

**Name: Mohamed Wagih Ahmed**

**ID: 14p6080**

**CSE423 Software Performance Evaluation Term Project**

In this document I’ll explain what my code does and how it works.

**Environment**: you need Python 3.7.1, Anaconda to run the Jupyter notebook and some other libraries installed like the ones below.

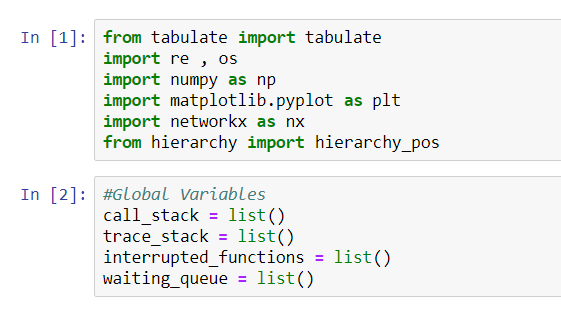
**Or you can simply open the .html file to only view the code and check the results without running the code (recommended).**

**Libraries and Global variables**

**tabulate library:** Prints the table view

**matplotlib:** Bar chart visualization

**networkx and hierarchy:** tree graph visualization, hierarchy is an open source piece of code. It is a workaround the networkx library as it doesn’t support hierarchal trees directly.

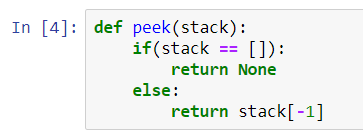


**Functions**

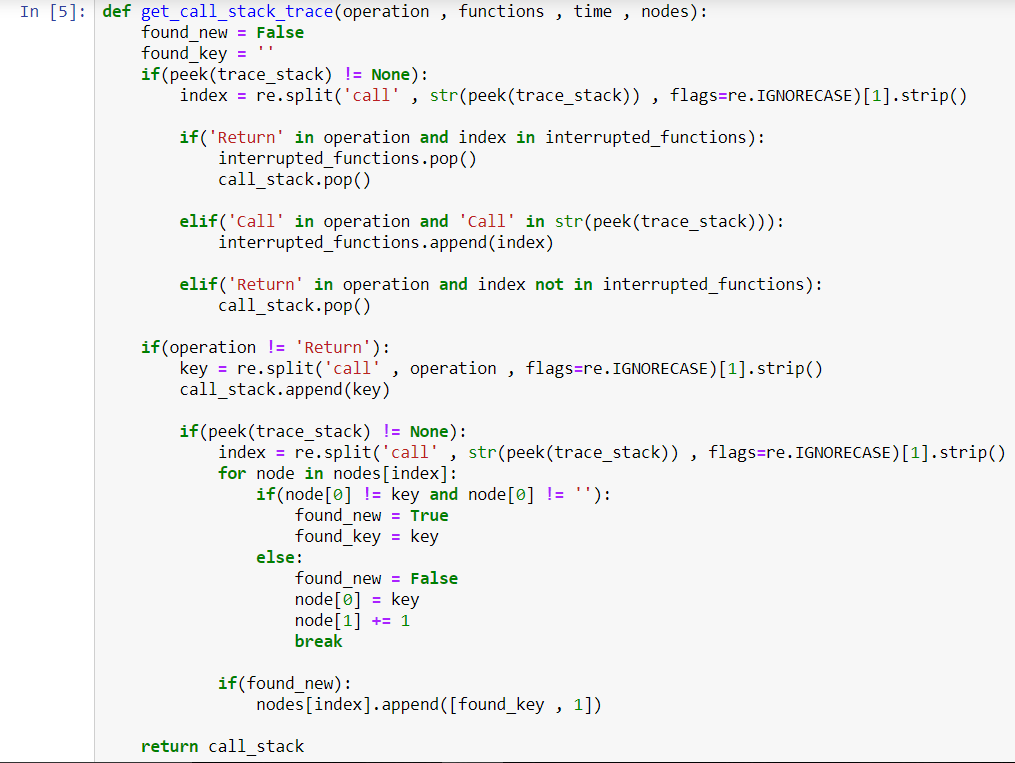
1. **show\_stack\_content(…):** Prints the stack’s content over time in a tabular form.



1. **peek(…):** Returns the top of the stack.



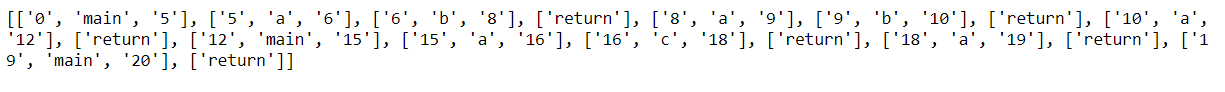
1. **get\_call\_stack\_trace(…)**: Responsible for showing the stack’s content over time and to fill the call context tree nodes values.



1. **read\_log\_file(…):** Responsible for reading the logs text file content, write the code\_flow variable’s content and to check if the entire program recorded in the logs file ended correctly or not.



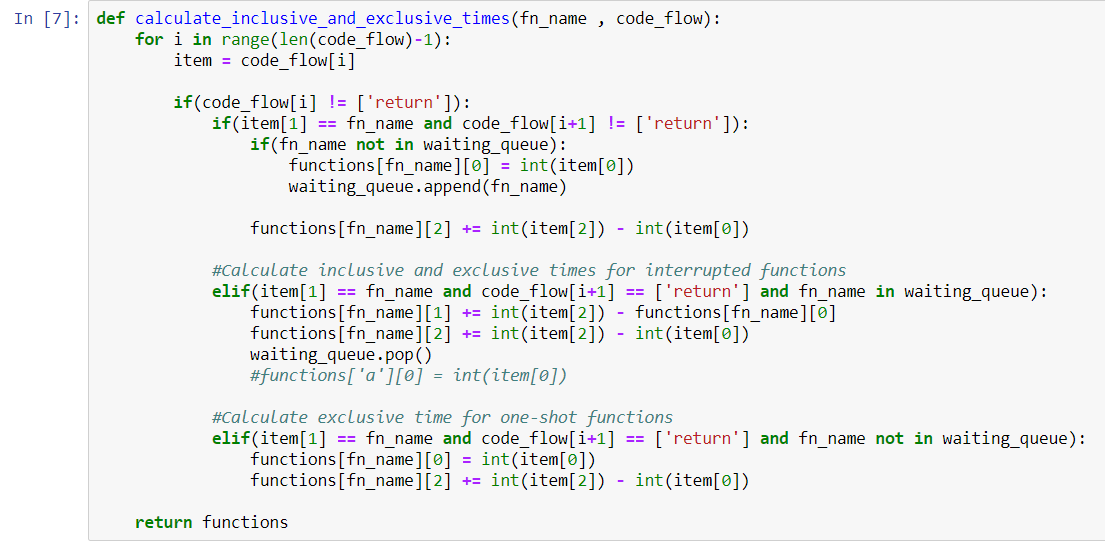
The code\_flow variable content looks like this after reading the logs file:



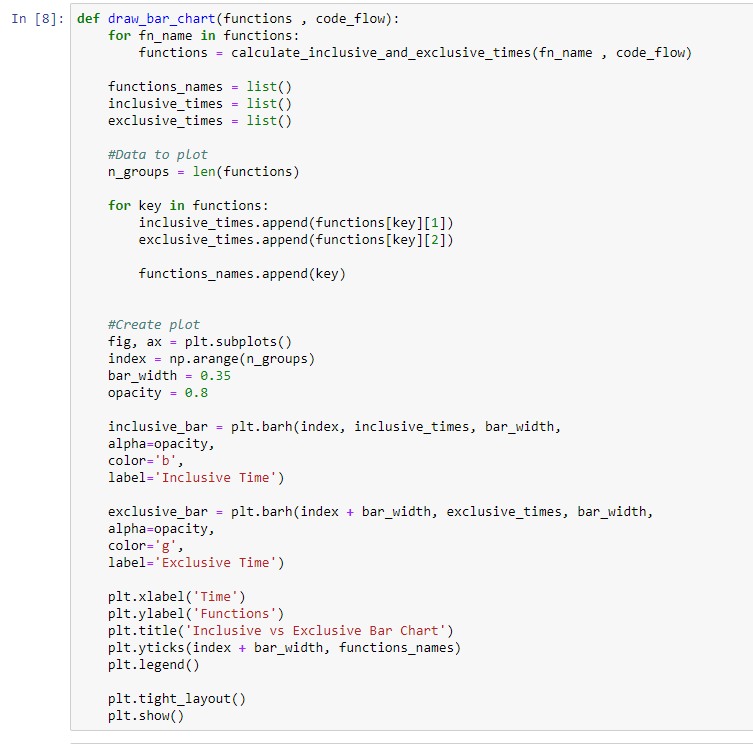
The code\_flow variable is any array of arrays of strings.

Each array is an instruction which has a start time, function name/return and a finish time.

1. **calculate\_inclusive\_and\_exclusive\_times(…):** Responsible for calculating the inclusive and the exclusive time of each functions, This was the toughest part in the project.  
   The trick behind this function is I simulated how each function will work in the OS by adding each interrupted function in waiting\_queue and by reading the code\_flow variable values I can add which function should be added to the queue for example if the main called function a that mean the main got interrupted by function a as in the code\_flow there was no return statement after the main was called initially and so on..



1. **draw\_bar\_chart(…)**: Responsible for the visualization of the results of the inclusive and exclusive times of each function in a bar chart.

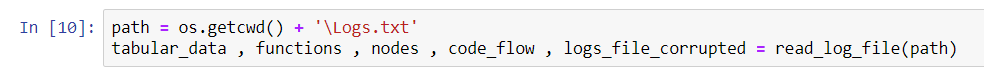


1. **draw\_CCT(…):** Responsible for the visualization of the Call Context Tree

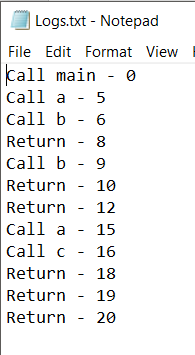


**Logs File**

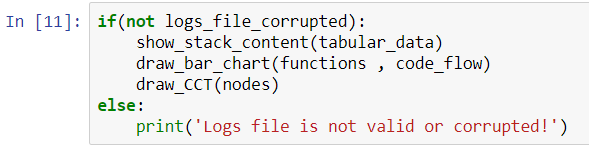
1. The path variable gets the current working directory and by changing the name of the logs file you can change the file being read.  
   Note that the logs file has to be in the same directory of the notebook or the python file.

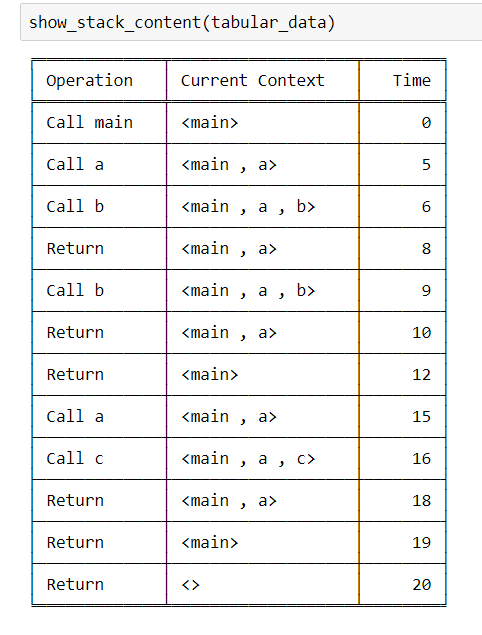
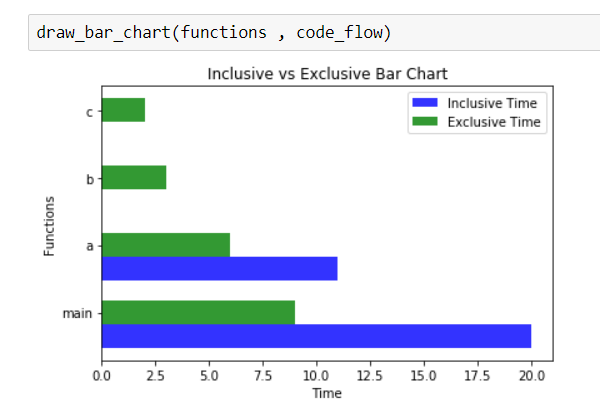
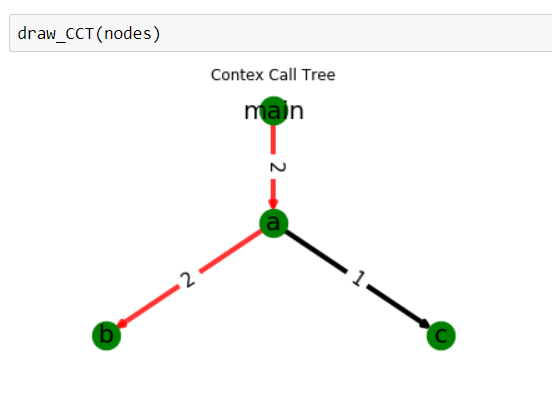


1. The Logs file should be a .txt file and the format is like the following:   
   **FUNCTION\_NAME WHITE\_SPACE DASH WHITE\_SPACE TIME\_STAMP**



**Results**

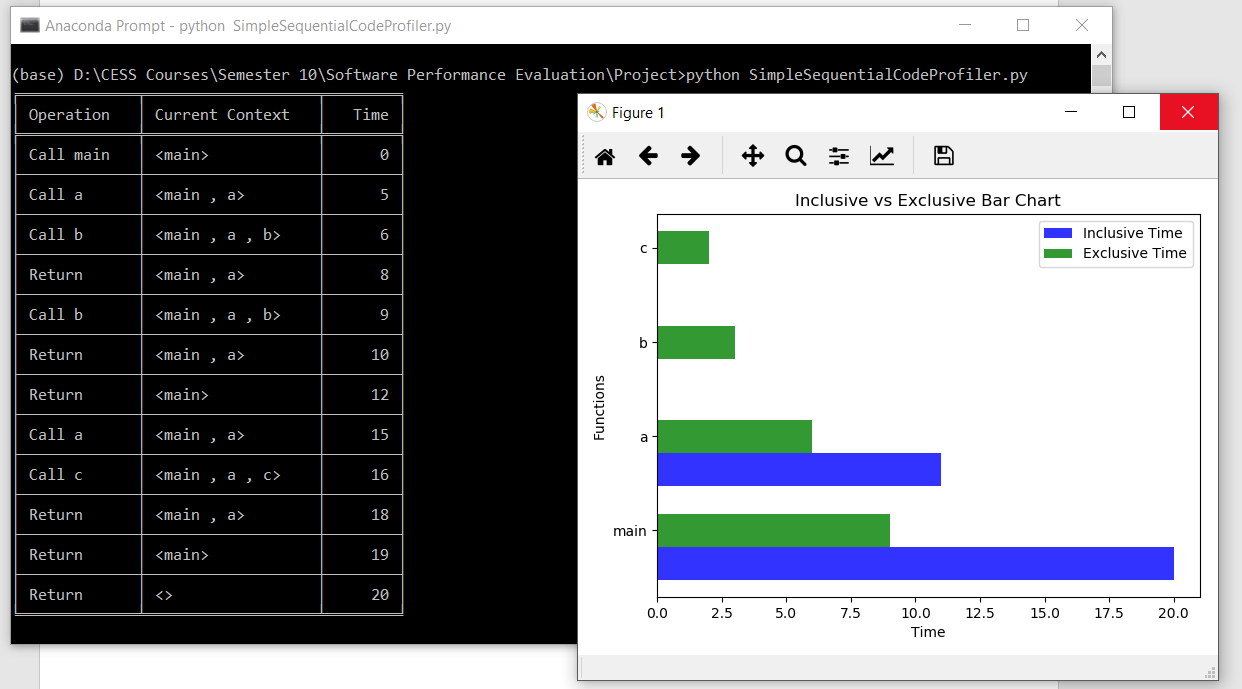
After reading the previous Logs.txt file and making sure that it is valid here are the results:  


1. **Call Stack Content over time**  
   
2. **Bar Chart**  
   Note that both functions b and c have no inclusive times that’s why the inclusive blue bar is not visible  
   
3. **Call Context Tree**  
   The red color on the path: main > a > b indicates that this is the hottest path (most frequent)  
   The black color indicates a normal (non-hot) path like main > a > c  
   

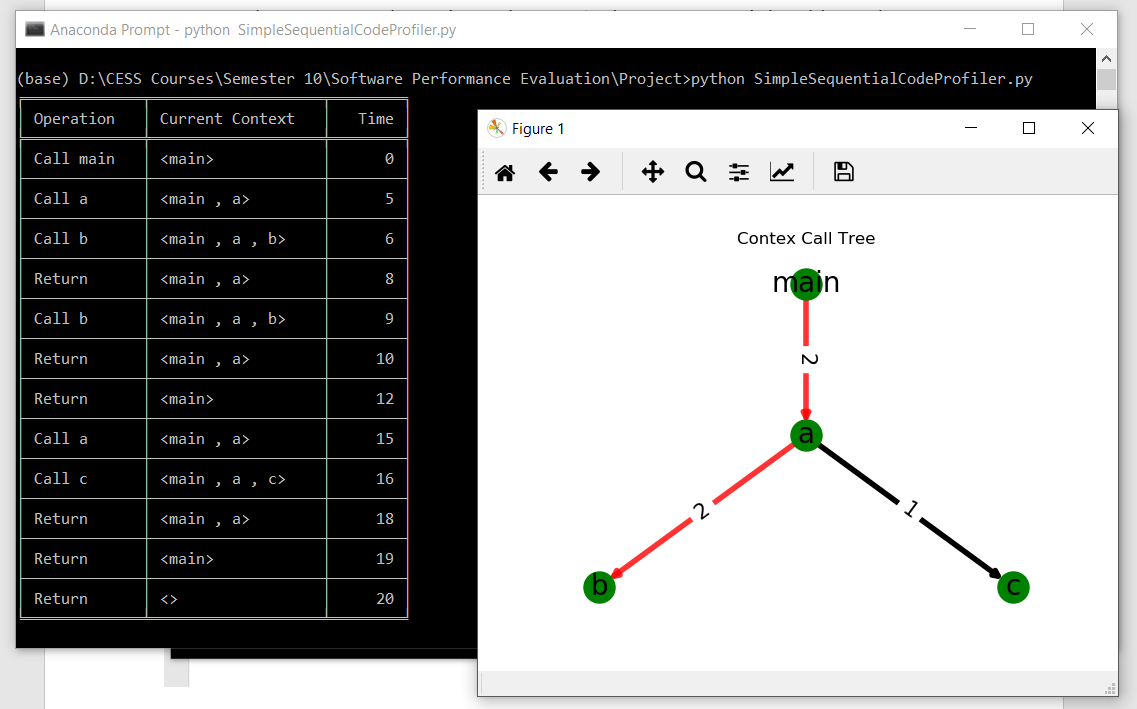
You can also try to run the code on the terminal BUT you won’t be able to change the logs file directly from the terminal you’ve to change it in the code I’ll change this later as I don’t have much time in my hands now.

Also, make sure that the logs file is in the same directory as the python file and the libraries mentioned above are installed in order to make the code run.

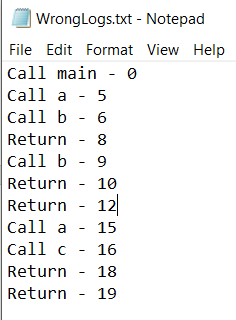
Type in the following command on the terminal in the python file directory folder  

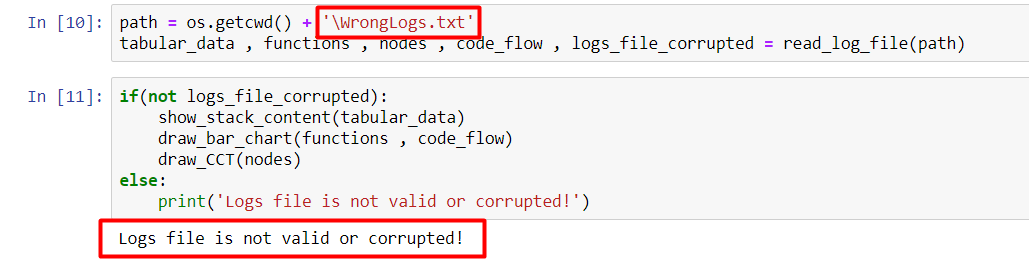
Press the ‘X’ button on the previous Figure1 window to make the next figure appears.



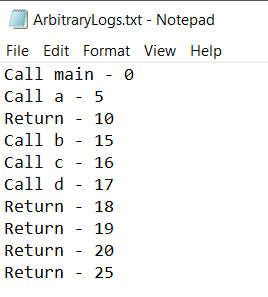
Next, I tried here to read a corrupted logs file named WrongLogs.txt by removing the main’s return statement.

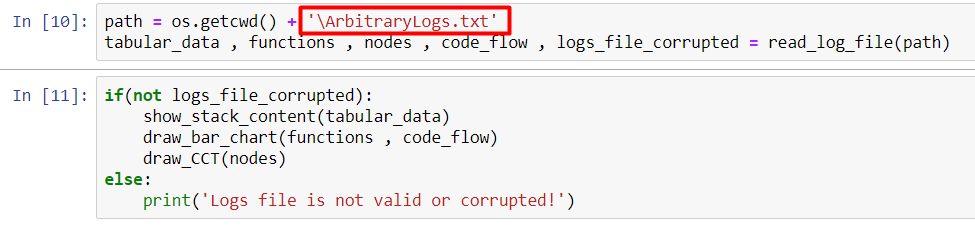


And here are the results:

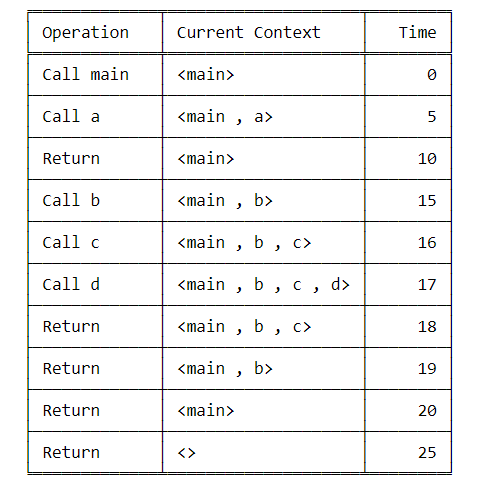
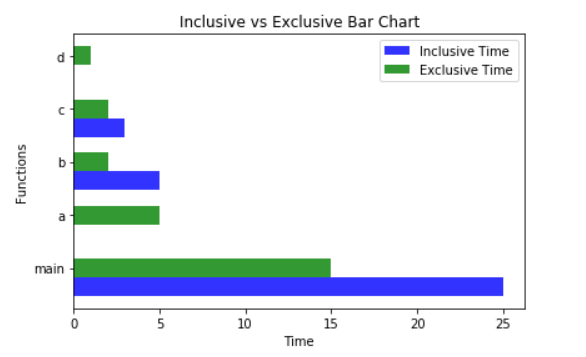


Finally, I tried another logs file I created to make sure that everything is working correctly.





And here are the results:

1. **Call Stack Content over time**
2. **Bar Chart**
3. **Call Context Tree**

