Program Design

March 23, 2020

Administrative Notes

Discussion about Homework 5

Discussion about Project 1

Schedule for the next couple of weeks:

- March 23, program design
- March 25, 2D lists and list slicing
- March 30, any remaining list issues and dictionaries
- April 1, dictionaries and catching up on topics - anything we've missed
- April 6, review for exam
- April 8, Exam 2

We'll start recursion *after* exam 2

Lab 6 - this week

Lab 6 will be posted to GitHub as soon as this lecture is over

- You may work on it whenever you want
- Your TAs will be available on Discord during your regularly scheduled lab times tomorrow to help you through it - Anna from 11:30 - 12:20; Jordon from 4:00 - 4:50
- You must submit the completed lab before midnight on Thursday, March 26

Designing Programs “properly”

The goals are:

- Get the program correct - it does what it’s supposed to do and doesn’t do anything else
  - Unpleasant side effects are a thing - e.g. accidentally deleting your data files
- Make the program understandable by yourself, and others
  - People who have to review it
  - People who have to support it later on
-

CMSC 201 teachers are harsh

We force you to write code properly

- Modular
- Well documented
- Using well-understood features
- Tested
- Evaluated (i.e., graded)

Modular code

Modular code allows parts to be removed and/or replaced without disturbing other parts of the program

- Think of your code as a gigantic Jenga game

Modules should be portions of code that perform an independent task

- Make them functions
- Swap out new functions, or change the contents of the functions with little or no changes to the rest of your project.
    - E.g., I could have you swap the function in Project 1 that reads a .csv file with one that reads a .txt file. As long as it returns a string, there should be no changes to the rest of the project

An example of modularity

- We’ll work through the “medal table” program and show how to make it modular