CMSC 201 Fall 2015
Lab 02 – Debugging

Assignment: Lab 02 – Debugging
Due Date: Friday, September 11th, 2015 by 8:59:59 PM
Value: 1% of final grade

In Lab 1, we logged onto GL and set up folders for 201 in your home directory. We also created a simple python program, and turned it in using the submit command. We'll be putting all of these skills to use in this lab.

As we've discussed in class, testing and debugging your programs is a large part of being a successful programmer. Sometimes, you may even have to debug other people's code!

In this lab, we'll be creating two files: **`lab2.py`** will be a file you create, and **`errors.py`** will be a file you copy into your directory, before finding and fixing the errors it contains. Both files will be submitted as part of the grade for Lab 2.

Finally, in-person labs with your TA start next week! At the end of Lab 2, you'll watch a short video to prepare you for using GL and emacs on the computers in the lab classrooms.

NOTE: You must watch all six of the videos for Lab 2, as they each contain important steps not covered in this document.

Part 1: Creating a complete Python program

We'll learn how to copy files into your account from an instructor's directory, and you'll write your first complete Python program. We'll also learn a few more useful emacs shortcuts.

Step 1:

Watch this video (https://www.youtube.com/watch?v=B5JpTHS_O88) and follow its instructions to complete the remainder of this part.

Step 2:

Copy the **`.emacs`** file into your home folder using the **`cp`** command (**`cp`** simply stands for "copy").

The **`.emacs`** file is used to customize the way the emacs program behaves. The file you just downloaded does things like allow you to use the mouse's scroll wheel when you are using emacs on the lab computers next week.

The period "**`.`**" in front of the file name indicates the file is a hidden file. If you simply type ls, you won't see it listed. To double check that you successfully copied the file, you need to use the command **`ls -a`**. The **`-a`** means "all" and will show all the files in that directory, even hidden ones.

Step 3:

Copy the **`lab2.py`** code from the video into a **`lab2.py`** file in your **`lab2`** folder.

*(HINT: You first need to create the **`lab2`** folder using the **`mkdir`** command, and the folder needs to be inside your **`Labs`** folder. For a reminder of how to create and navigate folders, refer to the instructions for Lab 1.)*

Step 4:

Make sure to update the information in the code at the top of the file to be your correct information. (Your name, date, lab section, and email. Also change "File" to **`lab2.py`**, rather than **`greeting.py`**.)

The pound symbols "**`#`**" you used in the **`lab2.py`** file are used to tell Python that any text after the pound symbol on that line is a *comment*. Comments are ignored by Python, and the text following a pound symbol does not need to follow any of Python's syntax rules.

Programmers use comments to explain what the code is doing, to leave notes to themselves, and to document things about the code. For example, the comments at the top of the file are called a "header block," and record who created the file, when, and what the file is supposed to do.

Step 5:

Learn and practice using a few basic emacs commands. Make sure to use **`CTRL+K`** to remove the third **`print()`** line, as seen in the video. Below, find a reference sheet of the emacs commands covered so far.

Command	Meaning
`CTRL+A`	Go to the beginning of the current
`CTRL+E`	Go to the end of the current line
`CTRL+K`	Remove everything on the line after the cursor ("kill")
`CTRL+Y`	Paste the line cut by the **`CTRL+K`** command ("yank")
`CTRL+X, CTRL+S`	Save the file and **stay** in emacs
`CTRL+X, CTRL+C`	Save the files and **exit** from emacs

Not covered in the video, but very useful commands for the terminal	
`"TAB" in terminal`	Hitting the tab key will autocomplete based on the available filenames. For example, typing "**`emacs la`**" and hitting tab will autocomplete "**`la`**" to "**`lab2.py`**"
`"up arrow" in terminal`	Hitting the up arrow will recall your previous command to the terminal. Hitting it again will pull up the command before that one; repeat as necessary.

Part 2: Finding and Fixing Code Errors

In this part of the lab, you'll be working on a Python file full of errors. We'll first explain how to find them, and how to understand the error messages. Following that, you'll solve the errors on your own.

Step 6:

Watch this video (https://www.youtube.com/watch?v=oHJESRqqtKQ) and follow its instructions to complete the remainder of this part.

Step 7:

Pay attention to the video as it explains how to read the messages Python prints out when it finds an error (also known as a "bug"). You will need to be able to do this yourself in the next step.

Step 8:

Copy the **`errors.py`** file into your **`lab2`** folder using the **`cp`** command. As instructed in the video, read the code and figure out what the program should be doing. Then use your new knowledge of finding and fixing bugs to create an **`errors.py`** file that runs without any errors.

(HINT: Remember, if there is more than one error, start with the message at the bottom first. Fix that one error, and then try to run the program again.)

Step 9:

Once errors.py works, add two lines to the file's header comment block.

```
# Fixed by:    YOUR_NAME
# Date fixed: TODAYS_DATE
```

(Optional Step):

Make a copy of the **`errors.py`** file called **`test_errors.py`** and play around with it to better understand what causes specific errors, and how Python will react. For example, what happens when you remove an equal sign? What happens when you add extra parentheses? What happens when you remove the parentheses from the line that starts with the code "**`average =`**"? (Is that a syntax error or a logic error?)

Part 3: Submitting Both of Your Files

In this part of the lab, you'll be submitting both of your Lab 2 files.
(If you completed the previous optional step, do not submit the ***`test_errors.py`*** *file.)*

Step 10:

Watch this video (https://www.youtube.com/watch?v=2EeteWBmTm8) and follow its instructions to complete the remainder of this part.

Step 11:

Submit the **`lab2.py`** and **`errors.py`** files using the **`submit`** command. Make sure to submit both files. Double check that you have successfully submitted the files by using the **`submitls`** command.

Part 4: How to Get Help

In this part of the lab, you'll be learning about the various ways that you can receive help if you are struggling with concepts or assignments in CMSC 201.

Step 12:

Watch this video (https://www.youtube.com/watch?v=8ng1RhQbrtQ) and follow its instructions to complete the remainder of this part.

(PROTIP: If you are having problems logging into GL, resetting your UMBC password and trying again has solved the problem for some students.)

If you are having account-based issues and need to submit a ticket with DoIT, start by going to my.umbc.edu. Click "Help" on the menu bar. On the Help page, select the category that best describes your problem. ("Account & Password" under the "Computing & Technology" category is a likely choice.) Fill out the form and click "Submit" at the bottom.

Part 5: Grading for Lab 2

In this part of the lab, we'll be covering how Lab 2 will be graded.

Step 13:

Watch this video (https://www.youtube.com/watch?v=_GD5gs6Z0aA) and follow its instructions to complete the remainder of this part.

Step 14:

Test your fixed **`errors.py`** Python program with different inputs to ensure that is runs correctly. Make sure you enable Python version 3 before testing:

```
/usr/bin/scl enable python33 bash
```

If your testing finds a bug, fix it, and make sure to submit the new version.

(HINT: For different inputs, try big numbers and negative numbers. Remember than an integer is a whole number, so your program won't work if you give it decimals or letters.)

(REMINDER: The headers of your files (the block of comments at the top) are very important. Double check that you have followed Steps 4 and 9 to correctly complete the headers for the **`errors.py`** *and* **`lab2.py`** *files.)*

Part 6: Prepping for Next Week

In this part of the lab, you'll learn how to use the lab computers.

Step 15:

Watch this video (https://www.youtube.com/watch?v=_a9rrgGcJco) carefully to see a demonstration of using GL and emacs on the lab computers. This will prepare you for next week, when in-person labs will begin.