Our test team in our company uses Linux shell scripts that perform stress and performance tests on our hardware and the communication to our back end. These scripts are copied and modified for specific device MAC addresses as we set up tests. This project will begin to automate the process of running the test scripts, monitor the progress, capture the data, and parse the results. The python program I developed is called autoscript.py. and was developed in Python 3.6 version on both Windows and Linux (Virtual Machine) systems.

This specific project scope has been automating the test scripts, monitoring the data and capturing the results. There are some improvements and features that can be added later to expand the usage of these items. For instance, the scripts are currently called without variable inputs. This is a feature to upgrade in future versions. Capturing the server logs from our syslog service and parsing those results are in future scope but would be the other end of the chain. Initiating the scripts are one end of the task but we heavily use the syslog output to trace issues. This would be a significant feature add, but is yet a future feature.

Time: approximately 15 hours

4 hours demo of file opening and command execution on a Linux Centos 7virtual machine, fine tuning of code.

11 hours developed on work Windows 7 system, then transferred via GitHub to Linux target for proofing of code.

Splitting development between Windows and Linux included some challenges. Linux/Posix command line calls using the ‘os’ module behaved slightly differently. For instance, I was seeing success on the Windows code and observing errors with the terminal seeming to hang when trying the program on Linux. I had to add the ampersand to the command string if running on Linux to fix this. The error was not obvious. Also, the tk windows look and feel between the systems caused me to modify the GUI until both looked decent when running. Also, Linux required running the program as Super User since I was getting tkinter errors when running as standard user. Run as ‘python autoscript.py’ in a Python 3.6 environment.

I used an example of tkinter program where it is created as a Class and run at the end of the file. This seemed to encapsulate the GUI and methods well. The example was from a book called ‘Starting Out with Python, 2nd Edition’ by Tony Gaddis from Pearson. Many nice examples of GUI planning and execution. Text boxes were not discussed in that book, however. That item and many others such as file dialogs and command execution were helped by doses of Google searches and StackOverflow results.

If I had more time, and what I will be continuing at work, is adding variable inputs and the parsing as described above. My manager is onboard with this Python automation and I can allocate time at work to continue improving what I have started with this project.

**Code organization:**

* Container Class
  + GUI Widgets
    - Buttons
      * Add file, Remove File, Run test, Quit
    - Labels
      * Some used as updating data
    - Text box
      * Added files shown here
  + Methods
    - add\_test
      * search for scripts to run
      * add to text box
      * add to script list
      * update label for Tests to be Run
    - remove\_test
      * can pop items off script list
      * updates text box and Run labels
    - run\_test
      * create directory for results
      * run each test as looping through script list
        + detect Linux or not for formatting on execution
      * update label for Tests that have been Run
    - update\_text\_box
      * This was separated from the other methods so the code was not repeated. Called form other methods.

Below is a screenshot of the program in action on our Linux Centos 7 Virtual Machine:

