Project Layout:

The top level directory of the project contains this readme, pseudocode.txt and a folder titled “src”.

Inside the “src” folder contains the source code files, along with a folder titled “Examples”.

Source code:

test.py is an early test file and can be ignored.

scheduler.py contains the implementation of both Matt’s Algorithm and the traditional greedy algorithm

filereader.py contains an algorithm for taking input files and converting them to a data set ready to be processed by the algorithms. Handles input format quite liberally.

filegenerator.py contains an algorithm for randomly generating example files, as well as an algorithm for printing the results of the processed data to file.

makeimage.py contains an algorithm for generating diagrams of the created schedules. Can save to png file, or show in a temp file.

gui.py contains the gui components + logic.

Examples folder:

The “Examples” folder contains three more folders. “Diagrams” contains all generated diagrams from the examples. “Input Files” contains the input files used to generate the example results, and “Raw Output” contains text file dumps of the results of the algorithms.

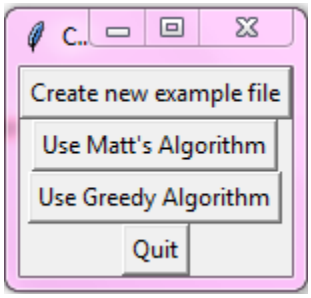
All input files, diagrams, and text output files are named correspondingly with the files that generated/were used to generate them.

How to use:

Enter terminal/cmd prompt from the ‘src’ folder and enter “python main.py”

This will launch the gui.

Main Panel:



There are four buttons on the main_panel.

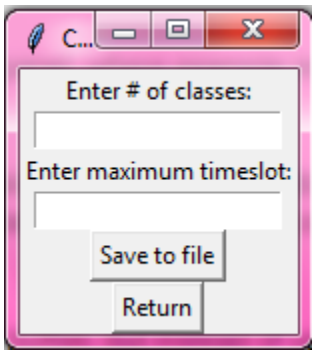
Create new example file will bring you to the example_panel, where example input files can be randomly generated.

Use Matt's Algorithm will first bring up a file browser dialogue. Once a compatible input file has been selected, the file's data will be processed using Matt's Algorithm. Upon completion, the results_panel will appear.

Use Greedy Algorithm works identically to the previous button, but instead of using Matt's algorithm, it uses the traditional greedy algorithm.

Quit closes the gui and terminates the program.

Example Panel:



There are two text fields and two buttons on the example_panel.

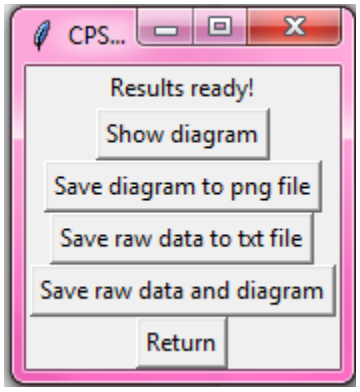
The first field, labelled "Enter # of classes:" is used to determine how many different classes should be randomly generated when creating an example file. This text field will only accept digits as input.

The second field, labelled “Enter maximum timeslot:” is used to determine what the maximum time can be for a class to end. Like the previous field, this will only accept digits as input.

“Save to file” will open a save file dialogue. Enter the desired name of the file (‘.txt’ is appended by default) and a randomly generated input file will be saved to the chosen path.

“Return” will return the GUI to the main_panel.

Result Panel:



The results panel has five buttons.

“Show diagram” will generate a diagram of the completed schedule in a temporary file.

“Save diagram to png file” will open a file save dialogue. Once a path and filename has been chosen (‘.png’ is automatically appended by default), a generated diagram will be saved to the path.

“Save raw data to txt file” Opens a file save dialogue similar to the previous button (but appends ‘.txt’ by default). Examples of both types of output are included in the “src/Examples” folder.

“Save raw data and diagram” will open the txt file dialogue first, then the png dialogue, allowing you to save both files at once. If you forget which file you are saving, it is shown in the top left corner of the dialogue.

“Return” will return the GUI to the main_panel.

Input

Input is given in the form of a text file.

The only requirement for valid input is that there exists two or more numbers in the file that are separated by something other than digits (e.g. whitespace, special characters, etc).

Note that class times are represented by strictly increasing numerals, with no special characters joining them (i.e. 4:56 will be treated as two separate numbers, 4 and 56). The program was designed to run with pairs of increasing numbers. Therefore, every even ‘nth’ number should be greater than the odd ‘nth-1’ number. If this rule is violated, unexpected results may occur.

Finally, classes that begin at the same time another class ends will be considered to conflict with that course, as it is assumed that there needs to be some non-zero interval for the first class to pack up and leave and the second class to set up. Allowing classes to begin and end at the same time without conflicting simply requires changing the ‘>’ to ‘>=’ in “scheduler.py” in the places denoted by comments.

Otherwise, there are no other formatting rules, which are left up to the user.

Output (examples):

The examples are located in ‘src/Examples’. There are eight examples in total. Three prefixed with ‘customtest’ and five prefixed with ‘gentest’.

‘customtest’ files were written by hand, and ‘gentest’ files were generated randomly by the filegenerator module.

The resulting output files and the input files used to generate them are located in separate directories. The directories are named accordingly to the file type they contain, but files across the three directories correspond to each other.

The diagrams are located in the “Diagrams” directory and are created at a relatively low resolution in order to keep file sizes low. Zooming in may be required to see the full detail of the diagrams.