Jasque Saydyk and Oshan Vinod Wijesooriya

Professor Vanderberg, Swenson, and Trice

CS 126L Section 2

15 February 2017

Lab 04 - Grade Calculator

1. Problem Statement

This program is a simple grade calculator that will test our ability to work with lists and create functions. This program requires the creation of three lists filled with data, then multiple functions that manipulate those lists. The first function calculates the percentage of possible points for a category, Homework, Quiz, or Test. The second function calculates a letter grade based on a given percentage. The last required function calculates the weight of a category to the overall course grade.
There are also some extra credit functions that will be implemented.
The following were the requirements of the program:

- Six lists were set up to contain the information given , with numerators and denominators in different lists.
- An variable had to be setup to contain the weights for each category : 20 % for quizzes , Homework 20% , Tests 60%
- Functions had to be defined for each specific task :
    - Percentage_per_catogory - calculates the sum of numerator and denominator to find the percentage of each category
    - Letter grade assigning - Return a grade equivalent for given percentage . 90% above = A , 80%-90% = B , 70% to 60% = C , 50% to 40 % = D
    - Weighted Score - returns a decimal value that states the amount each category contributes to the final grade.

- Print the percentages and the letter grade for each category
- Meet pep8 compliance

2. Planning

We planned to break the Numerator and Denominator of the given fraction to two separate list to help in calculations. Functions were defined containing conditional statements and basic mathematics pre-defined in the problem statement. Inbuild python functions were used to work with the lists Ex. Sum() function to add all elements in list. The Planning stage required minimal effort as the problem statement provided exact instructions to each requirement.

3. Implementation and Testing

The implementation of this program was incredibly straightforward, as most of the plan was written in the lab report to begin with. Most of the functions didn't require anymore than one to three lines of code, making them incredibly simple to write.

The extra credit, however, was a totally different ballgame and was fun to complete. The to_percent method required the creation of a list of percents using only recursion. We went through several iterations when completing this method, first relying on a global list to store all of the newly generated percents in it, which became an issue when the method had to be used for more than one time. We then tried using an initialization if statement in the method, with a global switch, which also failed. It was at this time I recalled my experience with Scheme and realized I should just make another function that had the parameter list that I needed to do the function.

As for the median method, it simply sorts the list given to it, then takes the biggest and smallest percentage from it, adds them together and divides them by two, then returns it. I did have an issue of getting an output of none, but that was related to a mistake I made in the to_percent method, where I failed to return the method-call of to_percent_full, thus the method would return nothing.

```
Homework Grade: 74%, Grade: C, Median: 50%
    Quiz Grade: 87%, Grade: B, Median: 60%
    Test Grade: 97%, Grade: A, Median: 97%
   Final Score: 90%, Grade: A
>>>
```

4. Reflection

We are pretty satisfied with this project, given its intended size. If there is one improvement to be made, it's for the math to actually be proven out on paper that it actually works. Although designing code to meet a test case is fine, having a solid proof that the math is actually correct is better, so you can truly identify the edge cases and know that the semantics of the code truly is correct. The last thing is that the sort method may need to be changed if it actually faces big lists, but that isn't a concern at the moment.