MATH 260: Python programming in math
Summer 2020 syllabus

Instructor/Office: Jeffrey Wong (Physics 029B/online by appt.)

Times/location: MTWThF 12:30-1:45, online (Zoom)

Course Website: Piazza (main site) and Course Sakai site (only for grades)

Textbook: No required text. Recommended: TBA

Course description: This is an introductory programming course in python that will provide a foundational background for programming in a mathematical setting. Students learn the basics of object oriented programming: memory storage and variable scoping, recursion, objects and classes, and basic data structures. A variety of numerical methods are introduced, with a focus on practical implementation, through a series of modules covering a variety of subjects such as physical modeling, genetics, and optimization.

Prerequisites: Students should have taken at least linear algebra (Math 216, 218 or 221), while no programming background is required. Not open to students who have taken CS 201.

Technology: Access to the following (free) resources is required: Zoom (for lectures), github (to share and turn in work), Python 3.x and an editor (e.g. a text editor or an IDE like PyCharm), LaTeX (for some homework and project writing). Information for getting started will be provided.

Grading: Course grades will be based on the following components:

- Final Project (30%)
- Homework: Lowest score dropped (45%)
- Midterm Exam: One, covering programming (20%)
- Participation (5%): this includes active participation in activities such as posting questions/answers on Piazza, attending office hours and final project discussions with the instructor.

Note that participation in synchronous class meetings is strongly encouraged but, if missed, is not penalized in the grade.

Exams: There is one midterm exam that covers programming fundamentals and **no final exam**. The midterm will be 'take-home' and open notes (with some restrictions), to be taken during a subset of a continuous three hour period of the student's choice.

Final project: Details will be provided around mid-way through the course. Students select a topic involving some application of computational mathematics. The project will involve learning and implementing algorithms to solve that problem and a written report.

Homework:

- Homework will be assigned approximately twice each week; typically due on the date of the next assignment.
- Homework should be turned in by the deadline to ensure full credit. If late, the homework should still be turned in for feedback (and partial credit).
- Working and studying in groups is strongly encouraged, as well as discussing code (writing code to be read by others is an excellent way to improve computational skills!). However, the final product - solutions and code - should be your own. Further guidelines for code and computational results will be provided throughout the course.
- Solutions should be complete arguments and code should be functional 'out of the box' (i.e. it runs and produces the required outputs). When explanations are asked for, they may reference code but should be complete and easy to follow (it should not be necessary to 'dig through' the code to get to solutions).
- Typing solutions (in LaTeX) is suggested; if handwritten, make sure solutions are readable when scanned (a high quality phone camera with good lighting should be sufficient). Please contact me if you have logistical issues with turning in work.

Ethics: Students are expected to follow the Duke Community Standard. If a student is found responsible for academic dishonesty through the Office of Student Conduct, the student will receive a score of zero for that assignment. If a student's admitted academic dishonesty is resolved directly through a faculty-student resolution agreement approved by the Office of Student Conduct, the terms of that agreement will dictate the grading response to the assignment at issue.

Overview of topics

The goal is to cover most if not all of the topics in the main areas listed. The details may change depending on time and interest.

- **Part I: Programming fundamentals:**
  - The basics (language function; syntax)
  - Control structures (loops, ifs, etc.)
  - Data types, memory management basics
  - Functional programming and recursion
  - Input/Output, plotting
  - Object oriented basics: classes, objects, OO programming ideas
  - Data structures (trees, arrays, etc.)
- **Part II: Modules in computational mathematics**
  - Each module will cover a topic, including: context, the application and the practical problem to solve, numerical methods and implementation.
  - The goal is to illustrate how the programming principles are applied, and to learn to code through meaningful example (plus, to learn some numerical methods and interesting mathematics along the way).
  - Modules will span approximately two to four class periods.
  - Possible topics include: discrete population models, solving ODEs numerically, Markov chains, integration, computer graphics (splines, Bezier curves), image processing/compression, simulated annealing,