University of California, Berkeley
Master of Molecular Science and Software Engineering (MSSE)

CHEM 280, 2 units
Foundations of Programming and Software Engineering for Molecular Sciences
Fall 2023

Course Description

This course provides an overview of topics relevant to programming and creating software projects. The course will be taught in collaboration with members of the Molecular Sciences Software Institute (MolSSI). Students will learn basic syntax, use cases, and ecosystems for Python and C++. Students will become familiar with tools and practices commonly used in software development such as version control, documentation, and testing. Central to this course is a hands-on molecular simulation project where students work in groups to create a software package using concepts taught in the course.

Students will work collaboratively in teams to implement a program which does Monte Carlo simulation of noble gas fluids. Monte Carlo techniques related to molecular simulation rely on random numbers and mathematical models of molecular interactions to predict equilibrium properties of a chemical system. The students will work to implement three versions of this project. In the first, they will implement the code using only the Python Standard Library. This version is coded mostly in class, with the students writing supplemental code or writing specific functions for homework. For the second version of this code, they must utilize the Python library NumPy. NumPy will make the code much faster and more efficient, and this exercise will provide an introduction to code vectorization. For the third version of the code, the students will work to implement the project using the programming language C++. Students are given guidance for writing of all code, with some exercises being optional bonus items.

Students will write their Monte Carlo code and confirm correct behavior through comparisons with benchmarks reported by the National Institute of Standards and Technology (NIST). After confirming behavior, the students will be asked to apply their simulation program to conditions not reported in the benchmarks in order to make predictions.

This project will introduce foundational concepts across software engineering, programming, and molecular science. The foundational concepts related to each topic are listed below:

Molecular Science:

1. This project will introduce students to Monte Carlo methods. Monte Carlo methods are used across many fields including biology, economics, finance, physics, and molecular science. The simulation of simple fluids (as used in this project), were first reported in the 1950s. More complex MC simulations continue to be utilized today.
2. This project will introduce students to the Lennard Jones potential energy function. This potential energy function is used in molecular simulation programs to model nonbonded interactions.
3. This project will introduce students to analysis of molecular coordinates to determine system properties, a foundational idea in molecular simulation and statistical mechanics.

Programming:

1. The student will use both C++ and Python for this project. Through this, they will learn the difference in interpreted vs compiled languages.
2. Python – This course assumes knowledge of basic python syntax. A free, optional, course was offered to the students by the instructors in the first two weeks of June to establish a baseline of programming competency. The instructors also gave students programming challenges in Python and have provided detailed feedback to the students on these programming challenges. In the bootcamp, the students
    a. Are introduced to the Python Standard Library
    b. Learn about Python coding style
    c. Learn about writing good python coding documentation
3. C++
    a. C++ syntax, standards, compilation, data types.
    b. C++ documentation strategies
    c. C++ testing

Software Engineering:

1. Version control – Throughout this course, students will use the version control software git to keep a record of their projects. Git is introduced to the students on the first day of instruction, and used throughout the course.
2. Testing – Students will be introduced to the idea of testing through manual verification, and later through the use of testing frameworks in both Python and C++
3. Documentation – Students will be introduced to standard code documentation formats for both Python and C++
4. Code Collaboration – Central to this course is the collaboration on the Monte Carlo code. The students will use the collaboration technique which is widely applied in the open-source software community. After completion of this course, students will be familiar with open-source software code contribution workflows that are used in for open source software in all big tech companies including Google, Microsoft, and IBM and for projects such as TensorFlow, Kubernetes, and the Linux Operating System.

Grading

- Individual Homework: 40%
    - 4 individual coding challenges
- Group Assignments: 30%
    - 6 groups coding assignments
- Participation and Code Review: 10%
    - Review group pull requests
- Final Project and Presentation: 20%

Individual Homework – During the course of the bootcamp, the students will be given four individual coding challenges which relate to the programming concepts that are taught, but are not part of their group projects.

Group Assignments – The students will be given assignments each day which relate directly to their Monte Carlo code. The students must divide tasks in their groups in order to implement these. The students will submit these to instructor-controlled GitHub repositories using the open-source software workflow.

Discussion/Code Review – In order for student code to be accepted into the instructor-controlled team repository, it must be reviewed and approved by at least one other team member. This code review technique is used for all open-source software projects. Students must submit code each day related to their project, and must also review the code of at least one other student.

Final Project and Presentation – On the last day of the course, students will be expected to give a final presentation about the final state of their project. This presentation should be 15-20 minutes per group, and should explain their software and performance, simulation results, as well as reflections from the experience. Students will be encouraged to present any creative work or extra code improvement they implemented.

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

Course format
Thirteen hours of Faculty-led instruction per week, 10 hours of Faculty-guided discussion and group work per week, and 10 hours of optional Faculty-led office hours per week to complete the course in 2 weeks.

Reading List and Resources
Online materials (the equivalent of a textbook for the class) will be produced for the course and shared through bCourses.

Prerequisites
Acceptance to MSSE program.

Course Requirements
Each student is required to attend all lectures, and to work on an assigned team on a software project. Each student must have a laptop with either a Linux or Mac operating system. Windows users should have the Windows Subsystem for Linux (WSL) installed. Students will also need to install the Conda package manager, and a text editor. Installation instructions will be provided to students before the class begins.

Office Hours
The instructors will be available for discussion with students for two hours each day after instruction.

Learning Objectives

Upon successfully completing this course, students will be familiar with foundational programming concepts and practices, and will have the skills to begin the MSSE degree.

Late Assignment Policy

Because of the short duration of the course, late submission for group assignments and code review will not be accepted. Individual assignments may be turned in late for a penalty of 10% per day late up until 5 days after the assignment due date.

Academic Accommodations

UC Berkeley is committed to creating a learning environment that meets the needs of its diverse student body including students with disabilities. If you anticipate or experience any barriers to learning in this course, please feel welcome to discuss your concerns with me.

If you have a disability, or think you may have a disability, you can work with the Disabled Students' Program (DSP) to request an official accommodation. The Disabled Students' Program (DSP) is the campus office responsible for authorizing disability-related academic accommodations, in cooperation with the students themselves and their instructors. You can find more information about DSP, including contact information and the application process on their website: dsp.berkeley.edu. If you have already been approved for accommodations through DSP, please meet with me so we can develop an implementation plan together.

Students who need academic accommodations or have questions about their accommodations should contact DSP, located at 260 César Chávez Student Center. Students may call 642-0518 (voice), 642-6376 (TTY), or e-mail dsp@berkeley.edu.

Course Schedule

Week 1: Introduce programming practices and software projects. Topics include text editors/IDEs, bash shell, Python syntax, C++ syntax, an overview/comparison of each language. Coding best practices including version control, code documentation and testing introduced.

Week 2: Coding best practices are used and expanded upon. Creating a software package, code profiling, and code performance concepts are introduced. Students work to improve code performance by binding C++ and Python. On the last day, they must give a presentation showing their project.

Daily Schedule

The course is taught daily with three hours (including breaks) of faculty-led instruction in the mornings, followed by faculty-supervised group work in the afternoons. Students should plan to be present during the lecture and group work. Office hours following group work are optional, with faculty available until 5 PM PT.

Time	**Topic**	**Notes**
9:00 AM - 12:00 PM PT	Hands-on lecture	The lecture block will include one 15-20 minute break.
12:00 PM - 1:00 PM PT	Lunch break	
1:00 PM - 4:00 PM PT	Group Discussion and Assignment	The students should be present for the 1:00 PM - 3:00 PM block, but can take short breaks as needed.
4:00 PM - 6:00 PM PT	Optional Office Hours	Students should continue to work on group assignments during this time if necessary.

Day	Lecture/Discussion Topic(s)	Assignment(s)
1	Introduction to command line navigation, version control using git, and code collaboration.	Group Assignment 1: Team Blogs and Group Introductions.
2	Introduction to Monte Carlo (MC)methods, Writing a Pythong script to calculate pi using MC, Introduction to the Lennard Jones equation for molecular systems.	Group Assignment 2: The Lennard Jones Potential and Cutoffs, LJ Tail Corrections, Periodic Boundaries Individual Assignment 1 available - due in one week.
3	Monte Carlo of a Lennard Jones fluid	Group Assignment 3: Acceptance criteria, initial configuration, and radial distribution functions for simulation analysis.
4	Introduction to NumPy - arrays, broadcasting. Revisiting MC using NumPy arrays.	Group Assignment 4: Rewriting MC code using NumPy Individual Assignment 2 available - due in one week.
5	Python packages, introduction to the pytest framework	Group Assignment 5: Making code into package, adding testing and analysis.
6	Introduction to C++, compiled languages, writing first C++ program.	Group Assignment 6: Writing C++ integration code, Riemann sums.

7	Common C++ data types, arrays, pointers, memory, and vectors.	Final Group Project Assigned (due on the final day of class) Individual Assignment 5 available.
8	C++ function calling and arguments, namespaces, and exceptions.	
9	C++ multi-file projects, documentation, testing.	
10	Group Project Presentations and Wrap Up	Individual Assignment 4 available.