University California, Berkeley
Master of Molecular Science and Software Engineering

CHEM 274A, 3 Units
Programming Languages for Molecular Sciences: Python and C++
Fall 2023

Jessica A. Nash and Benjamin P. Pritchard
janash@berkeley.edu ; bpp4@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: Three hours weekly of Faculty-led, asynchronous, web-based instruction; two hours weekly of web-based synchronous discussion; and two hours of web-based, synchronous lab every other week to complete the course in 15 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 four hours a week for a total of nine to eleven hours per week.

Prerequisites: Prior exposure to basic programming methodology, use of git software and GitHub website or the consent of the instructor.

Reading List and Resources:

All books are free or available through UC Berkeley's library. Readings are primarily provided as supplemental options for the student. They are not required, but are strongly recommended to get the most out of the course.

- 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. Available as e-book through UC Berkeley Libraries.
    - More C++ info: http://www.stroustrup.com/bs_faq.html

- Think Python: How to Think Like a Computer Scientist 2nd Edition, Alex Downey, O'Reilly. Available free online and as an e-book through UC Berkeley Libraries.
- Learning Python, 5th edition, Mark Lutz, O'Reily. Available as an e-book through UC Berkeley Libraries.
    - This book was published in 2013, but is still an excellent resource for learning intermediate to advanced Python. The author keeps a website (https://learning-python.com/python-changes-2014-plus.html) where he tracks updates and language changes.

Grading: There will be 2 programming projects, 5 problem sets, and asynchronous "lecture" exercises. Lecture exercises are interspersed between asynchronous video lecture material. An online coding platform with an auto-grader will be used for the lecture exercises. Submissions will be reviewed by GSIs if necessary. Programming assignments and weekly assignments will be graded by GSIs or faculty.

- 40% problem sets
- 20% programming assignments
- 20% lecture exercises
- 20% discussion and lab.

Assignment Types:

- **Problem Sets** are due biweekly and will cover material from the previous two weeks of the course.
- **Programming Projects** are larger assignments integrating concepts from multiple weeks. Both projects will be due at the end of the course.
- **Lecture Exercises** are integrated with the lecture material and are expected to be completed as you move through the course material. These will be short, single concept, programming problems which test concepts covered in lecture. These problems are autograded and must be finished by the end of the week.
- **Discussion and Lab** are sessions where you put concepts you learn in class into action with your classmates. Attendance is required, and each discussion and lab will have an assignment that is to be completed during the session and turned in at the end of the session. Some assignments will be completed collaboratively with a team.

Missed Synchronous Sessions: Missing synchronous sessions (discussion and lab) will result in no credit being awarded for the missed session unless the absence is pre-arranged with the instructor. Accommodations may also be made for emergencies.

Late work: Late work will not be accepted for the weekly lecture exercises. For biweekly problem sets and programming assignments, late work will be accepted for a penalty of 10% per day until a week after the assignment due date. Problem sets and programming assignments turned in more than a week late will not be accepted without pre-approval from the instructor.

Accommodations for Disabilities UC Berkeley is committed to creating a learning environment that meets the needs of its diverse student body including students with disabilities. 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. https://dsp.berkeley.edu/. If you already have an accommodation letter from DSP, please check to make sure that the letter is submitted through the DSP system (there is no need to email a separate copy). If you would like to set up an individual meeting to discuss your accommodations, please contact the instructors.

Collaboration Policy: Unless otherwise instructed, all assignments are to be completed independently and materials submitted as homework should be the result of one's own independent work.

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

Office hours: The instructors will be available 2 hours per week for one on one consultation during office hours. The GSIs will be available 4 hours a week. These synchronous office hours will be posted on the course website. The instructors 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

- Understand the Python Standard Library and the C++ Standard Library.
- Interact with a computer using a text interface (the shell) and use text-based text editors.
- Write bash scripts to automate running programs.
- Use remote computing resources.
- Write object-oriented Python and C++ programs which utilize language features.
- Develop the necessary skills to effectively interact with machine learning environments.
- Acquire the skills needed to develop high-performance computing software.

Course Schedule:

This course schedule outlines lecture topics, suggested readings and assignments due. **Note that there is a Discussion session every week and a lab session every other week.**

Week #	Concepts	Readings	Assignments Due
1	Course introduction and expectations. Organizing projects, introduction to code/project documentation. Python code formatters and linters (black, flake8). Python coding style (PEP8), type hinting.		☐ Lecture Exercises

2	Programming paradigms - procedural, object-oriented, and functional programming. Python class definitions. Encapsulation and inheritance. Introduction to the Linux command line: environment variables, shells, command line file navigation.	• Chapters 15-18 in Think Python • Chapters 2, 4 and 11 in The Linux Command Line	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Synchronous Lab
3	C++ class definitions, getters, setters, public and private members, constructors and destructors, and inheritance. Introduction to command line text editors (vi/vim)	• Chapter 16, 17,20 in The C++ Programming Language • Chapter 12 in The Linux Command Line	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Problem Set 1
4	Python standard library - Objects and data types. Mutable and immutable objects . Strings, lists, tuples, sets, dictionaries. Command line - common utilities (grep, sed, find)	• Chapter 7, 8, 10, 11, 12 in Think Python • Chapter 17 in The Linux Command Line	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Synchronous Lab
5	Python variable scope. Object introspection. List and dictionary comprehensions, and generators. Bash scripting	• Chapter 17, 20 in Learning Python • Chapter 24 in The Linux Command Line	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Problem Set 2
6	C++ templates and generic programming. C++ operator overloading. Command line: file permissions	• Chapter 18, 23, 24 in The C++ Programming Language • Chapter 9 The Linux Command Line	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Synchronous Lab

7	Resource management. C++ scope, RAII, and smart pointers	• Chapter 6, 13, 34 in The C++ Programming Language	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Problem Set 3
8	C++ standard library: iterators and common containers (vector, array, map, unordered_map, set, string, list)	• Chapter 30, 31, 33 in The C++ Programming Language	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Synchronous Lab
9	C++ standard library algorithms: fill, copy, find, sort. Streams, I/O, and file I/O	• Chapter 32 in The C++ Programming Language	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Problem Set 4
10	Python context managers, decorators and exceptions. Dunder methods and operator overloading in Python	• Chapter 30, 34, 35, 39 in Learning Python	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Synchronous Lab
11	Debugging and debuggers. Basic profiling and C++ memory sanitizers. Features of Integrated Development Environments (IDEs).		☐ Lecture Exercises ☐ Synchronous Discussion ☐ Problem Set 5
12	Python scripts and C++ programs with command line arguments: argv and argparse. Python packages, Python package managers and environments.	• Chapters 22-24 in Learning Python.	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Synchronous Lab
13	Python scientific applications and common packages		☐ Lecture Exercises ☐ Synchronous Discussion

14	Introduction to remote computing resources - developing on VMs and cloud computing	• Chapter 16 in The Linux Command Line	☐ Lecture Exercises ☐ Synchronous Discussion ☐ Synchronous Lab
15	C++ Scientific applications and common packages		☐ Lecture Exercises ☐ Synchronous Discussion ☐ **Programming Project 1 and 2 Due**