CS 360 Group Project: Python Contact Manager **DUE: November 6, 2016, 11:59pm**
DUE: Oct 30, 6pm Issues

Each group has been given a GitHub repository cs360f16/PythonContactManager-Group#

You are required to use PyDoc comments, good open source principles, coding standards, and good software development techniques on this project.

This is a simple contact manager like the first group project. None of the code has been implemented except for the call to main() and the display of the menu to the user. You must provide all of the code for this project.

ContactManger.py - contains main() and the function to display the menu.

ContactData.py - DO NOT ALTER THIS FILE - contains a List (DATA, a Global Variable) which contains all of the contact data. While this is a global variable, do not use DATA as such. DATA is passed to main() and from there must be passed to every function that needs it.

You may need to correct the formatting in the existing .py files to match our coding standards.

You are required to implement menu items 0-5 (0-4 for 2 person teams). Menu item 6 is a bonus.

You must write a Python file (also called a **module**) per functionality unit and include that module in ContactManager.py via the **from** command. This will help to reduce the number of merge conflicts that arise. *The minimum number of modules is three*.

You are required to use PyDoc comments, good open source principles, coding standards, and good software development techniques on this project. Add DocTest tests where you can. Note that DocTest tests the stdout (print()) plus the return value of a function!
https://docs.python.org/3/library/doctest.html

The existing repositories do not have any Issues, and **you are required** to add Issues. Adding Issues and then assigning these Issues to team members will help you stay on task and track everyone's work. The Issuses must clearly describe the functionality needed. For example: Will you provide partial matching for names or only exact matching?

Pull requests **must** receive a proper review from someone other than the submitter before the Pull request is merged into the group repository.

Your group must decide on a coding tool, Eclipse, Geany, or PyCharm and build the appropriate .gitignore file. Review the existing .gitignore file to see all the items you won't be putting onto GitHub!

A grading rubric is on the next page.

Each team member **must**: Resolve an issue, make a pull request, merge someone else's pull request.

If your team is having problems, let me know immediately.

GROUP: _____ / 50 points

Coding Standards Follow PEP-008 standards (2 pts) comments (PyDoc) (2 pts) Indent: 4 spaces (1 pts) Use DocTest (3 pts) No misuse of the variable DATA (1 pts) Use of modules (minimum of 3) (3 pts) Pythonic code / Design (non-repetitive, good use of Lists, and other Python structures) (5 pts)	
Open Source Principles Issues Created and Assigned (2 pts) README.md updated to explain usage and development process (How do I make a feature request or report a bug? I need clear instructions). (2 pts) Pull Requests are made from branches. (2 pts) Pull Requests Each team member **produced. reviewed, resolved** (9 pts) Team member 1_____|_____|_____ Team member 2 _____|_____|_____ Team member 3_____|_____|_____	
Menu item 0 (3 pts) All entries are displayed nicely to the screen	
Menu item 1 (3 pts) Search shows all entries for a given office location These entries are displayed nicely	
Menu item 2 (3 pts) An entry can be added	
Menu item 3 (3 pts) An entry can be deleted	
Menu item 4 (3 pts) Search shows all entries for a given First Name These entries are displayed nicely	
Menu item 5 (3 pts) Search shows the First Entry for a given email These entries are displayed nicely	
Menu item 6 BONUS (3 pts) The data can be written to a file in CSV format (comma separated values)	