CS510 : Computing for Scientists

Fall 2017

Instructor:	Dr. Justin Dressel	**Time:**	Tue 19:00 – 21:50
Email:	dressel@chapman.edu	**Place:**	203 Beckman Hall
Office:	110 Hashinger Science Center		

Office Hours: Tue/Thur 14:00 – 15:00, 110 Hashinger Science Center, or by appointment.

Course Page: https://github.com/chapman-cs510-2017f

Course Description: CS 510 is a graduate-level course intended to introduce modern computing tools and techniques to science-oriented students from diverse backgrounds. Assuming little prior knowledge, students will become proficient with a powerful set of inter-operable tools that are suitable for problem-oriented and data-intensive applications now common in modern science. While emphasizing the central role of data (structuring, processing, and visualization), students will use industry-best software development practices to develop efficient implementations and visualizations of numerical solutions to scientific problems. Students will be expected to complete programming assignments in freely available languages such as Python, Julia, C, and C++.

Course Textbook: None in favor of *free* web resources.

Python/SciPy Tutorial: http://www.scipy-lectures.org
https://lectures.quantecon.org/py/learning_python.html

LaTeX Tutorial: https://www.latex-tutorial.com

`bash` **Tutorial:** http://linuxcommand.org/lc3_learning_the_shell.php

`vim` **Tutorial:** https://openvim.com

`git` **Tutorial:** https://tutorialzine.com/2016/06/learn-git-in-30-minutes
https://try.github.io
http://tom.preston-werner.com/2009/05/19/the-git-parable.html

`julia` **Tutorial:** https://lectures.quantecon.org/jl/learning_julia.html
http://ucidatascienceinitiative.github.io/IntroToJulia/

`C` **Tutorial:** https://www.raspberrypi.org/magpi/learn-code-c/

`C++` **Tutorial:** http://www.cplusplus.com/doc/tutorial/
https://tfetimes.com/wp-content/uploads/2015/09/CppEssentials.pdf

Optional Textbooks:

Elegant SciPy: The Art of Scientific Python, Nunez-Iglesias *et al.*, O'Reilly, 2017.
https://github.com/elegant-scipy/elegant-scipy

Hands-On Machine Learning with Scikit-Learn & TensorFlow, Géron, O'Reilly, 2017.
https://github.com/ageron/handson-ml

Accelerated C++, Koenig and Moo, Addison-Wesley, 2000.

Student Assessment:

Class Participation and Homework (35%), Midterm Project (30%), Final Project (35%).

Instructional Methods and Strategies: The class is project-based, with extensive in-class work involving both groups and individual assignments. The philosophy is to learn by doing, with lecture guidance from the instructor only as necessary. Participation in class will be expected, and will require reading, searching the web, and studying assignments prior to class. Students will complete a midterm project and a final project, as well as regular weekly assignments.

Methods of Evaluation: In addition to oral and written in-class assessment of abilities, students will be challenged by a combination of weekly homework coding projects, as well as mid-term and final coding projects.

Collaboration Policy: I encourage you to discuss and study course material together. However, all work you submit for this course must be your own. Non-original material must be properly cited in a README file turned in with your assignments. Any incidents of academic misconduct will be dealt with severely in accordance with the Chapman University Academic Integrity policy (see below).

Blackboard: Assignment links will be posted on Blackboard. Grades will be posted on Blackboard. Updated course schedules and official announcements will be posted on Blackboard.

GitHub: Homework will be assigned and collected via `git` using http://www.github.com and Git Classroom. Your first task will be to ensure that you have an account, and know how to use it. The GitHub Organization for the course will be http://github.com/chapman-cs510-2017f.

Slack: Group discussion and contact with the professor will be facilitated by Slack, at http://scststudents.slack.com. Your second task will be to ensure that you have an account. Please notify the instructor if you need to be invited. The channel for this course will be `#cs510-17f` and is set to auto-notify the instructor. Note that this is a public forum, but private chats are also available.

CoCalc (formerly SageMathCloud)**:**

For ease of compatibility, we will be using http://cocalc.com as a browser-based development solution. Your third task is to ensure that you have an account, and that you inform the instructor of your username so that you may invited to the course project. Please use your Chapman email when registering your account, and choose a username such that your real name is clear. Your account will give you access to a virtual Linux machine running Ubuntu, complete with an accessible bash terminal, text editors like `vim` or `nano` or `emacs`, LaTeX, Jupyter notebooks, Numeric Python, Scientific Python, Pandas, C, C++, Julia, Octave, and many other useful tools: http://tutorial.cocalc.com/students/ You are welcome to use your own computer in addition to *CoCalc*, but it is your responsibility to install all software and ensure compatibility with *CoCalc*.

Course Learning Outcomes: By end of semester, the students should

1. Competently manipulate files, directories, and utilities in the Linux operating system.
2. Be able to set up a programming environment, showing ability to productively use:
   (a) Console editors (e.g., `vim`, `emacs`, `nano`)
   (b) Interactive interpreters and notebooks (e.g., Jupyter)
   (c) Compilation frameworks (e.g., `make`)
   (d) Version control (e.g., `git`)
3. Show facility using common data formats (e.g., CSV, JSON, HDF).
4. Use freely available programming languages for data processing and simulation:
   (a) Interpreted languages with compiled libraries (e.g., Python)
   (b) Just-in-time compiled languages (e.g., Julia)
   (c) Natively compiled languages (e.g., C, C++)
5. Use free data visualization libraries (e.g., `matplotlib`, `seaborn`, `bokeh`) to generate publication-quality figures for notebook-based reports.
6. Understand machine representations of numbers, and boxed/unboxed data structures well enough to perform rudimentary algorithmic efficiency and precision analysis.
7. Demonstrate the basic principles of software engineering, such as test-driven development, modular design, extensibility, verification, validation, optimization, and refactorization.

Computational and Data Sciences Program Learning Objectives:

1. Graduates will develop quantitative reasoning skills which will enable them to: solve problems by utilizing extrapolation, approximation, precision, accuracy, rational estimation and statistical validity; interpret data; and create quantitative models to describe natural phenomena.
2. Graduates will be able to apply the principles of computational science to scientific problems. Students will develop critical thinking, end to end problem-solving, and data analysis skills. With these skills, they will be able to: collect process and analyze data; prioritize different potential solutions to a problem; and use advanced mathematics and computing to solve scientific problems.
3. Graduates will be able to apply principles of applied mathematics to scientific problems in order to: evaluate the accuracy of approximations; and interpret the results of calculations.
4. Graduates will be able to apply principles of computer technology and computer science to scientific problems. In particular, they will be able to: use advanced high performance computer architectures including clusters and supercomputers. Create programs to manipulate and analyze data on HPC systems; construct solutions to scientific problems using advanced parallel algorithms and data structures; and analyze the performance of algorithms.

Major Topics Covered:

1. Unix/Linux filesystem and bash shell
2. Console editors: vim, emacs, and nano tutorials
3. Distributed version control: git
4. Python object-oriented programming, IPython/Jupyter notebooks
5. C++ object-oriented programming, make compilation system and linking
6. Bit-representations of numerical data, boxed data, pointers, and references
7. Data interchange formats: CSV, JSON, HDF
8. Python visualization libraries: matplotlib, seaborn, bokeh
9. Scientific data-processing: pandas, numpy, scipy, julia
10. Test-driven development: nose, CppUnit

Chapman University Academic Integrity Policy:

Chapman University is a community of scholars which emphasizes the mutual responsibility of all members to seek knowledge honestly and in good faith. Students are responsible for doing their own work, and academic dishonesty of any kind will not be tolerated anywhere in the university. At their discretion the faculty may submit work to plagiarism detection software, such as www.turnitin.com for review.

Chapman University Students with Disabilities Policy:

In compliance with ADA guidelines, students who have any condition, either permanent or temporary, that might affect their ability to perform in this class are encouraged to contact the Office of Disability Services. If you will need to utilize your approved accommodations in this class, please follow the proper notification procedure for informing your professor(s). This notification process must occur more than a week before any accommodation can be utilized. Please contact Disability Services at (714) 516-4520 or http://www.chapman.edu/students/studenthealth-services/disability-services if you have questions regarding this procedure, or for information and to make an appointment to discuss and/or request potential accommodations based on documentation of your disability. Once formal approval of your need for an accommodation has been granted, you are encouraged to talk with your professor(s) about your accommodation options. The granting of any accommodation will not be retroactive and cannot jeopardize the academic standards or integrity of the course.

Chapman University Equity and Diversity Policy:

Chapman University is committed to ensuring equality and valuing diversity. Students and professors are reminded to show respect at all times as outlined in Chapmans Harassment and Discrimination Policy: http://ow.ly/XEwTu Any violations of this policy should be discussed with the professor, the Dean of Students and/or otherwise reported in accordance with this policy.

Student Support at Chapman University:

Over the course of the semester, you may experience a range of challenges that interfere with your learning, such as problems with friend, family, and or significant other relationships; substance use; concerns about personal adequacy; feeling overwhelmed; or feeling sad or anxious without knowing why. These mental health concerns or stressful events may diminish your academic performance and/or reduce your ability to participate in daily activities. You can learn more about the resources available through Chapman University's Student Psychological Counseling Services here: http://www.chapman.edu/students/health-and-safety/psychological-counseling/

Approximate Course Outline:

Week	Dates	Topics/Notes
1	8/29	Orientation, Overview, Tools
2	9/5	Bash and Console Editors
3	9/12	Git, Python, and Nose
4	9/19	IPython and Jupyter
5	9/26	Modular and Object-oriented Design
6	10/3	Numpy and Matplotlib
7	10/10	Pandas and Data Formats (Midterm project assigned)
8	10/17	Julia and Just-in-time Compilation
9	10/24	C, Make, and Linking (Midterm project due)
10	10/31	C Pointers and Structs
11	11/7	C Arrays and C++
12	11/14	C++ Classes and CppUnit
13	11/21	**Thanksgiving Break**
14	11/28	Operator Overloading and Templates (Final project assigned)
15	12/5	Work on final project — class canceled (*Instructor at conference*)
16	12/12	**Finals Week** (Final project due)