CMSC 201, Section 24
Spring 2020

Project 1 – Descriptive Statistics

Due Date:

Design Document: Monday, March 30, 2020 Before Midnight
Project: Monday, April 6, 2020 Before Midnight

Value: 80 points

Collaboration: For Project 1, **collaboration is not allowed** – you must work individually. You may still see your TA and come to office hours for help, but you may not work with any other CMSC 201 students.

Make sure that you have a complete file header comment at the top of your file and that all information is correctly filled in.

```
# File:    FILENAME.py  (OR: FILENAME.txt)
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_USERNAME@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

Project 1 is the first assignment where you've had to turn in a "design document" in addition to the actual code. The design document is intended to help you practice deliberately constructing your program and how it will work, rather than coding as you go along or starting without a plan.

Instructions

For this project, you will be creating a single program, but one that is bigger in size and complexity than any individual homework problem. This assignment will focus on using functions to break a large task down into smaller parts.

You are ***required*** *to follow the design details provided to you*! You may add additional functions, but you must implement all the specified functions as described in the Design Information section below.

At the end, your Project 1 file must run without any Python errors.
It must also be called proj1.py (case sensitive).

Additional Instructions – Creating the proj1 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

You should create a directory in which to store your Project 1 files. We recommend calling it **`proj1`** and creating it inside a newly-created directory called **`Projects`** inside your **`201`** directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1.

Objective

The goal of Project 1 is to demonstrate a mastery of:

- Program design and the use of functions
- Reading from and writing to files
- Using and manipulating lists

In addition, you will demonstrate an understanding of the other control structures and language artifacts that we have used so far this semester.

This includes strings, floats and ints; casting variables from one type to another; and using basic arithmetic operators.

Task

This project deals with some basic descriptive statistics; specifically, mean, median, mode, and standard deviation. We have covered these in class on multiple occasions.

You will be given grades from four semesters of a computer science class, like CMSC 201. The data are not real – we have to respect student privacy!! – but they are very similar to the actual data. We want to compare how grades vary from semester to semester, based on methods used and material provided.

You will be given four data files. Your task is to read in each file, and compute the median, mean, mode and standard deviation of the grade sets in each of the four files. You will then create an output data file, where you will print these values for each of the four semesters.

Test Data

You are provided with test data files containing course grades as integer values, all of which are greater than or equal to zero. The files are what are called "comma-separated value" or ".csv" files. That is, they consist of values that are separated from each other, or *delimited*, by commas.

The data files are to be downloaded from Professor Arsenault's public directory.
Here are the Linux commands to copy the files from Professor Arsenault's public directory to your **`proj1`** directory. Don't forget the dot (.) at the end of each command! (It specifies to Linux to copy the file to your current directory.)

```
cp /afs/umbc.edu/users/s/m/smitchel/pub/cs201/proj1.py .
cp /afs/umbc.edu/users/s/m/smitchel/pub/cs201/S18_Sect18.csv .
cp /afs/umbc.edu/users/s/m/smitchel/pub/cs201/F18_Sect22.csv .
```

```
cp /afs/umbc.edu/users/s/m/smitchel/pub/cs201/F18_Sect36.csv .
cp /afs/umbc.edu/users/s/m/smitchel/pub/cs201/Combined.csv .
```

To look at the contents of any of these files after downloading them to your **`proj1`** directory, use the Linux **`cat`** command, as shown in the example below.

```
cat S18_Section18.csv
```

Coding Standards

Prior to this assignment, you should re-read the Coding Standards, available on the CMSC 201 course website.

For now, you should pay special attention to the sections about:

- Comments
  - ***File header comments***
  - ***Function header comments***
  - Note that "Input" and "Output" in the function header comment do NOT mean what is shown on the screen with **`print()`** or what is gotten from the user with **`input()`**.
    They refer to the **parameters** taken in, and the **return value**.
    (Both "Input" and/or "Output" may be None, if appropriate.)
- Constants
  - You must use constants instead of "magic numbers" or strings!!!
- Make sure to read the last page of the Coding Standards document, which prohibits the use of certain tools and Python keywords. (Note that you can use the built-in **`sqrt()`** function and **`sort()`** method – only – for this project.)

Additional Specifications

For this assignment, **you must follow the design overview** in this document.

For this assignment, you may assume that the data are valid. All integers are positive and meaningful; there will be no textual data where there are supposed to be numbers.

The project is worth a total of 80 points. Of those points, 10 will be based on your design document, 10 will be based on following the coding standards, and 60 will be based on the functionality and completeness of your project.

Design Document

The design document will make sure that you begin thinking about your project in a serious way early. This will give you experience doing design work and help you gauge the number of hours you'll need to set aside to complete the project. **Your design document must be called design1.txt.**

For Project 1, the design overview is presented in the Design Information section below, and you are **required to follow it**. For future projects, you will be creating the design entirely on your own and may choose to design it however you like.

Your design document must have four separate parts:

1. A file header, similar to those for your homework assignments
2. Constants
    a. A list of all the constants your program will need, including a short comment describing what each "group" of constants is for
3. Function headers
    a. A complete function header comment for each function
4. Pseudocode for main()
    a. A brief but descriptive breakdown of the steps your main() function will take to completely solve the problem; note function calls under relevant comments (if applicable).

Although you will be presented with a design overview, you must still create the function headers, and the pseudocode for **`main()`** on your own.

A start for your design is provided on Blackboard under "Assignments." Follow the layout and format of that document. You can also copy it using:

```
cp /afs/umbc.edu/users/s/m/smitchel/pub/cs201/design1.txt .
```

Don't forget the dot (.) at the end of the command!

NOTE: The sample design provided is not complete and is missing many function header comments. You need to add them! You may also add constants if you find it necessary.

Your **`design1.txt`** file will be compared to your **`proj1.py`** file. Minor changes to the design are allowed. A minor change might be the addition of another function or a small change to **`main()`**.

Major changes between the design and your project will cause a loss of points. This would indicate that you didn't give sufficient thought to your design.

(If your submitted design doesn't work, it is generally better to lose the points on the design and to have a functional program, rather than turning in a broken program that follows the design. The ultimate decision is up to you.)

Sample Output

The sample output is available as a separate file under "Assignments" on Blackboard and is called **sampleRuns_Project1_S19.pdf**. Look at the sample output before reading the notes below. (The format of your does not have to match the sample output exactly, but it should be similar and neat.)

Other Hints and Information

- Writing up and testing each of your functions individually will make your life much, much easier.

- Commenting your code as you write it or even before you write it (think pseudocode) will make the process much simpler and will also allow your TA and instructor to help you more effectively.

- You will be using the pre-defined Python **`sqrt()`** function and **`sort()`** method for this project.

- You may also want to temporarily use the Python statistical functions **`mean(), median(), mode(), and stdev()`** for debugging purposes. That way, you'll be able to compare the results that Python gives you with your own corresponding functions. Just be sure to remove these function calls before turning in your project. Here's an example of how to use one of these functions:

```
print("Python mean = " + str(statistics.mean(values)))
```

Design Information

You are required to implement and use at least the following 12 functions for Project 1, in addition to **`main()`**. You may implement more functions if you think them necessary, but the functions below must be implemented **and used**. The information for each function is given below.

Helper Functions

- **`def mean(values)`**
  - Computes the mean of a set of numeric values
    - **`values`** is a list of numeric (integer, floating point, or mixed) values
  - Returns the mean of the values
  - Preconditions:
    - **`values`** contains >= 1 value
  - Note that this function does not display the mean. It simply computes and returns it.

- **`def median(values)`**
  - Finds the median of a set of numeric values
    - **`values`** is list of numeric (integer, floating point, or mixed) values
  - Returns the median of the values
  - Preconditions:
    - **`values`** contains >= 2 values
  - Note that this function does not display the median. It simply finds and returns it.
  - You will need to sort the list of values from smallest to largest to find the median. Use the Python built-in **`sort()`** method, as you did in Homework 5, Part 2. Don't forget to make a local copy of **`values`** before sorting!

- **`def mode(values)`**
  - Finds the mode of a set of integer values
    - **`values`** is a list of integer values
  - Returns the mode of the values

- Preconditions
  - **`values`** contains integer values, all >= 0
  - **`values`** contains >= 1 value
  - The list of values is guaranteed to form a unimodal distribution
- Note that this function does not display the mode. It simply finds and returns it.
- You will need to sort the list of values from smallest to largest to find the mode. Use the Python built-in **`sort()`** method, as you did in Homework 5, Part 2. Don't forget to make a local copy of **`values`** before sorting!

- **`def stdev(values)`**
  - Computes the standard deviation of a set of numeric values
    - **`values`** is a list of numeric (integer, floating point, or mixed) values
  - Returns the floating point standard deviation of the values
  - Preconditions:
    - **`values`** contains >= 2 values
  - Note that this function does not display the standard deviation. It simply computes and returns it.
  - The standard deviation computation requires the use of a square root. Use the Python built-in **`sqrt()`** method. Use it as follows:

    `math.sqrt(`*`value`*`)` where ***`value`*** = a numeric value
  - ❖ **HINT**: Take advantage of the **`mean()`** function that you have written!

General Functions

These are the "heavy lifters" of the program, and do the majority of the work, and are almost always called from **`main()`**. They often call other functions, often more than once. You must implement and use all the general functions given below.

- **`def read_file(filename)`**

- Reads in the contents of a .csv (comma-separated value) file of values
- Takes in a filename as a string
- Returns the file contents, except for the first two rows, as a single string
- The first two lines of every data file are HEADER rows and do not contain student scores. This function should read in these two lines and essentially throw them away, as they are not needed.
- The rest of the file contains: student name, student id number, and six scores – three for projects and three for tests.

- **`def create_lists (file_contents)`**
  - Takes the string representing the actual data from a file
  - Returns two lists: one list containing the sum of the three project grades for each student; and the other containing the sum of the three test grades for each student

- **`def calc_values (project_list, exam_list)`**
  - Calculate the mean, median, mode and standard deviation of each of the project scores and the exam scores
  - Use the helper functions mean, median, mode and stdev to do this
  - Create a list containing in this order: project mean, project median, project mode, project standard deviation, test mean, test median, test mode, test standard deviation
  - Join the elements of that list with a tab to create a single string, and return that string. That is, you'll have a statement that looks something like

    result_string = "\t".join(results_list)

- **`def write_file (list)`**
  - Takes as input the list of strings containing the values from calc_values
  - Writes a single text file containing explanatory header information (see the sample on Blackboard), and then the calculated values for each of the four classes

Main Program

Start your main program by creating a list containing the four data file names Then execute a for each loop, going through the four files in that list. In that loop, you process a file by first calling the read_file function; then passing the resulting string to the create_lists function. Then you call the calc_values function.

When the loop has finished, you call the "write_file" function to print your output.

The main program should contain ***minimal*** code!! There should be no more code than is necessary to make the function calls described above.

Submitting

Once your **`proj1.py`** or **`design1.txt`** file is complete, it is time to turn it in with the **`submit`** command. (You may also turn the design and project in multiple times as you reach new milestones. To do so, run **`submit`** as normal.)

To submit your design file, use the command:

```
linux1[4]% submit cmsc201s PROJ1_DESIGN design1.txt
Submitting design1.txt...OK
linux1[5]%
```

To submit your project file, use the command:

```
linux1[4]% submit cmsc201s PROJ1 proj1.py
Submitting proj1.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your project and its design

correctly, since **an empty file will result in a grade of zero for this assignment.**