FastGrade User Manual

Nicholas LeBow

May 8, 2019

NOTE

FastGrade used pickle (`.pkl`) file to store program state and student data. Due to security risks associated with pickle files, only use such files if they were received from a trustworthy source or generated locally.

1 Installation

FastGrade requires only Python 3.6 and PyQt5 to run. PyQt is included in several python distributions, including Anaconda. To install, just move all files contained in the archive to the directory you wish to run it from. All program files will be generated either in the directory which contains the program file, or in a subdirectory unless the user specifies a different location. For certain export operations, the package `xlrd` is required if the user wishes to interact directly with Excel files. To start FastGrade, simply execute the file `FastGrade.py` either from the filesystem, the terminal, or from within an IDE such as PyCharm or Spyder.

2 Using FastGrade

2.1 Font Size

Font size can be change using **Ctr+** and **Ctr-**

2.2 The Initialize Tab

When the program is started, the **Initialize** tab is shown. All other tabs, and some controls, are disabled until the following information has been provided:

- **User name**: This is used to identify the output files created by individual users. It is included in the file names of comment definition ('DCQ') and comment assignment ('ACQ') files which are created after setting a new user name. Note that user names are not enforced: If two users set the same user name, they will 'impersonate' each other. For this reason, if DCQ or ACQ files will be shared or merged, all individual user names should be unique. To save a changed user name, the button **Set User Name** must be clicked.

> NOTE
>
> If you change the user name, do not forget to click **Set User Name**, otherwise the old user name will be used instead, leading to unintuitive behavior

- **Question number**: The question for which most subsequent operations will apply. Only student data and comments for this question will be displayed in the **Evaluate** and **Grading** tabs.

- **Student data location**: Specifies where student data will be loaded from. The target may be either a directory (assumed to contain one subdirectory for each student), or a python pickle file (`.pkl`). Select the appropriate target from the drop-down list.

> NOTE
>
> Loading student data from a directory will read and execute each relevant script it encounters. Depending on the number of student solutions, this can take a long time. The progress bar at the bottom of the **Initialize** tab shows an estimate of the work completed. Once the process completes, the file `student_data.pkl` will be generated in the program directory. This file may subsequently be loaded instead. Loading the pickle file is *much* faster than loading from a folder. Once an appropriate data file is available, it is generally preferable to load it rather than a directory.

- **CSV Output Folder**: The directory where output files will be stored. Defaults to the program directory.

- **Gradescale (optional)**: A python script which calculates the points to assign for a student solution externally. If not specified, the option **Use Custom Gradescale** in the **Grading** tab will not be available.

If **Autosave on Exit** is checked, closing the program will save all current comment definitions and assignments to a DCQ and a ACQ file respectively, with filenames corresponding to the current user name and the current question number. If the program is closed while this checkbox is unchecked, The user will be asked whether data should be saved anyway. When saving, any existing files with the same name will be overwritten without warning or confirmation. Data may also be saved at any time with **Menu** → **File** → **Save** or the shortcut **Ctrl+S**.

If **Autoimport Data for Current User** is checked, pressing the **Initialize** button will also search for and attempt to load all ACQ and DCQ files for the current user. This is for convenience in place of manually importing the files with the **import** buttons. In addition, when saving (eg. autosave on exit), existing files will be overwritten, which might lead to loss of data if data are not autoimported. The files are only searched for in the configured CSV output folder, and only files with the default naming scheme are considered. If no such files are found, this checkbox does nothing. The autoimported data can be overwritten by manually importing a different file.

Once the above settings are defined, use the **Initialize** button to load the relevant files. If the student data location was specified as a directory, it may take a long time for this operation to complete, as indicated by the progress bar at the bottom of the screen. After initializing the program, the import buttons and the remaining tabs become active.

NOTE

Saving data (including autosave on exit) will also save the configuration data in **Initialize** tab to the file `state.pkl`, located in the program directory. When the program is next started, the configuration will be read from this file (if present), so that it does not have to be re-entered again. However the **Initialize** button must still be pressed in this case, and any data files required must still be imported.

Load Exisiting Config File may be used to select a `.pkl` file to import. Imported config files are not modified. State data is generally saved to the file `state.pkl` in the program directory. This file contains only a minimal set of config data. It does not contain any student or grading data, which will be saved in DCQ/ACQ files instead. The state file is created or overwritten when data is saved (**Ctrl+S**, or save on close). When the program starts, it will attempt to load this file automatically.

Use the **Import** buttons to load existing comment definition (DCQ) and comment assignment (ACQ) files. If the **Merge instead of Overwrite** checkboxes are checked, data loaded from a file will be merged into the existing internal list of comment definitions/assignments, instead of replacing the internal list with data loaded from file (the default behaviour). If identical entries are present in both the internal list and the file data, only one instance will be kept after merging.

NOTE

Regardless of the question they originally applied to, and of the name of the selected file, all data read from DCQ/ACQ files will be loaded for the **current** question!

NOTE

Load DCQ files before ACQ files, since individual comment assignments will only be loaded if the corresponding comment definition is already present, and otherwise ignored.

NOTE

Opening files produced by FastGrade (particularly DCQ and ACQ files) in an external program while using FastGrade is **not** recommended, as other programs may prevent FastGrade from writing changes to those files. This may lead to instability or loss of data. In addition, be aware that saving a .csv file in Excel may change the delimiters used, making the file unreadable for FastGrade.

2.3 The Evaluate Tab

This tab displays student data and allows existing comments to be assigned to students. The **Student Code** area displays the contents of the script the current student submitted for the current question. The **Output** section shows (from top to bottom):

- The **file contents** of the solution file loaded for the current student and question.
- The **terminal output** generated by the student code, including errors and stack traces.

  NOTE

  The output window displays both program output and exception traces, separated by a newline character. The entire contents is passed to the analysis script, which must be able to handle trailing newlines and error traces in its input.

- The (or an) **expected output** for this question. This content is generated externally by the analysis script.

  NOTE

  Depending on the nature of the question, the expected output may not be unique. For example, if the question requires a Python `set` object to be printed, the student code may print the correct set, but with a different ordering of elements compared to the expected output, due to set's inherent lack of defined order.

- **Correctness analysis**, generated externally by the analysis script. This should indicate whether the student solution's output is correct, regardless of presentation (to automatically detect cases such as the example above).

- **Image output** produced by the script when run, if any. For image output to be loaded, scripts should save to 'plot.png' and not specify an absolute path (for example, using matplotlib: `fig.savefig('plot.png')`).

  > NOTE
  >
  > matplotlib clears image data from memory after `fig.show()` has been called. Therefore, `fig.savefig()` must be called BEFORE `show()` to ensure that the file is saved correctly.

> NOTE
>
> There is a time limit set for student script completion (default: 4 seconds). This may be changed in `FastGradeEngine.__init__()` but the default should be sufficient for most cases. Some functions, such as reading user input from the terminal, or displaying a matplotlib figure with `show()`, will block execution until the user reacts. Such calls will result in timeout messages, even if the script itself is otherwise correct.

The **Comments** section shows a checkable list of comments and their associated point values which may be applied to the current student. Negative total points are changed to 0 (not 0.0 to indicate that there are negative points). The contents of this section is stored in ACQ files. If the program is configured to use sum grading, an **Initial Points** entry is shown at the top of the list. This entry is automatically assigned to all students and may not be deselected. If a gradescale script is being used instead, this entry shows the script's path. In the second case, the point values displayed have no bearing on the actual number of points assigned as this assignment is handled externally by the gradescale script. The point values of initial and checkable comments may be adjusted, and comments may be added or removed, in the **Grading** tab.

In case alternative comment assignments were loaded (**Comment Difference with ACQ file** under the **Selection** tab), then this selection is shown between the checkboxes and the comments. These alternative assignments can be removed from the program by clicking on the **Remove alternative evaluation** botton. This will also hide the button. The alternative assignments can then only be viewed again by reloading the alternative ACQ file under the Selection tab.

The **Navigation** section shows information about the current student, question and selection. The points value displayed applies only to the current question. To move between students, use the up/down arrow keys, or type the **folder name** of a student into the field **Current Student**. When the cursor is placed in the **Current Student** field, the arrow keys may also be used to navigate. When the end (or beginning) of the list of students is reached, it will wrap around and continue from the beginning (end). Only students which are in the current selection will be displayed. If a student folder name is entered which does not exist or is not present in the current selection, the view will not be changed.

NOTE

The use of arrow keys to navigate between students may not work correctly under OSX, since those keys are rebound to moving the cursor.

As in the **Select** tab, the width and height of individual components on screen is adjustable by using the separators between them, and they may be collapsed and restored as required.

2.4 The Grading Tab

This tab allows users to create, modify and remove comment definitions, and select the grading mode to use when calculating point scores.

2.4.1 Adding new Comments

Type a description and click **Add/Rename Comment**. The description must not be empty, and must not be identical (regardless of capitalization) to an existing comment for this question. It may also not contain commas to ensure compatibility of DCQ file data with the CSV format. Creating a comment and assigning points to it are two separate operations. Comment creation will always assign the default point value to the new comment. After creating a new comment, it will be selected in the comment list.

NOTE

When adding new comments, make sure that no comment is already selected in the list. If an existing comment is selected, that comment will instead be renamed when **Add/Rename Comment** is clicked. To deselect a comment in the list, double click it.

2.4.2 Renaming existing Comments

Select the comment to rename from the list, enter the new name for that comment and click **Add/Rename Comment**. If no comment was selected, this will create a new comment instead. Renamed comments retain their point values and remain assigned to students.

2.4.3 Setting the Point Value of an existing Comment

Use the **Comment Points** control to set the desired point value, select the comment from the list (if it isn't already selected) and click the **Set** button. Setting the initial points value works the same way, but there is a separate input field and button, and the 'Initial points' comment does not need to be selected from the list.

2.4.4 Deleting Comments

Comments may only be deleted if they are not assigned to any students, in order to avoid undefined comment assignments. To delete a comment, check the checkbox **Show only unused comments**, then select the comment to delete and click **Delete Comment**. If the comment is not shown in the list of unused comments, it must first be unchecked for all students in the **Evaluate** tab before it may be deleted.

2.4.5 Grading Modes

The grading mode may be toggled between **Point Sum Grading** and **Gradescale** modes. With point sum grading, the point values of all comments assigned to a given solution (including the **Initial Points** value, which is applied to all solutions) are summed to obtain the total score for that solution. For gradescale grading to be available, a gradescale script must be set in the **Initialize** tab. In gradescale mode, the point values for all comments are ignored and a list of assigned comments is passed to the external script, which is expected to produce an appropriate point value for the solution. In both modes, the total score for each student is calculated by adding the point values for all questions.

2.5 The Select Tab:

This tab provides options to filter the list of students displayed (and accessible) in the other tabs based on specified rules, and is typically intended to be used **after** grading. At any time, the selection may be reset to include all students by clicking the **Reset Selection** button. A list of currently selected student folders is shown on the left, and a list of previously saved selections is shown in the lower center of the tab. The onscreen size of the various components may be changed by dragging the separators between them.

NOTE

The selection files generated by this tab store lists of student folder IDs in CSV format. These files can be modified by hand to produce user-defined selections not obtainable using the tools offered, however it is then the user's responsibility to ensure that modified files represent valid selections. Program instability or loss of data may result from inappropriate modifications.

NOTE

Applying a selection may modify the results produced by some export options in the **Combine Data** tab to include only students in the current selection. Some export options may also ignore the current selection and always export data for all students.

The current selection is called "**default**", is always present in the **Available Selections** list and cannot be deleted. All students in this selection are displayed in the **Currently selected students** list. Applying selection criteria in the **Define Selection Rule** area will not modify the current default selection until the **Apply Selection Rule** button is pressed. Changing the current selection will set the current student in the **Evaluate** tab to be the first student of the updated selection.

The current selection can be modified by defining constraints which students must meet to be included in the selection. To do so, check the appropriate checkbox in the **Define Selection Rule** area to activate a constraint, and define the required parameters as needed. By default, the **All with Points in Range** constraint considers points assigned for **all** questions. Points calculation may be restricted to the current question only by clicking the **Only consider points ...** checkbox.

A list of defined comments for **all** questions is provided for the **All with Comments ...** constraint. The question to which a given comment applies is indicated by a question number in parentheses next to the checkbox for that comment, since the same comment may be defined independently for several questions.

NOTE

Checking and then unchecking a constraint will **not** revert the current selection to its state before the constraint was checked! This is because constraint modifications always apply to the **current** selection. This is necessary for the set operations (AND/OR/EXCEPT). Use the **Reset Selection** button to revert the selection to include all students.

A snapshot of the current selection may be saved by entering a name in the field next to the **Save Current Selection** button, and then clicking the button. The save button is disabled until a valid name has been entered. This will create a new selection file, which will appear in the **Available Selections** list. Saving a selection with the same name as an existing selection will overwrite the original file. Saved selections may be deleted by selecting them in the list and pressing the **Delete chosen Selections** button.

To load an existing selection, select it in the list and press the **Load chosen Selection** button. This will overwrite the current (default) selection with the data contained in the selected file. The saved file will not be modified by subsequent operations, which will be applied to the current (default) selection.

The current selection may be modified by applying additional constraints to it, or by combining it with a set of saved selections. To do this, select the saved selections from the list and press the appropriate **Modify Current with Selected ...** button. This applies a logical operation to the current selection:

- **AND**: The resulting selection will contain all students present in the initial selection, which are also present in **all** selections from the list.
- **OR**: The resulting selection will contain all students present in the initial selection, which are also present in **any one** selection from the list.
- **EXCEPT**: The resulting selection will contain all students present in the initial selection, which are **not** present in **any** selection from the list.

To use Comment Comparison:
This is intended for finding discrepancies between the comment assignments made by the current user, and those made by a different user. It will also load the alternative comment assignments and visualize them under the **Evaluate** tab. It requires access to the other user's ACQ file for the relevant question.

- Assign comments, or import an ACQ file for the current question.
- Specify a selection to start from as above, reset the current selection, or select an existing selection and press **Load chosen Selection File**
- Press **Comment Difference with ACQ File** and select the other user's ACQ file to compare with. The current selection will be modified to include only those students which were not assigned identical comments by both users.

  NOTE

  Both ACQ files should refer to the same question. The way in which definitions and assignments are identified internally means that loading files created for different questions will result in unrecognized assignments disappearing, even if the comment string matches.

- Continue working with the resulting selection, or save it as described above.
- The alternative comment assignments can be viewed under the **evaluate** tab. The evaluate tab will now also show the **Remove alternative evaluation** botton. By pressing this botton, all alternative comment assignment are removed (but the ACQ files are not affected), the alternative comment assignments are no longer visable, and the remove-botton is hidden. The alternative comment assignments have been assigned to all students (the default selection) for the current question. Changing the current question will therefore show empty alternative selections. When a new ACQ File is loaded for comparison, it will replace the current altertive comment assignments.

2.6 The Combine Data Tab

NOTE

The files generated in this tab are intended for external use and readability. They **cannot** be re-imported. Only DCQ and ACQ files may be imported.

This tab provides routines for generating reports, typically in CSV format. When an output option is selected from the dropdown list, a description of the function and an output sample will be displayed. To apply the function, select it from the list and click the **Generate** button. All output files generated will be saved in a subdirectory `/overview` of the CSV output folder currently set in the **Initialize** tab. Files with the same name will be overwritten without warning.

2.6.1 Generating Grade Reports

The option **Generate grade report formatted class overview file** is intended for reporting the final scores for all students after grading is complete. It requires the user to select a reference file (either an Excel file, or a CSV file) which will be used to determine the order in which to output student records, and to identify missing students.

NOTE

For Excel files to be imported, the Python package `xlrd` must be installed. If this package cannot be loaded, the export operation will not produce any output. However, the user may bypass this by manually copying the student list from the Excel file, saving it as a CSV file and importing that instead.

3 Internal Operation details

3.1 Components

Fastgrade is designed with two major components: `FastGradeEngine`, which handles all data processing and I/O, and `FastGrade`, which manages the user interface and passes data from it to the engine component. Additional objects include `Students`, which contain all data associated with a specific student, `Filters`, encode a set of selection constraints, and `CommentTuples`, which store information relating to a specific comment.

3.2 Expected Data Format & Structure

Student data imported from the file system is assumed to be stored in a single directory which contains one subdirectory for each student. Student files for all questions are expected to be found in this subdirectory. Files in the top directory, or in deeper subdirectories will be ignored. Student files are assumed to be named according to the format used by OLAT. Only one single student file is loaded for each question. If there are multiple files present for a single question, the most recent file (as determined by the timestamp appended to its file name) is selected. Files which do not include a timestamp in the file name are assumed to be the first (oldest) version to be uploaded, and are only selected if there are no timestamped files for that question.

3.3 Importing

FastGrade does not operate directly with student files in the file system. Instead, student data must first be imported. During this process, all relevant student scripts are read and executed, and the output is collected along with its analysis. All this data is stored in a `.pkl` file in the program directory. After importing, the original student data is not accessed again and may be removed. In particular, student scripts are not executed again after importing. Depending on the class size and number of questions on the exam, importing this data may take a long time. Once a data file has been generated, it may be used directly, providing much faster initialization times. It may also be distributed to users who do not have access to the original student data.

3.4 DCQ and ACQ Files

FastGrade stores mutable data in human-readable CSV files in the configured CSV folder. A separate data file is generated for each user and question. DCQ files store comment definitions, including a randomly-generated Comment ID value. This value is used to associate comment definitions with comment assignments and should be globally unique. The precise value of the ID is not important and may be changed to something more meaningful as long as all IDs remain unique. ACQ files store lists of comments assigned to each student for a given question. Only the comment ID is stored, and each ID in an ACQ file must match one in the DCQ file used for that question.

While the name of the user which created a DCQ or ACQ file is included in the filename, along with the number of the question it was created for, in principle any DCQ/ACQ file may be imported by any user for any question, as long as corresponding DCQ/ACQ file pairs are used (i.e. files containing the same comment IDs).

4 FastGrade Code Settings

Some settings are hard-coded and cannot be changed from the user interface. If required, they can however be changed in the program file.

NOTE

Do not change the settings described below unless you know what you are doing! Inappropriate modifications may result in unexpected behaviour or program instability!

NOTE

After changing hardcoded settings, using the program with files generated before the change (including DCQ and ACQ files, student data files, program state files, etc.) may result in unexpected beaviour, instability or loss of data. Any existing `state.pkl` file should be deleted before restarting the program.

All configurable settings are contained in the method `FastGradeEngine.__init__()`, along with a brief description of their purpose. Do not modify anything other than the following settings, as all other variables may be changed by the program during operation. If the following settings are modified, remember to update the descriptions (including at the top of the program file) to match.

- `input_textfile`: Name of a text file in the program directory which will be passed to each student script when it is run. The contents of this file will be protected from modification by the student scripts. If this file does not exist, an empty file will be created.
- `output_plot`: Name of an image file in the program directory which will be monitored for student output. Student scripts may save image data to this file, and the image will be displayed in the Evaluate tab if it exists. Other image files will be ignored. This file is automatically reset after each student script is run.
- `config_path`: Name of a pickle file in the program directory. This is used to save configuration settings (such as the data entered in the Initialize tab). This file will be read at program startup, and overwritten when saving (including save on exit). If it does not exist, a new file will be generated upon the next program run.
- `student_data_pkl_path`: Default name of a pickle file in the program directory. Used to save student data. If student data is imported from a directory structure, this file will be overwritten.
- `analysis_module`: Name of a Python module expected to be found in the program directory. This will be used to obtain correct solutions, and to analyze the output generated by student scripts. The output it generates will be displayed in the Evaluate tab. See the sample script `FastGradeAnalysis.py` for more information.
- `total_questions`: The number of questions on the exam.

  NOTE

  Fastgrade uses 0-based indexing internally, but assumes 1-based indexing for question names. Thus, for a 10-question exam, references to the question number (e.g. in student file names) are expected to be numbered 1,2,..10.

- `initial_font_size`: Initial font size.

- `student_data_id_val`: Arbitrary ID string, used to identify compatible `.pkl` files. Can be modified to enforce version compatibility (i.e. only files generated with the same version of the program can be loaded). Should be a random string which is unlikely to be used for anything else.
- `timeout_seconds`: Time period allowed for each student script to complete. Added to avoid problems with scripts running infinite loops, or waiting for user interaction. The 4-second default should be sufficient for most situations. If this is set too low, even correct student scripts may not have sufficient time to complete, particularly when plotting, etc. Increasing the timeout can increase the time required to import student data from the file system, as each script is executed during this operation.
- `verbose`: Boolean flag. If set to `True`, FastGrade will print various notification to the terminal during operation. These are intended for debugging and can generally be ignored during normal operation.
- `subprocess_shell`: Boolean flag. Sets the mode used to execute student scripts. Enabling this may lead to unexpected behaviour on different platforms. Leave it disabled unless you know what you are doing.
- `initial_points_str`: Reserved comment description used to assign default points to all students.
- `DCQ_file_header`: Column names separated by commas, written to the first line in DCQ files. This string is used to identify valid DCQ files. Can be changed to ensure version compatibility for DCQ files.
- `ACQ_file_header`: Column names separated by commas, written to the first line in ACQ files. This string is used to identify valid ACQ files. Can be changed to ensure version compatibility for ACQ files.
- `ACQ_file_header`: Column name(s) separated by commas, written to the first line in selection files. This string is used to identify valid selection files. Can be changed to ensure version compatibility for selection files.
- `student_file_regex`: Regular expression used to filter valid student scripts when loading student data from the file system. Don't change this unless OLAT changes the naming scheme for student files.