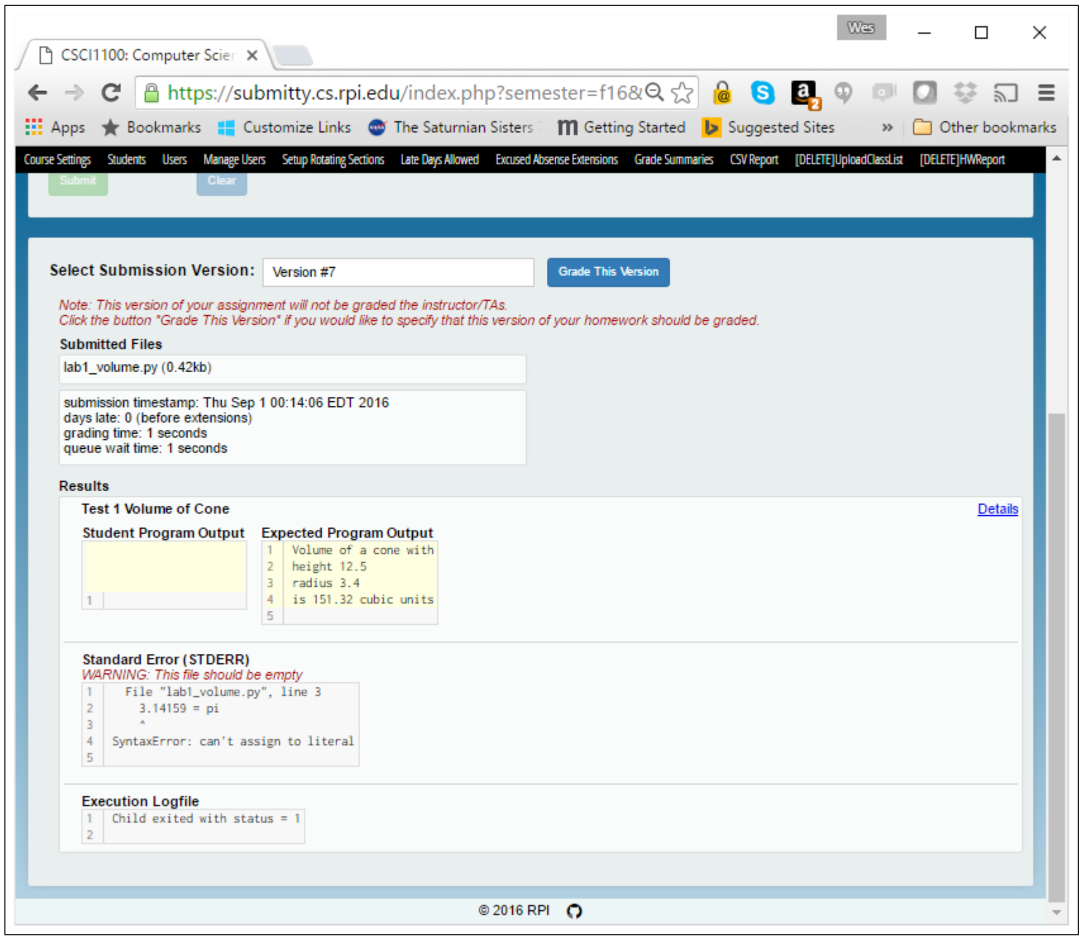
(a) Look at `lab1_volume.py` in the IDE and, without making any changes, attempt to run it (click the green triangle/arrow). It will complain of syntax error(s). Do **NOT** fix these errors yet. Instead, upload this file as is to Submitty to see what happens.

(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/index.php?semester=f16&course=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 Piazza site under Resources 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.



Select Submission Version: Version #7

*Note: This version of your assignment will not be graded the instructor/TAs.*
*Click the button "Grade This Version" if you would like to specify that this version of your homework should be graded.*

**Submitted Files**

lab1_volume.py (0.42kb)

```
submission timestamp: Thu Sep 1 00:14:06 EDT 2016
days late: 0 (before extensions)
grading time: 1 seconds
queue wait time: 1 seconds
```

**Results**

**Test 1 Volume of Cone**

**Student Program Output**

```

```

**Expected Program Output**

```
Volume of a cone with
height 12.5
radius 3.4
is 151.32 cubic units

```

**Standard Error (STDERR)**

*WARNING: This file should be empty*

```
  File "lab1_volume.py", line 3
    3.14159 = pi
    ^
SyntaxError: can't assign to literal

```

**Execution Logfile**

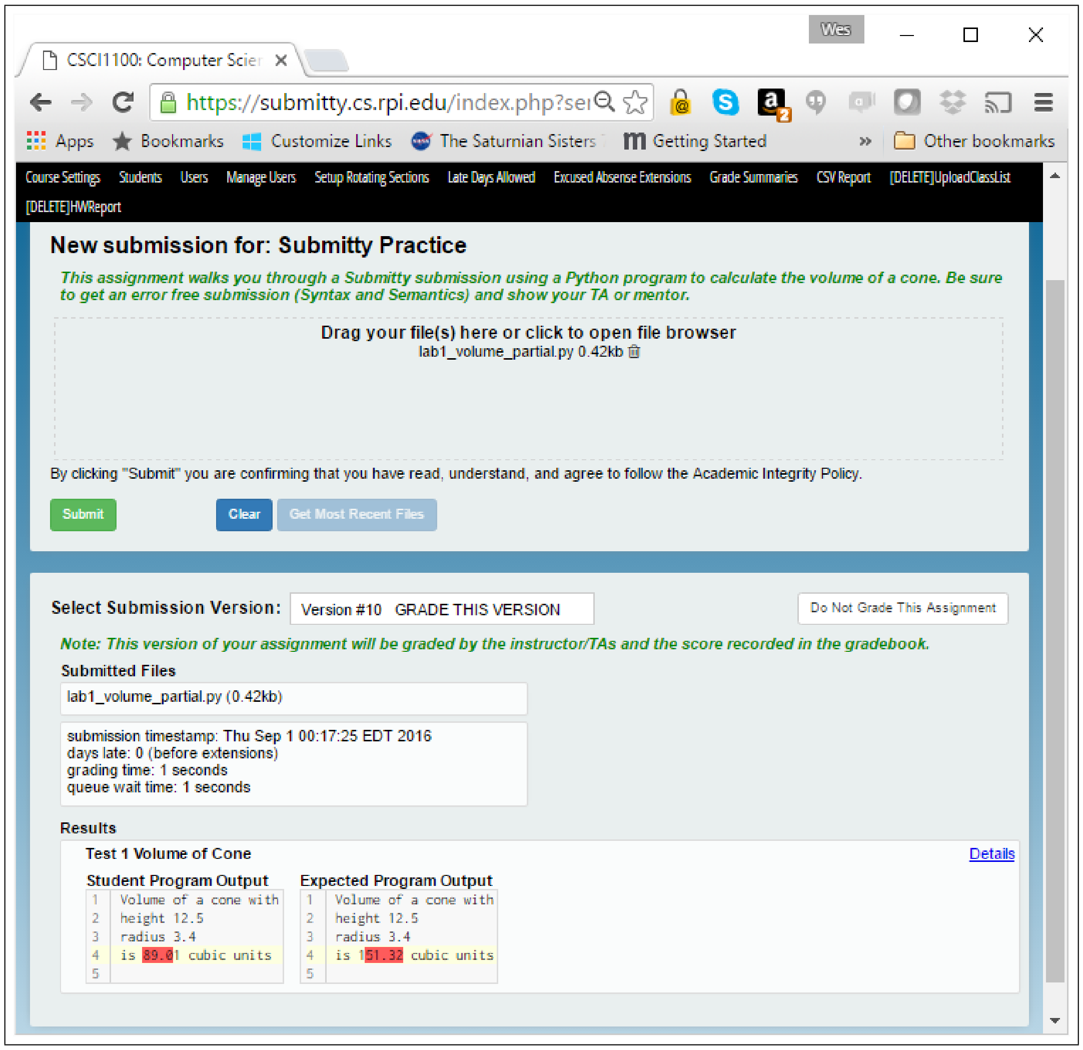
```
Child exited with status = 1

```

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 Wing IDE and use Wing 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 Wing until you have made sure that the program runs on Wing 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



**New submission for: Submitty Practice**

*This assignment walks you through a Submitty submission using a Python program to calculate the volume of a cone. Be sure to get an error free submission (Syntax and Semantics) and show your TA or mentor.*

Drag your file(s) here or click to open file browser
lab1_volume_partial.py 0.42kb

By clicking "Submit" you are confirming that you have read, understand, and agree to follow the Academic Integrity Policy.

Select Submission Version: Version #10 GRADE THIS VERSION

*Note: This version of your assignment will be graded by the instructor/TAs and the score recorded in the gradebook.*

Submitted Files

lab1_volume_partial.py (0.42kb)

```
submission timestamp: Thu Sep 1 00:17:25 EDT 2016
days late: 0 (before extensions)
grading time: 1 seconds
queue wait time: 1 seconds
```

Results

Test 1 Volume of Cone

Student Program Output

```
Volume of a cone with
height 12.5
radius 3.4
is 89.01 cubic units

```

Expected Program Output

```
Volume of a cone with
height 12.5
radius 3.4
is 151.32 cubic units

```

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 a mentor that you have successfully submitted both correct and incorrect versions of the program to Submitty.

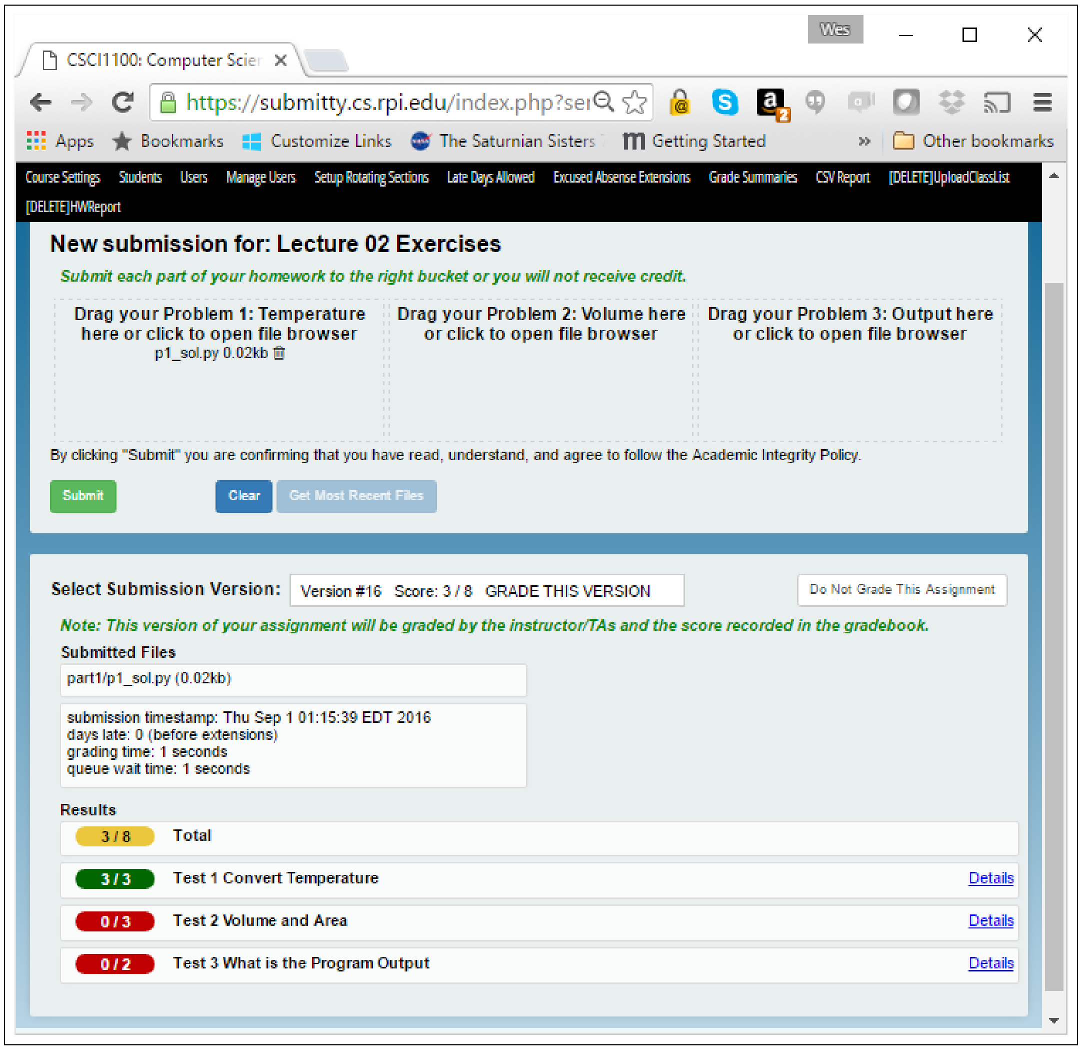
**Final note on the homework server:** You have seen at this point that multiple submissions of the same assignment are accepted on the homework server. There is no limit to the number of times you may submit. The server keeps track of all submissions. Please be aware, however, that the TAs will only 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.

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 CS1100 start page using the links at the top of the page. This time, select **Lecture 02 Exercises**. Since Lecture 02 asked you to submit 2 python programs and 1 text file, you will notice that there are 3 submission blocks on the page labeled for the 3 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.



New submission for: Lecture 02 Exercises

*Submit each part of your homework to the right bucket or you will not receive credit.*

Drag your Problem 1: Temperature here or click to open file browser
p1_sol.py 0.02kb

Drag your Problem 2: Volume here or click to open file browser

Drag your Problem 3: Output here or click to open file browser

By clicking "Submit" you are confirming that you have read, understand, and agree to follow the Academic Integrity Policy.

Select Submission Version: Version #16 Score: 3 / 8 GRADE THIS VERSION

*Note: This version of your assignment will be graded by the instructor/TAs and the score recorded in the gradebook.*

Submitted Files

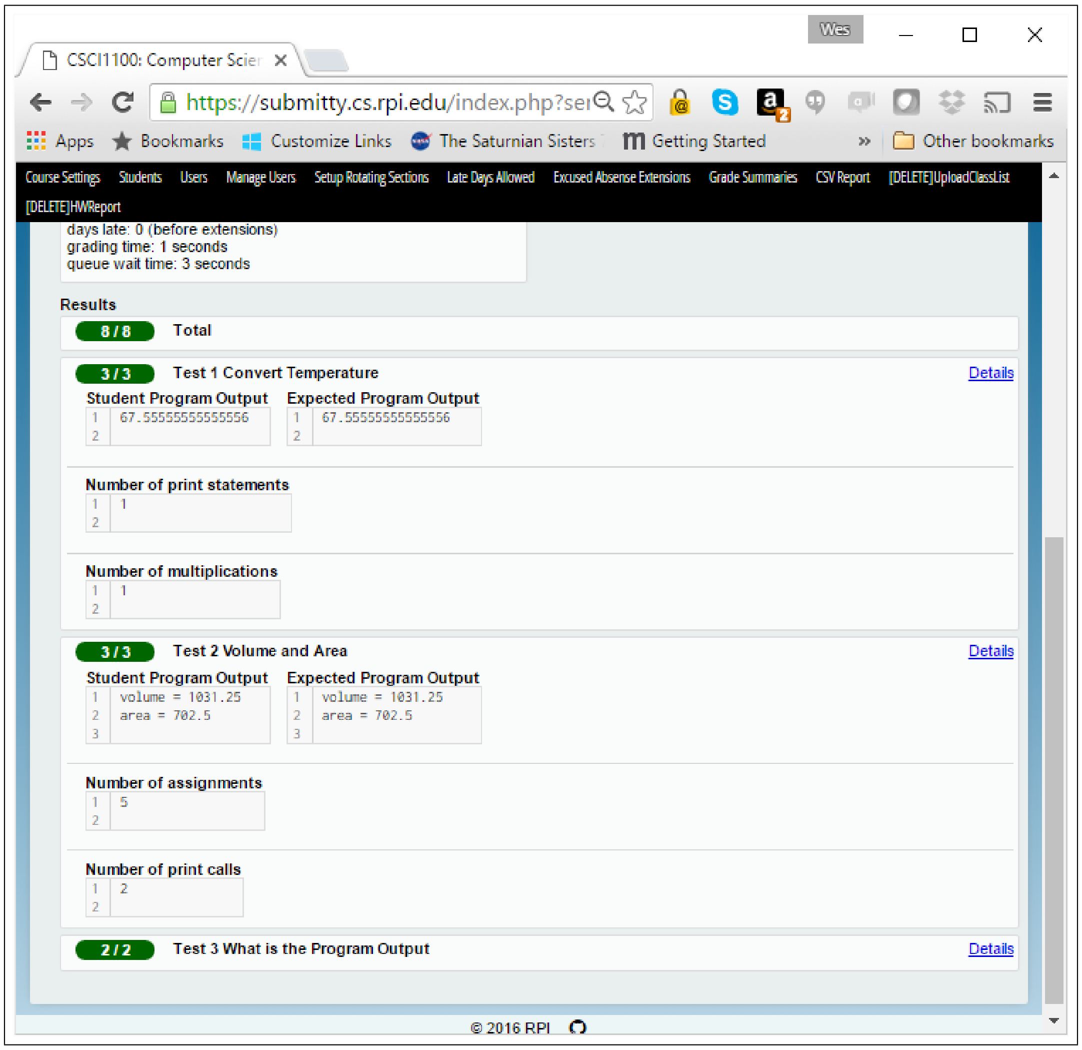
part1/p1_sol.py (0.02kb)

```
submission timestamp: Thu Sep 1 01:15:39 EDT 2016
days late: 0 (before extensions)
grading time: 1 seconds
queue wait time: 1 seconds
```

Results

| | | |
|---|---|---|
| 3 / 8 | Total | |
| 3 / 3 | Test 1 Convert Temperature | Details |
| 0 / 3 | Test 2 Volume and Area | Details |
| 0 / 2 | Test 3 What is the Program Output | Details |

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.


CSCI1100: Computer Scie
https://submitty.cs.rpi.edu/index.php?se
Course Settings  Students  Users  Manage Users  Setup Rotating Sections  Late Days Allowed  Excused Absense Extensions  Grade Summaries  CSV Report  [DELETE]UploadClassList  [DELETE]HWReport
Details

days late: 0 (before extensions)
grading time: 1 seconds
queue wait time: 3 seconds

Results

8 / 8 Total

3 / 3 Test 1 Convert Temperature

Student Program Output

```
67.55555555555556
```

Expected Program Output

```
67.55555555555556
```

Number of print statements

```
1
```

Number of multiplications

```
1
```

3 / 3 Test 2 Volume and Area

Student Program Output

```
volume = 1031.25
area = 702.5
```

Expected Program Output

```
volume = 1031.25
area = 702.5
```

Number of assignments

```
5
```

Number of print calls

```
2
```

2 / 2 Test 3 What is the Program Output

© 2016 RPI

For Lecture 2 we are counting print statements, multiplication operators and assignment 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 `https://github.com/Submitty/Submitty/wiki/STUDENT-SUBMITTER`. Expect them to be updated as the semester proceeds.

**To complete Checkpoint 2:** show a TA or a mentor that you have successfully submitted correct Lecture 2 Exercises to Submitty.

## 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 write a program that prints out this size difference. 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 to type into.

(b) In this new program, create a variable called `base10size` and assign 128 to this variable to represent the number of base 10 “gigabytes”.

(c) Create a new variable called `base2size` and assign it the size of 128 base 10 “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).

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

(e) Finally, write a print statement to output the result in the following format using your variables:

```
128 GB in base 10 is actually 119 GB in base 2, 9 GB less than advertised.
```

Now, save this (very short) program to a file called `lab1_part3.py` in your Lab 1 dropbox folder. Run it to make sure it is correct. Once it is complete, reuse your code to print the same information for hard disk sizes of 256 GB, 512 GB and 1024 GB as well. If you did this right, all you have to do is copy all your code, and change only the value of the base 10 variable. It looks repetitive right? We will talk about that in class in a few weeks!

**No hard-coding:** You must use three variables for this solution, with no hard coding of values other than the initial assignment to the variable `base10size`. What is hard-coding? It is writing the solution directly into a program. In these early exercises and homeworks, the problems are easy so we may know the answer is before we start. We use these easy problems to begin to learn the basics so that we can solve harder problems. As an example, the following print function all is using hard-coding:

```
base10 = 128
print("128 GB hard disk")
```

The following print function call is not using hard-coding:

```
base10 = 128
print(base10, "GB hard disk")
```

Of course, we need to hard-code the value for the initial variable for now until we learn how to write code that asks the user to type in values.

**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 and should not use hard-code values.