Grade Calculator II

psp-08-03

1 Overview

In this lab we will be building a grade calculator to help you determine your current grade using a weighted average. Your program will read grade data in from a text file, and then generate an html report based on that grade data. You will have to learn how to parse an arbitrary text-based data format, and calculate the necessary information based on the that data.

2 Learning Outcomes

By the end of this project students should be able to:

- read and write programs that use functions for task abstraction;
- utilize file input/ouput functionality;
- parse semi-complex text encodings;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students' problem solving process.

3 Pre-Lab Instructions

Do this part before you come to lab:

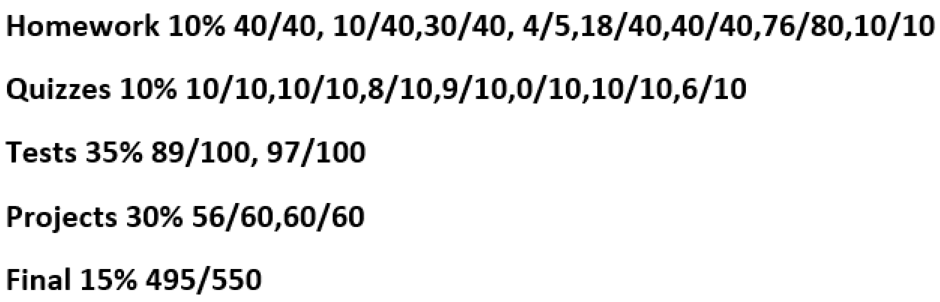
- Read Problem Space Chapter 8: Input and Output.
- Familiarize yourself with the string methods and the concept of slicing from Problem Space Chapter 3, in the Strings section.
- Using these string operations, write psudocode for how you would split a string in two, based on the middle most comma. (without loops, and you may assume an odd number of commas)
  ex: “1, d2, 3, 4, 64, 42, 3, dag” -> “1, d2, 3, 4” and “64, 42, 3, dag”

4 Lab Instructions

Do this part in lab:
Before you begin writing your own code take a moment to look at the example input text and output html provided with this lab.

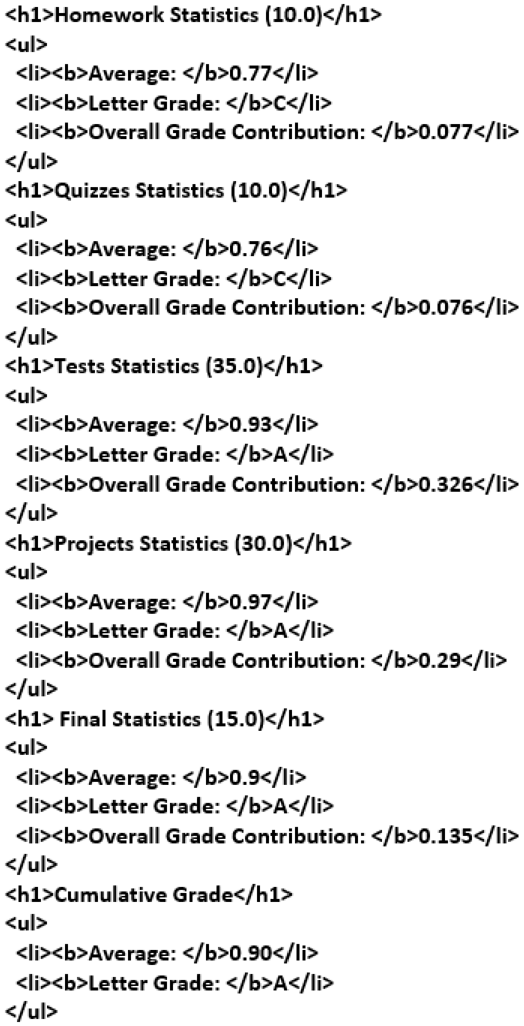
Important: you must make a program to read the grade file AS IS. You may not modify the input file to make it easier for you to parse.

The input file contains one line for every grade category. In this case, we have Homework, Quizzes, Tests, Projects, and a Final. Each line contains the category name, the percentage of the total grade for that category, and a list of assignment scores. These assignment scores are represented as points earned out of points possible, separated by a slash. Each of these three items is separated by a space. In the points list, each assignment is separated by a comma, and within each assignment the points earned are separated from the points possible by a slash. In addition, it is legal in this format to have spaces between the assigments. Open the input file and ensure you understand the layout before continuing.



```
Homework 10% 40/40, 10/40,30/40, 4/5,18/40,40/40,76/80,10/10

Quizzes 10% 10/10,10/10,8/10,9/10,0/10,10/10,6/10

Tests 35% 89/100, 97/100

Projects 30% 56/60,60/60

Final 15% 495/550
```

The output file is an HTML document containing a generated report. For each homework category there is a header, which shows the category name and the total weight, followed by an unordered list showing your average in that category, a letter grade based on the average, and an overall grade contribution, which is the category average multiplied by the category weight. After all of the categories, there is a final section for the cumulative grade, which contains the sum of all of the grade contributions, and a matching letter grade. For this assignment it is not necessary that you understand the HTML code in the document. It is only necessary that you can generate it inside your program, using the example output as a guide. When opened in a browser, the result should look like a very plain website. Open the output example in a text editior, and make sure you understand the layout.



```
<h1>Homework Statistics (10.0)</h1>
<ul>
  <li><b>Average: </b>0.77</li>
  <li><b>Letter Grade: </b>C</li>
  <li><b>Overall Grade Contribution: </b>0.077</li>
</ul>
<h1>Quizzes Statistics (10.0)</h1>
<ul>
  <li><b>Average: </b>0.76</li>
  <li><b>Letter Grade: </b>C</li>
  <li><b>Overall Grade Contribution: </b>0.076</li>
</ul>
<h1>Tests Statistics (35.0)</h1>
<ul>
  <li><b>Average: </b>0.93</li>
  <li><b>Letter Grade: </b>A</li>
  <li><b>Overall Grade Contribution: </b>0.326</li>
</ul>
<h1>Projects Statistics (30.0)</h1>
<ul>
  <li><b>Average: </b>0.97</li>
  <li><b>Letter Grade: </b>A</li>
  <li><b>Overall Grade Contribution: </b>0.29</li>
</ul>
<h1> Final Statistics (15.0)</h1>
<ul>
  <li><b>Average: </b>0.9</li>
  <li><b>Letter Grade: </b>A</li>
  <li><b>Overall Grade Contribution: </b>0.135</li>
</ul>
<h1>Cumulative Grade</h1>
<ul>
  <li><b>Average: </b>0.90</li>
  <li><b>Letter Grade: </b>A</li>
</ul>
```

Step 1:
Our first goal will be to create a function designed to read in input, and convert it into a data structure. Below are function specifications you are required to follow. Take note that the filehandle parameter is not a file path. This function should take a stream that has already been opened, and the function doesn't care where its data is coming from. Designed this way, if we wanted to read grade data from a different source, such as a web socket or other data stream, our function would still work. The return value of this function is a data structure of our own design, storing all of the categories with their names, weights and score lists.

Function: read_grade_data(filehandle)

Parameters:
filehandle - An open file handle ready for reading.
Return:
Data strucure of your design, containing all of the information in the file.

At the bottom of your file, add code to open a file handle and run read_grade data. Add some temporary print statements here to ensure your data is being stored correctly. When this code is graded, it will be tested against a different file, with a different number of categories, with different names and numbers of assignments, so ensure that your code doesn't rely on the categories used in the example.

Step 2:
Now we will build a function to use our grade data to build a report. Note that this function also requires an open file handle, not a file path. You may, if you wish, create additional functions to help you compute the values for the report, but it is not required.

Function: write_grade_report(filehandle, data)

Parameters:
filehandle - An open file handle ready for writing.
data - Your structure holding all grade data.
Return:
None

At the bottom of our file, add code to open a new file handle in write mode and call write_grade_report. Once your program is complete, it will print nothing to the screen, but will silently generate the HTML report.

When you have completed the lab run pep8 against your code until all formatting errors have been corrected and your code is PEP 8 compliant. See the Getting Started lab if you need instructions on running the program, or the pep8 documentation found here.

5 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical.

Your lab report should begin with a preamble that contains:

- The lab assignment number and name

- Your name(s)
- The date
- The lab section

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in **your** own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific funtions, classes or numeric requirements given to you, they should be represented in this bulleted list.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why.

3. Implementation and Testing

In the third section you should describe how you implemented your plan. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code (Submitted in BBLearn as a .py file)
- a screen shot of your running application / solution
- results from testing

4. Reflection

In the last section you should reflect on the project. Consider different things you could have done to make your solution better. This might include code organization improvements, design improvements, etc.

You should also ask yourself what were the key insights or features of your solution? Were there alternative approaches or techniques you could have employed? How would these alternatives have impacted a different solution?

5. Partner Rating

Every assignment you are required to rate your partner with a score -1, 0 or 1. This should be submitted in the comment section of the BBLearn submission, and not in the report document. You do not have to tell your partner the rating you assign them. A rating of 1 indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 0 indcates that your partner met the class expectations of them. Rating your partner at -1 means that they refused contribute to the project, failed to put in a resonable effort or actively blocked you from participating. If a student recieves three ratings of -1 they must attend a mandatory meeting with the instructor to dicuss the situation, and recieving additional -1 ratings beyond that, the student risks losing a letter grade, or even failing the course.

Colophon

This project was developed by Dr. James Dean Palmer of Northern Arizona University. Except as otherwise noted, the content of this document is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.