Homework 5

Due April 28, 2017

Submissions are due by 11:59PM on the specified due date. Submissions may be made on the Blackboard course site under the Assignments tab. Late submissions will not be accepted. Make sure your name and FSUID are in a comment at the top of the file.

1 boggle_gui.py (100 points)

In this final assignment, we will be revisiting the first module we put together this semester: boggle.py. This time, you need to extend your module to add the following features:

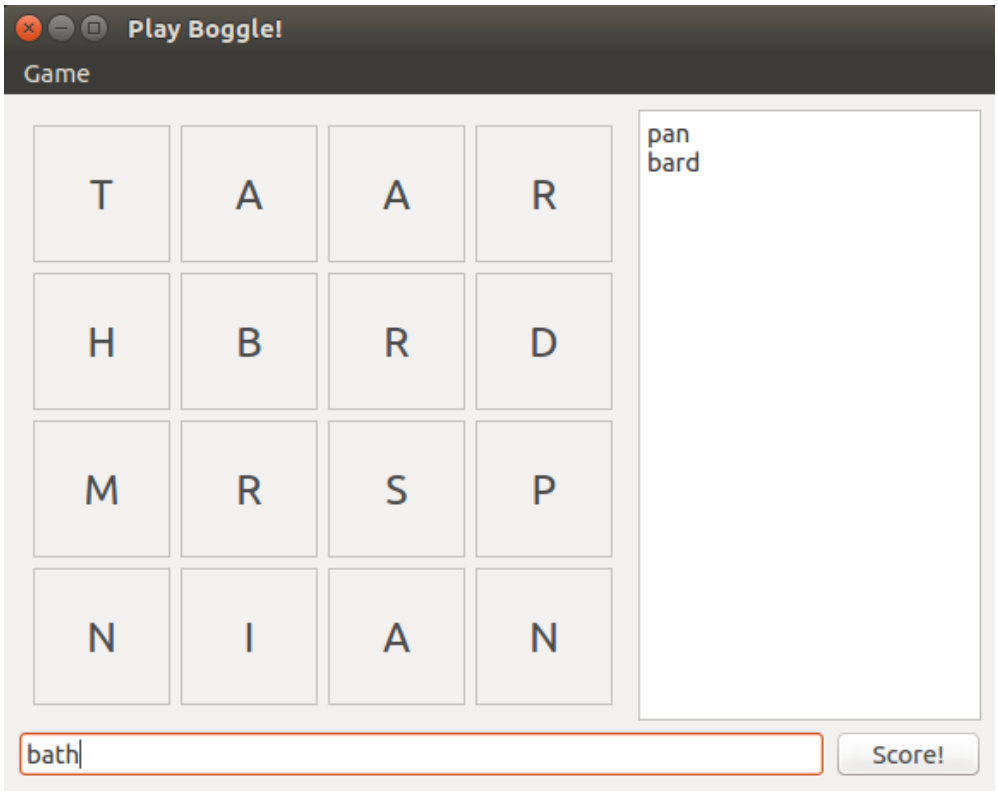
- GUI.
- Saving/loading games.

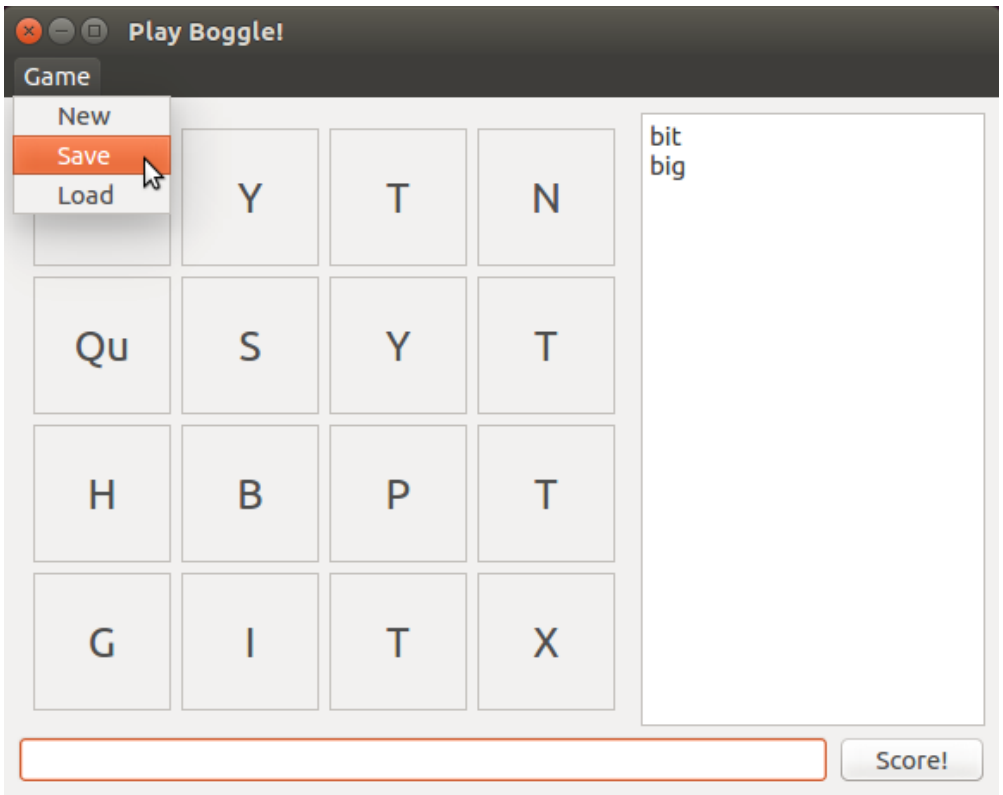
All of the game requirements of the first assignment hold (e.g. grid must be 4x4, dice must be randomly shuffled, same game rules and scoring apply, etc). You may repurpose your original boggle.py module for this project, but you will likely need to make significant structural changes so it may be best to start fresh and add in previously-used code bits as necessary. You will likely need to use a few of the things we've discussed this semester, including standard library modules like random, data persistence modules like sqlite3 or shelve, as well as, of course, a GUI toolkit (PyQt5). Please feel free to be creative with the look and feel of the application and search for creative, pythonic solutions to the challenges you encounter (including revisiting your original boggle.py code – look for ways to make it better!).

GUI

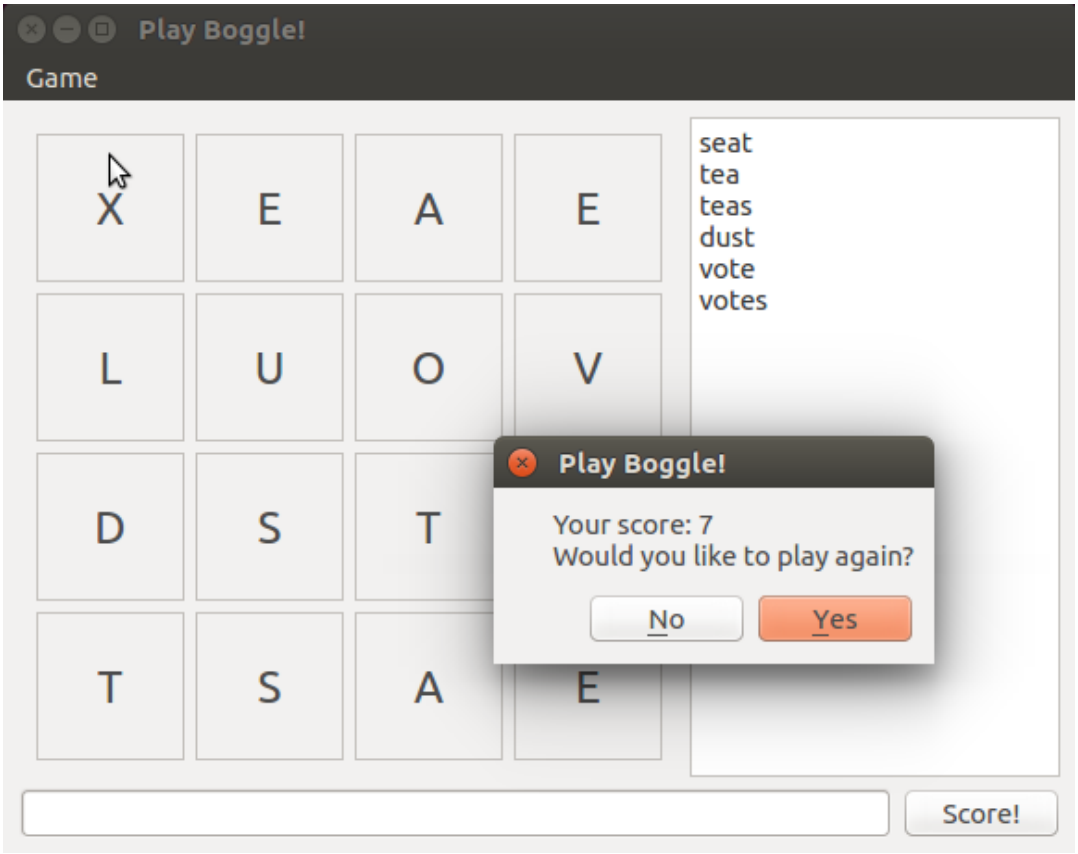
You may structure your GUI however you'd like as far as aesthetics and layout go. The following must be present, however:

- A descriptive window title (e.g. "Play Boggle!").
- The dice grid: a 4x4 display of the 16 die faces in which the user looks for words.
  Suggested widgets: QtWidgets.QLabel. Hint: notice that it inherits from QFrame, which has styling options.
- A text box displaying a list of words already entered.
  Suggested widget: QtWidgets.QTextEdit.
- A line editing field for submitting words. Pressing enter in this field should add the word to the text box.
  Suggested widget: QtWidgets.QLineEdit.
- A button to start the scoring process. Clicking this button should end the game and start the scoring process.
  Suggested widgets: QtWidgets.QPushButton
- A Game menu with options to start a new game, save a current game, or load an existing game.


Play Boggle!
Game
T A A R
H B R D
M R S P
N I A N
pan
bard
bath
Score!


Play Boggle!
Game
New
Save
Load
Y T N
Qu S Y T
H B P T
G I T X
bit
big
Score!

Ending the Game



When the game is ended by the user, a window should pop up (suggestion: QMessageBox), telling the user what their score is (no details required) and asking them if they'd like to play again. The default option should be "Yes". If they select to play again, all text should be cleared and the dice shuffled. If they select no, the application should close.

Saving and Loading Games

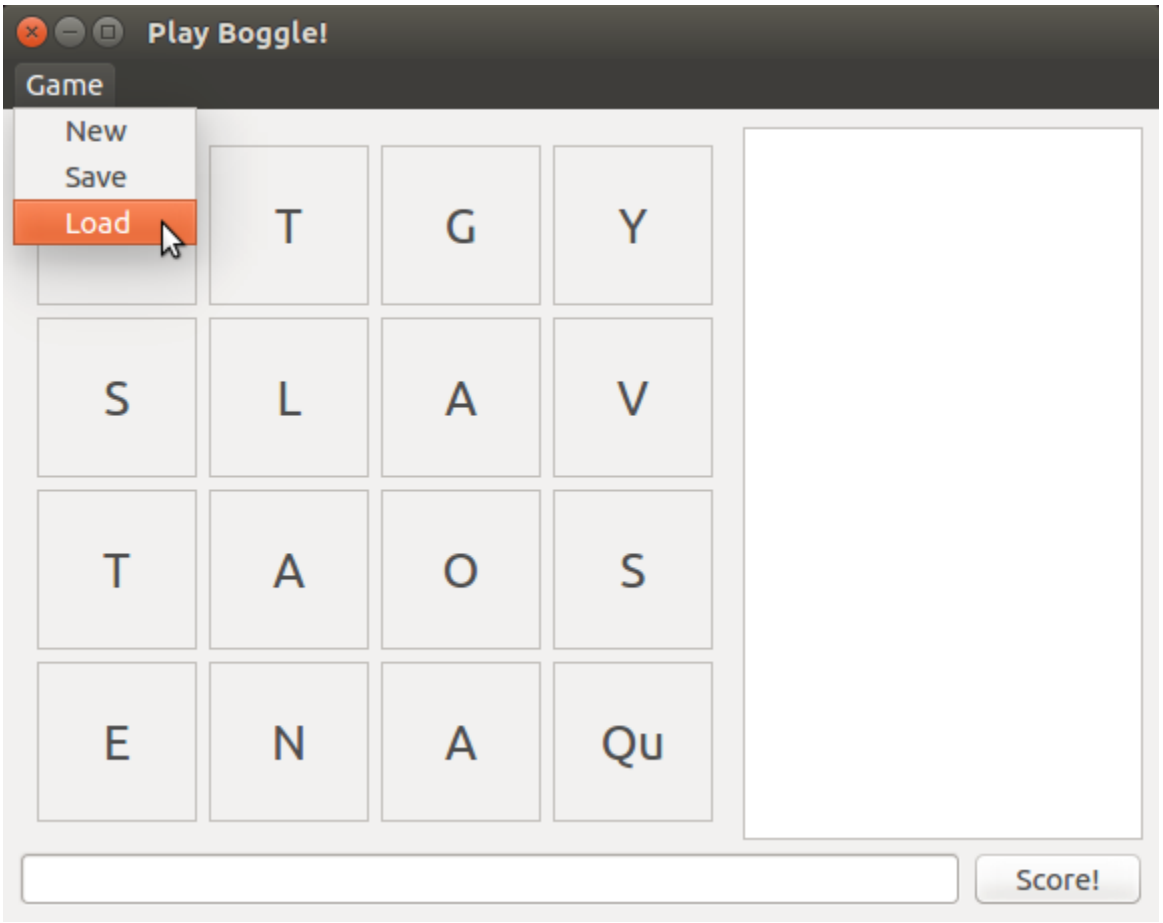
You will need to allow users to both save and load games. Saving and loading a game means:

1. Saving and restoring the dice configuration.
2. Saving and restoring the words already entered (i.e. the ones in the right text box).

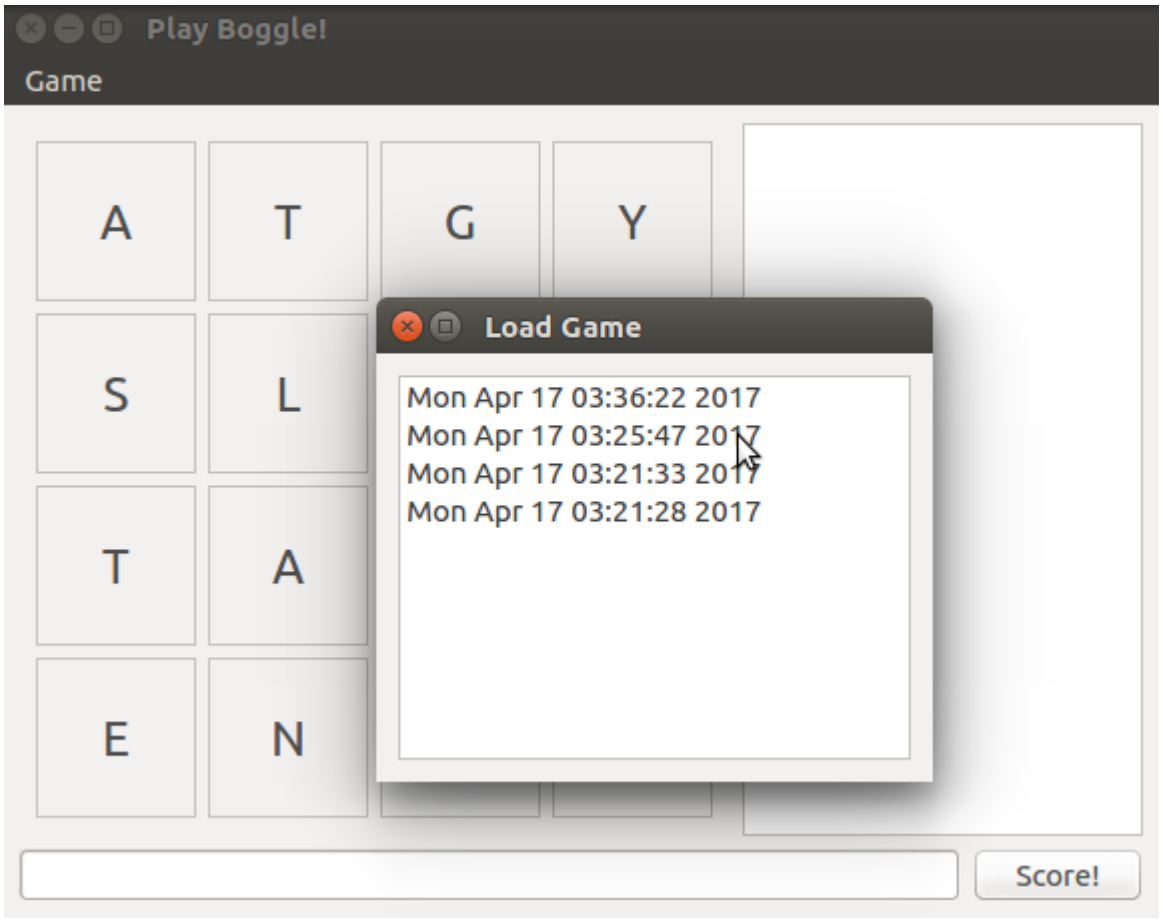
Saving and loading should work between application instances so you will need to use some persistent storage scheme (e.g. file data, sqlite database, shelve, etc). The choice of scheme is up to you – please just be sure to include any supplementary files if necessary (an sqlite file, for example).

Saving can be performed by selecting the Save action from the Game menu in the menu bar. When the user selects the Save action, the current state of the game should be recorded and saved, but the user will not see any further action take place. Saved games will be identifiable by a time stamp indicating when the game was saved.

Loading can be performed by choosing the Load action from the Game menu in the menu bar.



In this case, a dialog box should pop up (suggestion: QDialog and QListWidget may be of use here) listing the saved games available, identifiable by the time and date that they were saved. The saved games should be ordered, with the most recent game appearing first. The game which is selected by the user (by clicking on the saved game), should replace the current state of the game in the main window.



You are free to use whichever modules you'd like for this assignment as long as they are from the standard library or PyPI. If you find a module or code from elsewhere that you would like to use, you should come talk to me first. Your GUI can be in any style or layout you'd like as long as it has the required components.

In summary, I will be looking for the following requirements to be met while grading:

- Boggle game logic and scoring is correct.
- 5 listed GUI components (dice grid, word list, word input box, score button, and Game menu) are present and functional. You may add more if you'd like.
- User is allowed to start new game or load a saved game.
- User can end the game by pressing the right-hand side push button, displaying a pop up box with the score. The user should be allowed to quit the application or continue with a new game.
- User can opt to start a new game, save a current game, or load a saved game through the Game menu.
- When the user loads a new game, they can select from a list of saved games. Saved games are identified and ordered by the time and date they were saved.