Grade Book Program Overview

The grade book program that I created is for a teacher to use. When the user runs grade_book.py, the following menu appears with various options for the user to choose from:

```
Please enter a number corresponding with an item in the list below (press 0 to quit):
   1. To list all students.
   2. Add a student.
   3. List all categories.
   4. Add a category.
   5. Enter and calculate a student's final grade.
   6. List all students' grades.
```

If the user types in *1* to list all students and there are students in the students.txt file, each student in the file will be listed on a separate line where each line looks like the following: `10000, Rudi, Betts`. If there are no students in the students.txt file, an error message will appear notifying the user that no students are in the grade book and to use option "2" to add a new student.

If the user types in *2* to add a student, they will be prompted to enter a student_id, which must be a five-digit number and must not already exist; otherwise, an error will prompt the user to enter a valid or unassigned ID number. Next, the user will then be prompted to enter the student's first name and last name, which must only contain letters; otherwise, an error will prompt to re-enter the first and last name. If the student_id, first name, and last name are input successfully, they will be written to the student.txt file and will now display in the list of all students.

If the user types in *3* to list all categories, any categories within the categories.txt file will be listed on a separate line in the following format: `'100', 'Assignments', '35%'`. If there are no categories in the categories.txt file, an error message will advise the user that there are no categories in the grade book and to use option "4" to add a category.

If the user types in *4* to add a category, they will be prompted to enter a category_id. The category_id must be a three-digit number and must not already exist; otherwise, an error message will display stating the input was invalid or already being used. Next, the user will then be prompted to enter a name for the category, which must use only letters; otherwise, an error message will alert the user of an invalid entry. Finally, the user will then be prompted for a category weight. The category weight input will be checked to make sure that it is numeric and that it is greater than 0 but no more than 1, outputting an error message if these criteria are not adhered to. If the category_id, category_name, and category weight are input successfully, they will be written to the category.txt file and will now display in the list of all categories.

If the user types in *5*, they will be prompted to select a student by student_id from the grade book. If the student does not exist, the user will get an error stating the student does not exist. When the user types an ID number corresponding with a student in the grade book, they will get a result similar to the following: "`Your selected student is ['10000', 'Rudi Betts']`". The user will then be prompted to enter a grade for each category. They will see a prompt like the following: "`Please enter a grade in the Assignments category for Rudi Betts as a decimal between 0 and 1:`". If the user enters non-numeric input, an error will be output and have the user try again. When the user has entered a grade in the correct range, they will get a message that says something like the following:

```
The grades you have entered so far for Rudi Betts in the Assignments category are [0.9, 0.8].
Would you like to enter another grade in the Assignments category for Rudi Betts? Type 'y' for yes
or 'n' for no:
```

If the user types *y*, they will be re-prompted until they enter *n*, indicating that they don't want to enter another grade for the student in that category. If there is another category that the user has not entered grades for, they will be prompted with a similar set of messages. Once all grades have been entered, a message like the following prints with the student's name, categories with weights, entered grades, and calculated overall grade:

```
The grades for Rudi Betts:
Assignments (35% weight): ['90%', '93%', '87%']; Quizzes (15% weight): ['70%', '80%', '90%']
Total Grade: 87%
```

If the user types in *6* to list all students' grades and there are grades in the student_grades_list.txt file, each student and their grades in the file will be listed on a separate line, like the following:

```
Student Name, Grades in Each Category, Total / Final Grade
Rudi Betts, Assignments (35% weight): ['90%', '93%', '87%']; Quizzes (15% weight): ['70%', '80%', '90%'],
87%
```

If the user types in *0*, the program will terminate.

If the user types in any other number beside *0* through *6*, they will get the following message: "`ERROR: you have entered an invalid option. Please try again.`"