University California, Berkeley
Professional Masters in Molecular Simulation and Software Engineering
Online Degree

Chem 274A, 3 Units
Introduction to Programming Languages – C++ and Python
Fall 2021

Prof. Teresa Head-Gordon
274 Stanley Hall
thg@berkeley.edu

Course description:

This course provides in-depth coverage of programming concepts and techniques required for scientific computing, data science, and high-performance computing using C++ and Python. The course will compare and contrast the functionalities of the two languages. Topics include classes, overloading, data abstraction, information hiding, encapsulation, file processing, exceptions, and low-level language features. Numerous exercises based on molecular science problems will provide the hands-on experience needed to learn these languages. This course serves as a prerequisite to later MSSE courses: Data Science, Machine Learning Algorithms, Software Engineering for Scientific Computing, Numerical Algorithms Applied to Computational Quantum Chemistry, and Applications of Parallel Computers.

Contribution of this course to the broader curricular objectives: Required course for all MSSE students.

Course format: Five hours Faculty-led, asynchronous, web-based instruction, four hours of web-based synchronous discussion and two hours of web-based, synchronous lab per week to complete the course in 8 weeks. GSIs will go over homework assignments and practice exercises that are quantitative, prepare students for their homework assignments and post answer guides to homework assignments after they are submitted. Outside class work should comprise about 9 hours a week for a total of eighteen hours per week.

Reading List and Resources:
The Linux Command Line (5th edition), William Shotts. Can be downloaded for free (http://linuxcommand.org/tlcl.php)

The C++ Programming Language (4th Edition), Bjarne Stroustrup, Addison-Wesley. ISBN 978- 0321563842. May 2013.

http://www.stroustrup.com/bs_faq.html

Absolute C++ (6th Edition), Walter Savitch and Kenrick Mock, Pearson, 2016

A Computer Science Tapestry: Exploring Computer Science and Programming with C++ (2nd Edition), Owen L. Astrachan, McGraw-Hill. 1999.

Think Python: How to Think Like a Computer Scientist 2nd Edition, Alex Downey,

O'Reilly

Grading: There will be 4 well-staged programming assignments, 8 problem sets (weekly) and in-class asynchronous exercises. In-class exercises are interspersed between asynchronous video lecture material. An online coding platform with an auto-grader will be used for the in-class exercises. Submissions will be reviewed by GSIs if necessary. Programming assignments and weekly assignments will be graded by GSIs or faculty.

- 40% weekly problem sets (8 assignments making up 5% of grade each)
- 40% programming assignments (4 assignments making up 10% each)
- 20% in-class exercises and discussion participation

Prerequisites: Prior exposure to basic programming methodology or the consent of the instructor.

Course requirements: Each student is required to view all of the online lectures, do all the online quizzes, submit all homework assignments, and discussion postings for grading. A laptop, workstation, or access to a UNIX style account is required, as is installation of an Emacs or VI editor.

Office hours: The instructor will be available 2 hours per week for one on one consultation by appointment. The GSIs will be available 10 hours a week. These synchronous office hours will be posted on the course website. The instructor will also be available for synchronous open class discussion two hours per week. These will be archived and available to students**.**

Learning objectives for this course: Upon successfully completing this course, students will be able to

A. Develop the necessary skills to effectively interact with machine learning environments.
B. Acquire the skills needed to develop high-performance computing software.

Course Schedule:

Week 1: Introduction to Linux and the command line. Environment variables, shells, command line file navigation, piping and redirection, common utilities (grep, sed, find), and bash scripting. Regular expressions, and the vim text editor. File permissions, File manipulation: tar, cat, more, compression, bzip2, etc.

Week 2: Introduction to object oriented programming. Programming paradigms. Procedural vs. object oriented and functional. Object oriented programming in Python.

Week 3: Python data structures. Mutable and immutable data types. Strings, lists, tuples, sets, dictionaries. Variable scope. Object introspection. List comprehension, dictionary comprehension, lambda functions.

Week 4: Class definitions in C++, getters, setters, public and private members. C++ templates & generic programming. Comparison to Python, dunder methods, use of decorators. New and delete vs. garbage collection.

Week 5: . C++ standard library. Vector, array, map, unordered_map, set, string, list. Common algorithms (fill, copy, find, sort). Streams & IO

Week 6: Resource management. C++ RAII, smart pointers. Python context managers & garbage collection.

Week 7: Debugging (gdb, pdb). C++ memory sanitizers. Python scripts and C++ programs with command line arguments - argv and argparse. Python package managers. Elemental profiling.

Timing/hotspot detection

Week 8: Scientific applications and common packages. Pandas, Eigen, Numpy vector operations: apply, mapping function over numpy arrays. Scientific visualization, plotting with Matplotlib.