Lecture 21

File IO and Exceptions Review and an Introduction to Dictionaries

24 Germinal, Year CCXXX

***Song of the day*: *Juban District* by Ginger Root (2021).**

Review

When we download your exam grades from Gradescope, they come out in the following format:

```
Course Name,section_name,Total Score,Max Points,Status,Submission ID,Submission T
INTRO TO PROGRAMMING & PROBLEM SOLVING, Section LB7,95.0,100.0,Graded,115742079,2
INTRO TO PROGRAMMING & PROBLEM SOLVING, Section LB1,96.0,100.0,Graded,115742080,2

INTRO TO PROGRAMMING & PROBLEM SOLVING, Section LB3,99.0,100.0,Graded,115742081,2
INTRO TO PROGRAMMING & PROBLEM SOLVING, Section LB7,83.0,100.0,Graded,115742082,2
```

Figure 1: The first 5 lines of **grades.csv**, containing the midterm 1 grades. Names and IDs have been removed for obvious privacy reasons.

While Gradescope does us the favour of calculating the average (a.k.a. the mean), the median, and the standard deviation, these operations are simple enough for us to do them ourselves. We'll use Python's `statistics` module for that, since it contains the methods `mean` , `median` , and `stdev` . All three of these methods accept a list of numbers and returns another number.

What we'll be programming is a few functions that will extract the grades from the file, calculate the statistics, and print them into an output "report" file.

The `extract_student_grade()` Function

> *sig*: `str => float`

The best way to begin a problem like this is to consider **the operation that we will have to perform on a single line** of the file. For example, if we have the following line:

```
INTRO TO PROGRAMMING & PROBLEM SOLVING, Section LB3,64.0,100.0,Graded,115742944,2(
```

Figure 2: A single line from **grades.csv**. See **figure 1** for the header.

We need to write a function that will extract that `64.0` , since that is the student's grade according to the header in **figure 1**. If we can write a function that can extract a grade from one line, we can extract a grade from every single line.

Write a function called `extract_student_grade()` which will accept a line (an `str` object) similar to that in **figure 2** and will return the grade contained within it.

You may assume that the line will always contain the same amount of commas `,` , and the order of the data points will always be the same.

Sample usage:

```
line = "INTRO TO PROGRAMMING & PROBLEM SOLVING, Section LB3,64.0,100.0,Graded,115
grade = extract_student_grade(line)

print(line)
```

Output:

```
64.0
```

Do not move on until you are sure that you understand this part. If you don't, raise your hand and ask for any clarification.

The `get_list_of_grades()` Function

> *sig*: `str => list(float)`

Now that we have a function that can extract a single grade from a single line, let's write a function that can iterate through every single line in the file, extract its grade, save it in a list, and return that list.

Write a function `get_list_of_grades()` that will accept one string parameter, the name and path of the file seen in **figure 1**. If the file contained inside this parameter exists, then the function will return a list of numbers, i.e a list of all the grades inside that file. If the file doesn't exist, then the function will return an empty list.

Sample usage:

```
grades = get_list_of_grades("grades.csv")
print(grades)
```

Output:

```
[95.0, 96.0, 99.0, 83.0]
```

Note: This output assumes that only the four grades shown in **figure 1** exist in the file. The actual file could contain any number of lines.

Again, do not move on until you are sure that you understand this part. Ask questions.

The `create_stats_report()` Function

> *sig*: `str, str => None`

Finally, write a function that accept two string parameters, both filenames: the filename of the original CSV file holding the grades, and the name of the file onto which we will write the grades' statistics.

The function will make use of `get_list_of_grades()` to extract a list of grades from the original grades CSV file. Once we have this list, we will use the `statistics` module's `mean`, `median`, and `stdev` methods to caculate the average, median, and standard deviation of this list, respectively (all three methods accept a list of numerical values as arguments; see **appendix** for sample uses of the `statistics` module if you want).

Once you have these three values, open a file using the function's second parameter as a filepath, and write onto it the following lines:

```
Amount,Average,Standard Deviation,Median
AMOUNT_VALUE,AVERAGE_VALUE,STD_DEV_VALUE,MEDIAN_VALUE
```

Note: `AMOUNT_VALUE` represents the number of grades in the original file, `AVERAGE_VALUE` the calculated average, `STD_DEV_VALUE` the calculated standard deviation, and `MEDIAN_VALUE` the calculated median value.

Sample execution:

```
input_filepath = "grades.csv"
output_filepath = "report.csv"
create_stats_report(input_filepath, output_filepath)
```

The contents of `report.csv` will be:

```
Amount,Average,Standard Deviation,Median
193,73.75,21.53,78.0
```

Appendix

Sample use of `statistics` module:

```
import statistics

grades = [100.0, 95.0, 50.0, 79.9, 60.4, 99.1, 80.5]

average = statistics.mean(grades)
median = statistics.median(grades)
std_dev = statistics.stdev(grades)

print("Average: {}\nMedian: {}\nStandard Deviation: {}".format(average, median, s
```

Output:

```
Average: 80.7
Median: 80.5
Standard Deviation: 19.45421976504498
```