Assignment 5 Program Grading Rubric

Your final program will be evaluated on the criteria below. Because of the diversity of programs being worked on, there cannot be a detailed rubric like there has been for labs and assignments.

Grading Criteria:

1. Program Agrees with Proposal: Final program agrees with proposed project (or Hangman alternate). If you talked with the instructor about any changes from the proposed project, these changes will be taken into account for this criteria.
2. Topic 1: Topic 1 used correctly and is important to the program
3. Topic 2/Library: Topic 2 or library used correctly and is important to the program
4. Documentation: Documentation provides clear description of program to someone who will never look at its source code.
    - If your program depends on any external libraries, include links to each library so that the grader can install the library.
    - If you think it's necessary, include how to run the program. If the grader cannot compile or run your program, this could delay the grading of your project or result in a lower grade on the project.
5. Code Readability: Source code is well-organized, including short and descriptive variable names, proper indentation and helpful comments
6. Good Programming Practices: This is a catch-all for any problems that do not fall into any of the above categories.

Grading Rubric:

Each criteria above will be ranked on a scale of 1 to 5. Since some criteria are more important than others, the ranking will be scaled by a weight (column W). The higher the weight, the more important the criteria.

The next page shows the grading rubric.

Category	W	1	2	3	4	5
Program Agrees with Proposal	15	Program does not match proposal (or Hangman) at all		Program generally matches proposed project, but there are some major changes		Program very closely matches proposal (or Hangman).
Topic 1	15	Not used in program		Implemented mostly correctly, but there are some flaws, bugs, or problems		Used correctly and shows evidence of understanding topic
Topic 2/Library	15	Not used in program		Implemented mostly correctly, but there are some flaws, bugs, or problems		Used correctly and shows evidence of understanding topic
Documentation	10	No documentation or instructions on running the program (if applicable)		Some documentation, but not adequate to describe program and how to run it (if needed)		Classes and methods properly documented with comments, program adequately described, and instructions on running program provided (if needed)
Code Readability	5	No comments, inconsistent indentation and spacing, and variable names that are usually undescriptive		Some comments (or too many comments), code not always indented properly, strange spacing		Code properly commented and indented/spaced correctly and consistently. Variable names are short and descriptive
Good Programming Practices	10	Program has many major and minor programming errors		Program contains some major errors and/or many minor errors		Program contains no major programming errors, and very few minor errors

Topics:
Below is a brief description of each of the topics that could be used for the project and what would constitute showing evidence of understanding the topic.

- Lists, Tuples, Sets, and/or Dictionaries – Holds collection of data
  - Can be any of the four listed above
  - Data stored/accessed/modified
- Files - Loading information from a file and saving information to a file
  - Reads and/or writes files
  - Program does not crash because file does not exist or because files are in inconsistent format
  - Reading and writing files follow consistent format.
- Exception Handling - Handling (and possibly intentionally creating) “exceptional” situations
  - Exception handling environment set up correctly, preventing program from crashing
  - Program handles any reasonable exception thrown
  - Program creates and throws exceptions when appropriate
- Functions – Writing your own functions to be used elsewhere in your program
  - At least three functions written
  - At least two functions takes multiple arguments
  - At least two functions return a non-`None` value
- Classes and Objects - Creating classes and using them
  - Create at least one class to represent a *thing*
  - All appropriate data encapsulated in class(es)
  - Class(es) provide standard and appropriate methods
  - Fields and methods created correctly
  - Creating objects of the class(es) created