Coding Review Strategy

- Readability and Maintainability
  - Self-explained variable/function name?
  - Is the Documentation there?
  - Is there meaningful internal comments?
  - Is there any magic values (constant using)?
  - Is it immediately obvious what a section of code is doing?
- Code Design
  - Is there unnecessarily repeated code (Copy and pasted)?
  - Too many lines in one function (Can the function be broken down?)
  - Are there functions/classes/variables which share too much information?
  - Are there classes/functions which handle too many functions at once or perform things in a rigid order (hard coded functionality)?
- Functionality
  - Did we handle the exceptions and errors correctly?
  - Does the code do what it is supposed to do?
- Testing
  - Do we have a good test coverage?
  - Are the test results accurate?
  - Are the tests clear and concise?
  - Are there any unnecessary or repetitive test cases?

Also think about "each team member gives general recommendations to the team on how to improve the quality of the development." for the video

Who's Reviewing What:

- Taiki: student_question_set.py (lines 1-200ish) (first half of file)
- Alex: student_question_set.py (lines 200ish - end of file) (second half of file)
- Ali: question_type_select.py (line 1 - 200ish) (first half)
- Nate: login_windows.py, multiChoiceQuestion.py
- Wanjing: questions_type_select.py (line 200 - end)

Wanjing(from function parseNumAndProceed() to the end)

The code's readability is satisfied, some of the methods have valid documentation and meaningful internal comments but functions inside <MultiChoiceQuestionPage> are missing their documentation and internal comments; it's good that all variables' names are self-explained but I do see some magic number showing in codes, perhaps these numbers can be signed to different variables that have valid, easily understood names. Roughly reading through the code brought up no confusion on variables names or having trouble understanding the functionality of a method. Anyhow the layout of this python file could be better if class <MultiChoiceQuestionPage> can be moved to a new file.

The init method of <MultiChoiceQuestionPage> are handling too many functions, and it has too many lines of code. It would be better if this class has different methods for creating and lying buttons so that after the breakdown, init method has fewer lines of codes and it handles only the relationship with its own master tk frame.

Methods in class <MultiChoiceQuestionPage> require attention in error handling, the good thing is all codes do what it suppose to do regards to its documentation or its method name.

Alex (from function processQuestions until EOF)

Variable and function names are generally okay with the exception of some variables which are frequently reused in many place without any explanation (‘index’ variable is the worst for this). Existing comments are ambiguous and unclear and some functions (save progress) just aren’t documented at all. Many snippets of code which deal with lists or reading from file use ‘magic values’ a lot, specifying certain indices to read from without explaining why or using constants (save highscores, checkans). The if/else block for displaying messageboxes is frequently repeated, and there are multiple different functions/code blocks related to file I/O which all accomplish the same thing but do it in a different way. Function length is okay, but many still violate single responsibility principles by not deferring more specific actions to functions. Hard coding in functions is frequent due to the mentioned magic numbers. Many functions also do not seem to try to error handle at all, and as a result, it is very easy to completely break them with basic ‘bad’ inputs such as empty lists. As a whole, the functions still do not perform what they are supposed to as there are noticeable bugs with the way question sets are read (always shows same set). Due to the fact that the class contains a combination of UI and backend functions that cannot easily be separated, automated testing is not done on it.

Nate:

multiChoiceQuestion.py - Good, code is readable; everything is clear. Variable names and function names are well written and understandable. Almost all docstrings are missing type contracts. Class could potentially use setter functions for ease of use.

login_windows.py - Good, code is readable and organized for the most part. Variable and function names are clear. Some functions have naming inconsistencies, such as using/not using underscores. Some functions are missing parts of or all documentation. Function length is good and functions are broken down appropriately. For now, the code functions how it should and there are no errors. Logging in as a student or a professor take the user to the appropriate windows correctly. The internal comments help well understand blocks of code.

Taiki:

Overall the code was good with thorough commenting to support it. Going down the code review checklist, the variable and function names were mostly self-explanatory although there were come one letter variable names that may be confusing for some. As mentioned earlier, internal commenting and the DosString were present throughout the code with meaningful content inside. Although magic values were used in the code, most of the use cases would have been impractical as it would’ve taken up an unnecessary amount of space at the top of the file. Even without the commenting, it was clear what each block of code did even with limited knowledge of the libraries in use. In terms of code design, there was little to no copy and pasted code as function calls and loop were used appropriately to avoid this. So called “megafunctions” don’t appear in the code, other than the init where all of the UI components were initialized. Breaking them off into functions would've made no sense because they are only called once at the beginning and not repeated anywhere so the long length of the init function is justified. Private variables (ex. self._foo) were used in appropriate places to hide information from other classes, and all of the public variables were UI objects which are usually public. None of the code had any hard coded functionality other than the init which has to no choice but to be hard coded. The QuestionSetGridView class however can be split up into multiple classes as it handles both the main menu screen and the answering of the question sets. There were some error handling involved in a few file IO blocks of the code, but most others were not for example file stream closing. From opening and running the code, the code seem to be working as intended and according to what is on the DocString.

Ali Basit Abdul-Hashim
Readability and Maintainability
Self-explained variable/function name?
- Though the variable names make sense after you become familiar with the program, it is pretty hard to understand what they are variables of, or what they contain at the first few analyzations.
Is the Documentation there?
- For the most part, documentation was there, but no doc tests to show examples of how the function/method works.
Is there meaningful internal comments?
- A few places did not have comments but worse than that different sections of code, for example selecting the answer type, multiple choice option, and number of questions in a set field were all jammed together without any comments or new line/empty line space division, making it extremely difficult to find a section or element you are searching for.
Is there any magic values (constant using)?
- No
Is it immediately obvious what a section of code is doing?
- No, you must either already be familiar with the program, or spend a good part of a day learning the different components of the program and how they interact with each other.

Code Design
Is there unnecessarily repeated code (Copy and pasted)?
- Not really.
Too many lines in one function (Can the function be broken down?)
- Some can be broken down, but it wouldn't help with anything, so I believe it is fine the way it currently is.
Are there functions/classes/variables which share too much information?
- Since we're using python, no not really.
Are there classes/functions which handle too many functions at once or perform things in a rigid order (hard coded functionality)?
- Classes, functions, and methods seem to be in decent proportions
- There are some hard coded elements such as the placing/positioning of the different fields and options on the gui. For example, if you want to add a new element (an option or input field) you have to either add it at the end so that you don't have to change the

positioning (column and row) of the existing elements, or you have to change all the column numbers and row numbers of the previous elements in order for them not to conflict. This is seen best in class QuestionTypeSelectPage.
- Also self._numPossibleAnswers = 4

Functionality
Did we handle the exceptions and errors correctly?
- Yes

Does the code do what it is supposed to do?
- Yes

Testing
Do we have a good test coverage?
- No, there is no doc testing or manual testing done

Are the test results accurate?
- N/A since there are no tests

Are the tests clear and concise?
- N/A since there are no tests

Are there any unnecessary or repetitive test cases?
- N/A since there are no tests