Astro/Earth - 119 Spring 2019
Introduction to Scientific Computing in Python

Instructor:
Thomas Goebel, tgoebel@ucsc.edu , office hours: Mon & Wed. 11 a.m. to noon (or by appointment), Earth & Marine Sci. C366

TA: Huazhi Ge, hge2@ucsc.edu, office hours: ???, Earth & Marine Sci. ???

Lectures: Mon & Wed, 8 - 9:35 a.m., 109BE

Lab Sections: Thu 6 - 8 p.m., 109BE

Prerequisite(s): Mathematics 11A or 19A or 20A or Applied Mathematics or Statistics 15A.

Course Description: The ability to quantitatively analyze data using a scientific programming language (R, perl, python, matlab... something beyond MS excel) is an essential skill for any engineer or scientists. Being able to write code that synthesizes your knowledge of mathematics and physics in a slick, well-structured program is considered an art by some, but certainly useful for many. Moreover, it creates ample opportunities to land a job maybe even at Alphabet or Netflix, if you feel like it.

This course provides a rough overview into data analysis and modeling with python, starting with what is a programming language and ending with solving PDEs. The aim of the class is to introduce students to a wide set of tools that can be used to solve problems in pure and applied sciences. The introductory character of the class limits the amount of time spent an each topic. This should hopefully not lead to frustration but rather encourage to dig deeper in areas of your own personal interest. A large part of the course will be a final project which encourages exactly that: find a problem your interested in and solve it. The class is designed to leave enough room for you to develop critical thinking and analysis skills within the framework of the python programming language.

From Learn Python the Hard Way by Zed A. Shaw: *"Programming as a profession is only moderately interesting. It can be a good job, but you could make about the same money and be happier running a fast food joint. You're much better off using code as your secret weapon in another profession. People who can code in the world of technology companies are a dime a dozen and get no respect. People who can code in biology, medicine, government, sociology, physics, history, and mathematics are respected and can do amazing things to advance those disciplines."*

Course Objectives:

At the completion of this course, students will be able to:

1. Create scripts and functions for quantitative data analyses
2. Invert for model parameters and fit functions to observed data
3. Create figures and animations
4. Construct numerical models using finite-difference methods

Text(s) and Resources

- *Programming for Computations - Python: A Gentle Introduction to Numerical Simulations with Python*, 1st Edition, by Svein Linge and Hans Langtangen.
  http://hplgit.github.io/Programming-for-Computations/pub/p4c/p4c_Python.pdf
- Think Python, 2nd Edition, by Allen B. Downey
- Google's free python class https://developers.google.com/edu/python/?csw=1
- Python style guide https://docs.python-guide.org/writing/style/

Software Requirements

Ideally, students will bring their on laptops to complete the assignments for this course but desktop computers will be available during class hours. Install the Anaconda python package on your computer (make sure to install **Anaconda2 for python 2.7**, newer python versions such as 3.0 may not be compatible (change in syntax etc.) and you may have trouble running some of the examples).
Anaconda: https://www.anaconda.com/distribution/
PyCharm: https://www.jetbrains.com/pycharm/
Atom: https://atom.io/packages/ide-python
I highly recommend using an Integrated Development Environment to develop good coding habits. Examples are: **pycharm, spyder, eclipse, atom, IDLE and visual studio** in order of preference. You can start with using a jupyter notebook but more complex projects are easier managed through an IDE. The TA can assist with potential installation problems during the first week, but ideally you should come with having both python-anaconda and your favorite IDE installed.
Summary:

1. Install python anaconda and use python2.7 if you are not planning on using a university computer
2. Pick and install your favorite Development Environment. Spyder comes with the anaconda package and IDLE should ship with any standard system python installation. You will have to install other IDEs if you prefer to use them.
3. Bring your laptop to class every day and make sure it is fully charged.

Grading

Lecture Quizzes	5%
Homework	45%
Midterm Exam	20%
Final Project	30%

Tentative Course Outline

Time frame: April 1st - June 13th, with finals from June 10th-13th, expected work load: 15 hours/week including 4 hours lectures + lab

Introduction to python and quantitative computing

- Week 1 (04/01 - 04/05)
  - What is open-source software and why you should use it, python package managers and IDEs (pycharm, visual studio, spyder),
  - Elements of python: modules, functions (discrete vs. continuous), strings, scalars and floats
  - Python scripts and best practices in programming
  Reading: 1.1, 1.2, 1.3, 1.5.3, 1.5.5, 1.5.8, 1.5.9, 1.5.11, 2.2

- Week 2 (04/08 - 04/12)
  - vector and matrix operations for fast computations, python lists and numpy arrays, indexing and loop structures
  Reading: 1.4, 2.1, 2.3, 2.4, 2.5
  Homework 1 - *functions for basic geometric area computations, 35 points* (due 4/14)

Data analysis

- Week 3 (04/15 - 04/19)
  - Working with data files, plotting, animations and geo-referenced maps
  Reading: 1.5.7, 2.6
  Homework 2 *seismicity animations, geo-referenced plotting with basemap, 45 points* (due 4/21)

- Week 4 (04/22 - 04/26)
  - Statistics and model fitting, Least-Squares and Maximum-Likelihood
  Homework 3 *Power law fitting and rate computations, 40 points* (due 4/28)

- Week 5 (04/29 - 05/03)
  - Machine learning: 0R, 1R, confusion matrix, Naive Bayes, Neural Networks, Decision Trees
  Homework 4 *Number recognition and neural networks, 40 points* (due 5/5)

- Week 6 (05/06 - 05/10)
  - Debugging, automated testing, python style guides, interpolation, numerical integration and differentiation, root finding
  Reading: 3.1, 3.2, 3.3, 3.4, 3.5, 3.6
  Midterm - data analysis

Numerical modeling with finite difference

- Week 7 (05/13 - 05/17) - ODEs, direction fields, Euler Formula and Runge-Kutta
  Reading: 4.1.2 - 4.1.5; 4.3.1 - 4.3.7
  Discuss topics for final project

- Week 8 (05/20 - 05/24) - Logistic Equation, Forced vibrations
  Reading: 4.3.12, 4.3.13
  Homework 5 *Undamped, force oscillator, 40 points* (due 5/19)

- Week 9 (05/27 - 05/31) - Partial differential equations - Diffusion equation
  Reading: 5.1.1 - 5.1.5
  Homework 6 *Heat diffusion, 40 points*(due 5/26)
- Week 10 (06/03 - 06/07) - Partial differential equations - Diffusion equation
  Work on final projects

Course Policies

- **General**
  - **All code must have comments or you will loose points even if it runs smoothly.**
  - Grades will be maintained on https://canvas.ucsc.edu/.
- **Attendance**
  - Attendance is expected for each lecture. At the start of each class there will be a question to be answered and turned in.
- **Homework**
  - Homework assignments will be posted and turned in via Canvas
  - Homework assignments are due by the end of each week, i.e. Sunday before midnight
  - Total possible points will be reduced by 30% on late assignments.
  - You are encouraged to collaborate on homework assignments but what you turn in must be your own work. This means you may not copy-paste anyone else's work. For more on this see Academic Honesty in Programming.
- **Final Project** The final project is a big part of the class and will give you the opportunity to apply much of what you have learned to a problem of your choice. There are three types of problems that you can work on:
  1. General data analysis and statistics applied to a particular Astro/Earth data set
  2. Machine Learning applied to Earth/Astro data set
  3. Mathematically model an Astro/Earth system and solve relevant ODE or PDE.

  Your final project will include a well-documented python code and a short (5 pages max.) paper. More details about the final project will be discussed toward the end of the class.

Academic Honesty in Programming

(Taken from Mark Krumholz's Winter 2015 syllabus.)

Plagiarism is defined as copying the work of another and presenting it as your own, and is no more acceptable in computer programming classes than in other contexts. Here are a few guidelines that apply to computer programming in particular:

- It is unacceptable to copy and submit as your own all or substantial portions of another as your work, with or without attribution. It is acceptable to copy a few lines of code, or even a small subroutine, and incorporate those into your own, more complex program, provided that you acknowledge your source. This need not be a formal footnote; a short comment in the source code is fine, for example

  ```
  # This line is taken from http://www.url.com/neat/programming/idea
  ```

- In the same vein, it is unacceptable to post the entirety of a homework question on a message board like http://stackoverflow.com/ and request assistance with it. However, it is acceptable to ask general questions regarding specific tasks that you must accomplish as part of the assignment.

- For more on academic integrity in computer programming classes, please see the very thorough discussion at http://www.cs.cornell.edu/courses/CS1133/2014fa/about/integrity.php.
- If in doubt about whether something is acceptable, please ask. You will never be penalized for asking.

DRC Accommodations

UC Santa Cruz is committed to creating an academic environment that supports its diverse student body. If you are a student with a disability who requires accommodations to achieve equal access in this course, please submit your Accommodation Authorization Letter from the Disability Resource Center (DRC) to me privately during my office hours or by appointment, preferably within the first two weeks of the quarter. At this time, I would also like us to discuss ways we can ensure your full participation in the course. I encourage all students who may benefit from learning more about DRC services to contact DRC by phone at 831-459-2089 or by email at drc@ucsc.edu.