## Source Code Explained

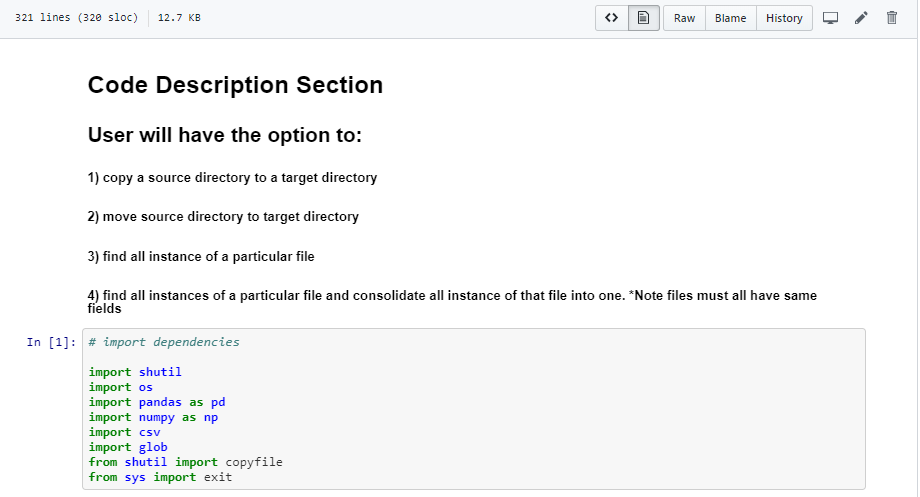
Overview:

In order to copy all historical fleet utilization data folders/files, in different directories throughout the company’s data warehouse and then create a new single directory for all these files to live, the OS and Shutil Python libraries were used. The OS module must walk from root to branch in each directory, this includes each customer, each time period, and each data folder to find files of interest and then copy those files to the new single directory

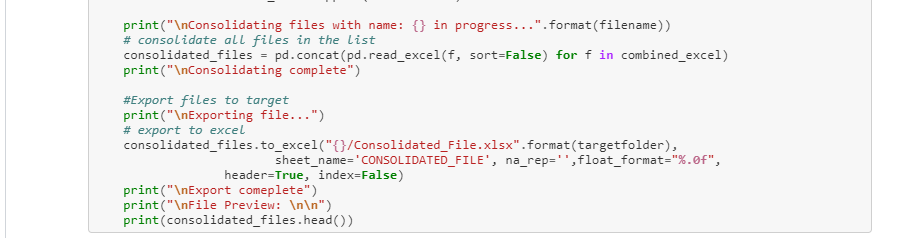
Four primary functions were built in order to accomplish these requirements: (1) copyDirectory, (2) moveDirectory, (3) findFile, and (4) findConsolidateFiles. Each function communicates with the Operating System directly in order to manipulate the file system. The copyDirectory function allows users to make a copy of an existing directory with all its contents and output the copied directory to a new directory. The moveDirectory function allows users move an existing directory to a new location. The findFile function allows user to search for all instances of a file and receive the file locations of all instances of a file name. The findConsolidateFiles function allows users to search for all instances of a file in a given directory, then consolidate all instances of the files into one single file. The findConsolidateFiles function is backbone of the File Explorer Application and is the function which solves the business requirement needed.

The backend code to test the four primary functions were built first using Python’s Juypter Notebook. Once the core backend functionality proved to function correctly, PyQt5 was used to translate the code to a frontend graphical user interface.

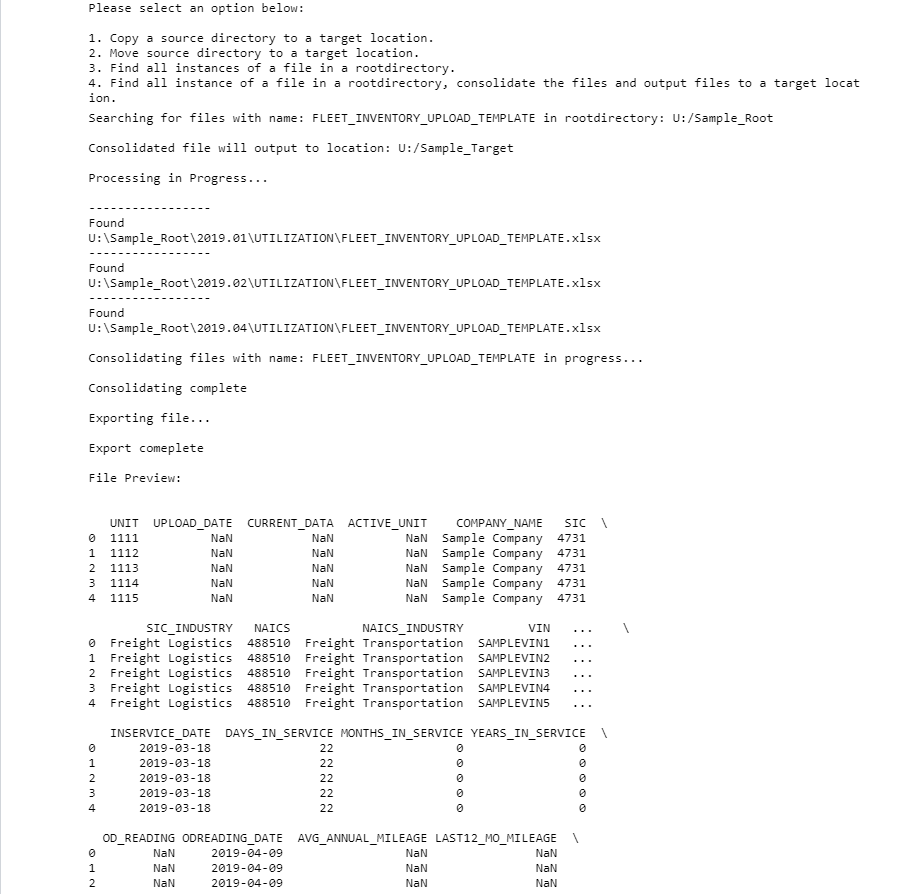
Juypter Notebook backend source code:







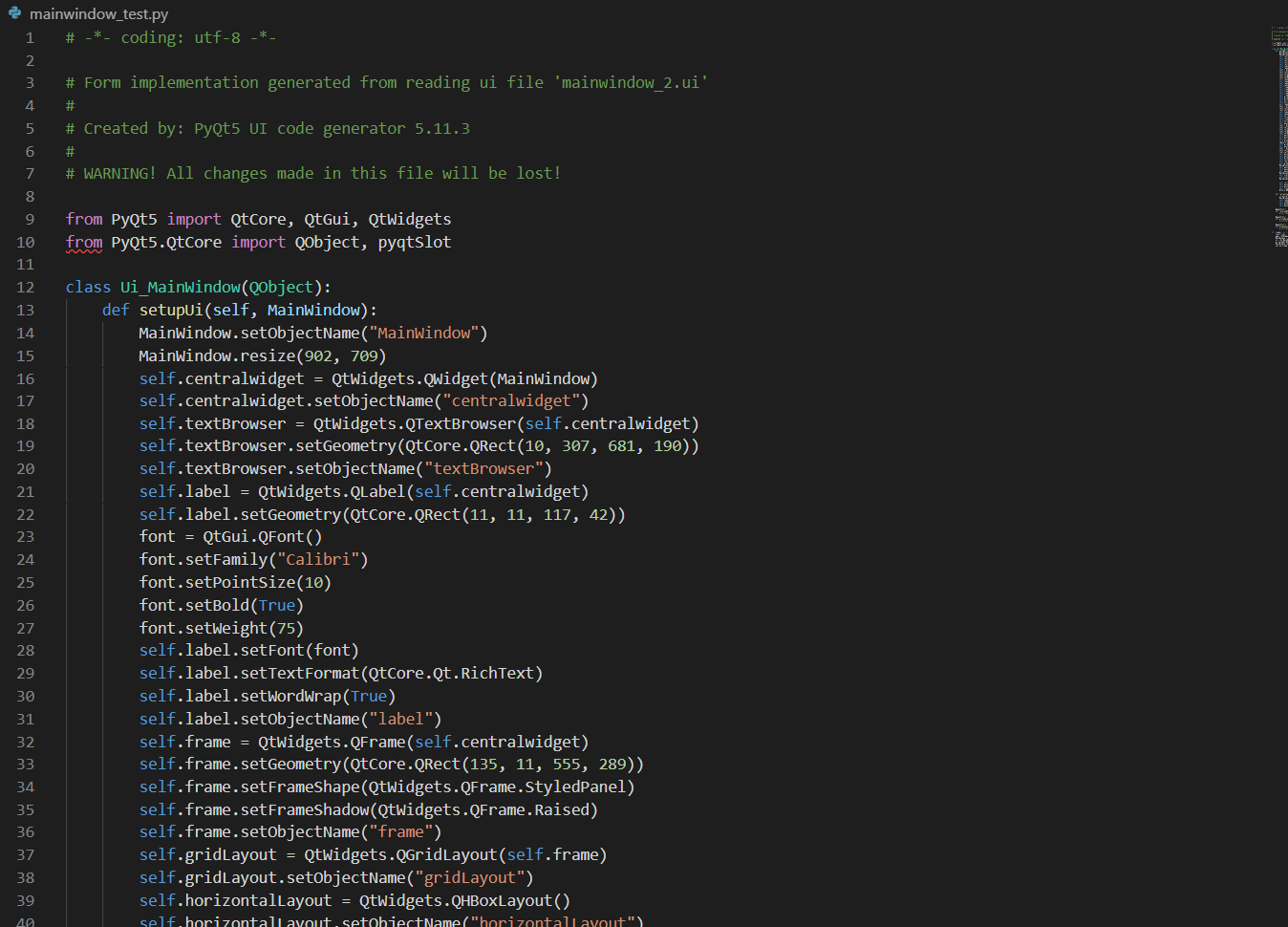
# Testing Section



The core frontend source files responsible for executing all functionality include: mainwindow\_fsat.py, fileExplorer\_model.py, model.py, pandas\_table.py. All other source file help to support these core source files.

**Mainwindow\_fsat.py** – builds the skeleton structure of the GUI and defines key methods.

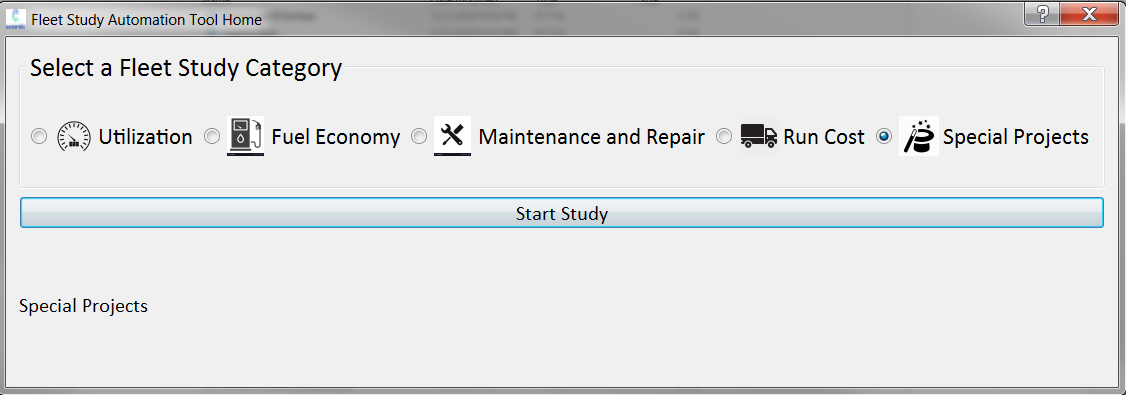
The code below outlines the UI components and placement. It also connects our key functions to methods





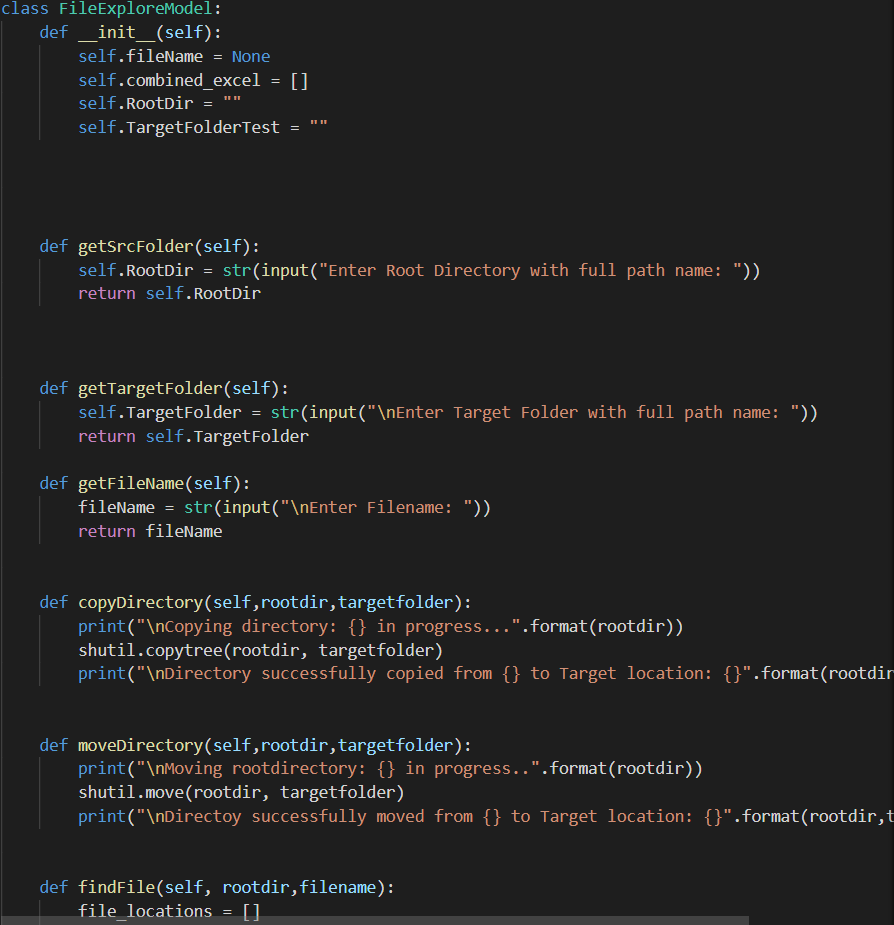
The code below defines a method to translate the UI to mainwndow\_fsat and launches the UI

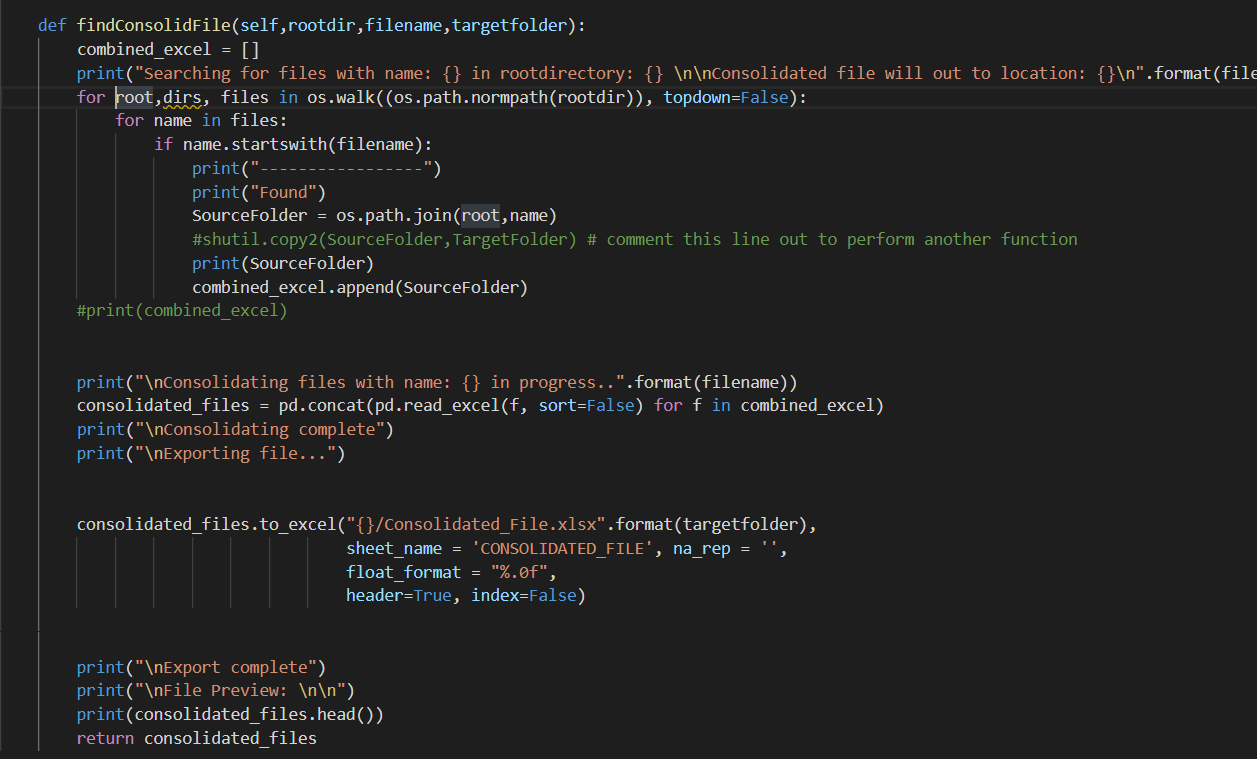


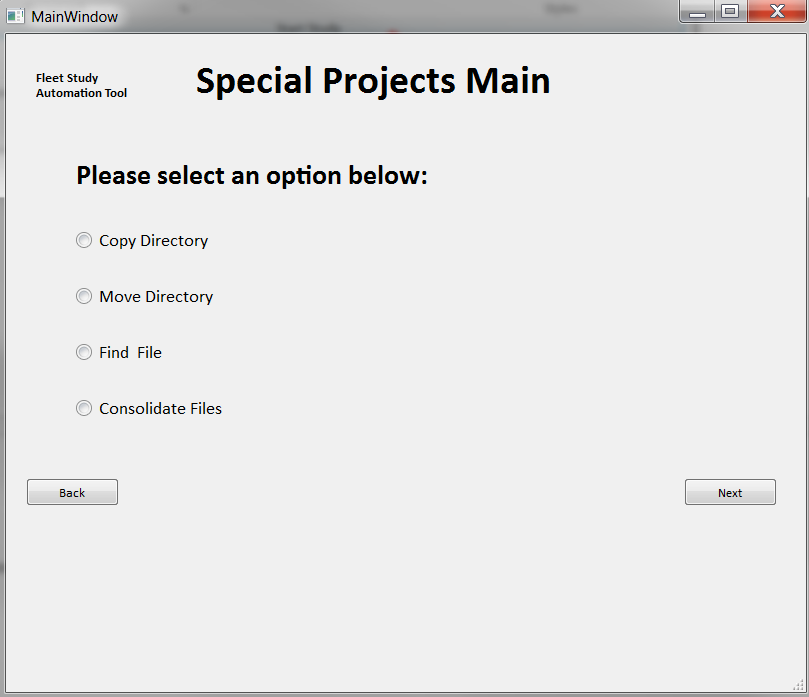


**fileExplorer\_model.py** –fileExplorer\_model.py uses the Model-View-Controller concept. The idea behind the Model-View-Controller paradigm is that the GUI should be separate from the logic of the application, with very well defined links between the two. This makes it easier to modify the logic of the model without having to change any code relating to the GUI. Similarly, editing code relating to how the GUI operates in response to user input should not influence the code controlling the logic of the mode

A FileExplorerModel class is created with methods that allow for our core four functionalities of the program.

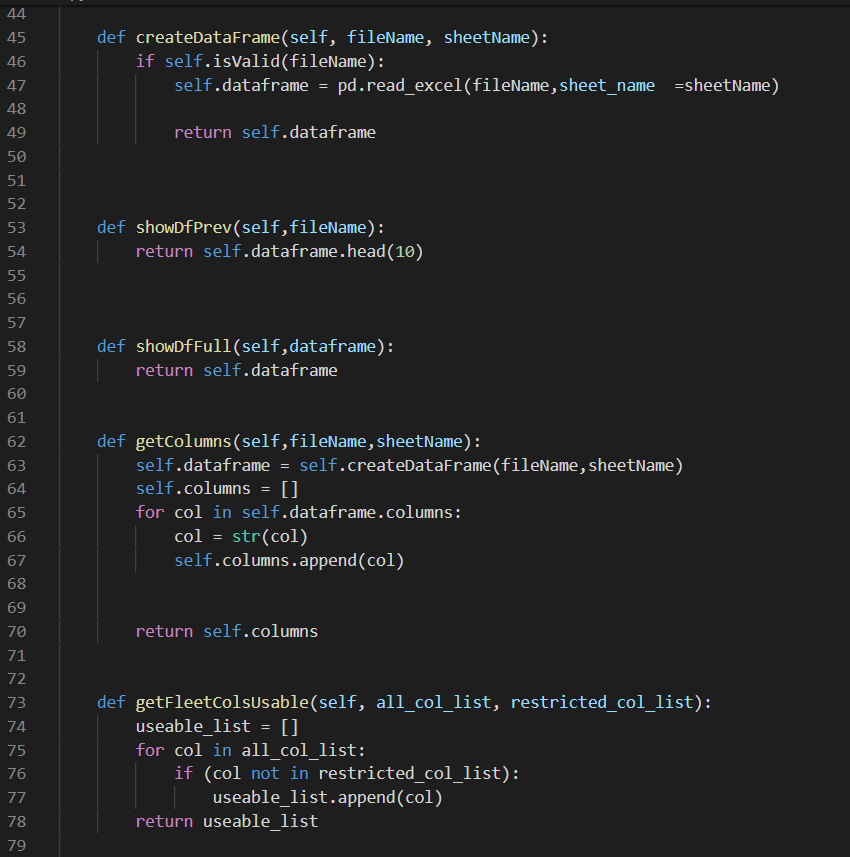


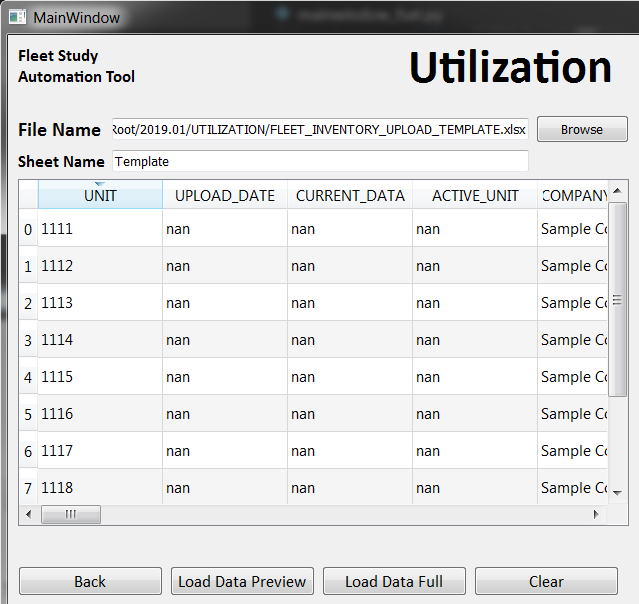




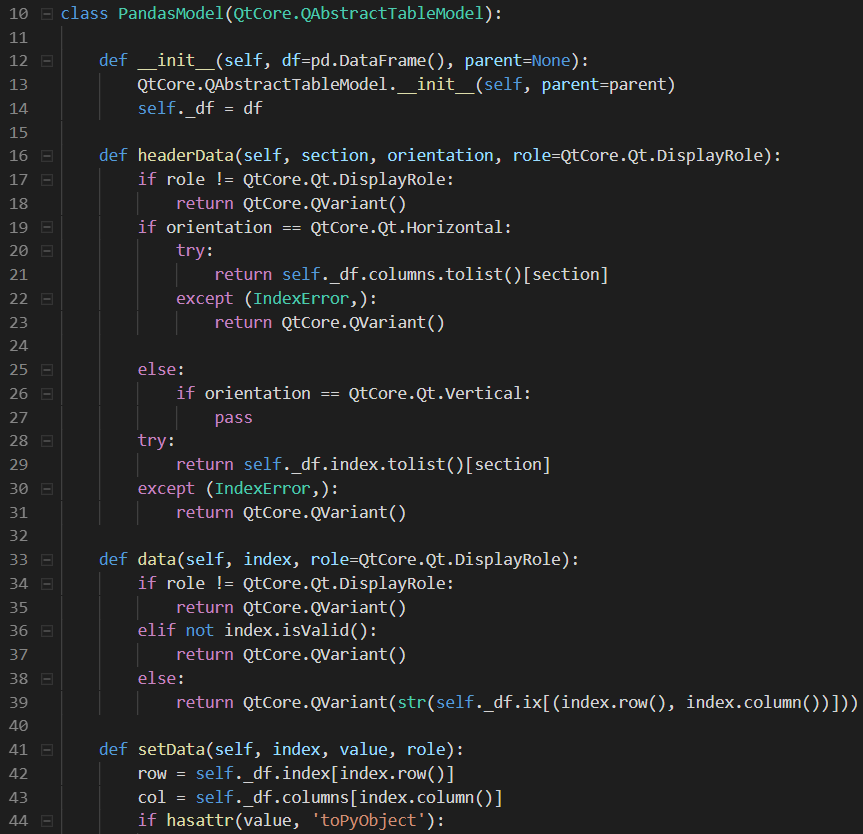
**model.py** –model.py uses the Model-View-Controller concept. The model.py file contains core functionality of the program. The model.py checks the validity of file names and contents then builds uses the pandas\_model class to output file contents in a pandas data frame on the programs GUI

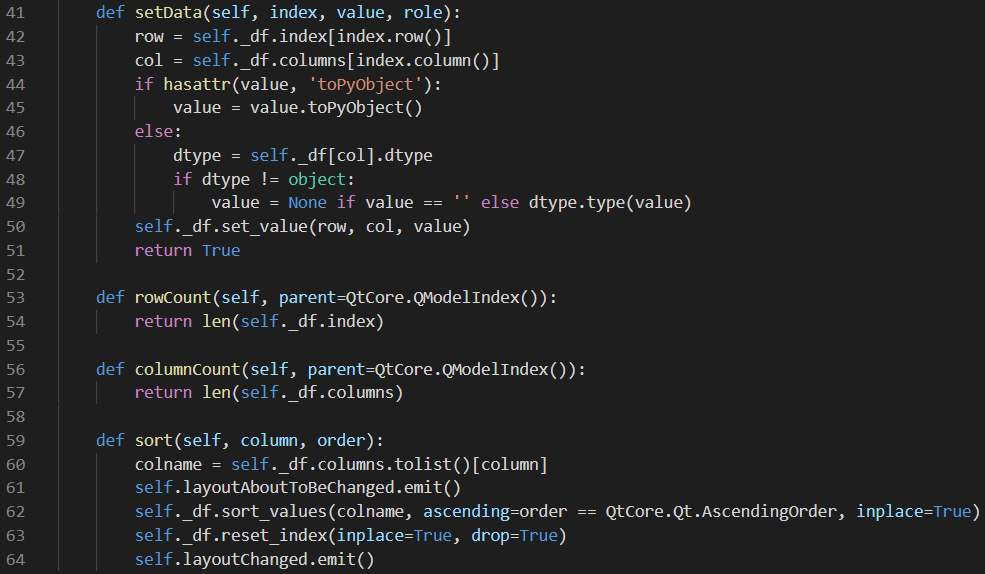






**pandas\_table.py** –pandas\_table.py is the final model class which uses the Model-View-Controller concept. The pandas\_table.py file creates a file that builds a method that allows for a dataframe to be displayed on a GUI.





To see a full step-by-step walk through of the File Explorer Application and functionality, please see the ‘File Exploration Application User Manual’ documentation.