Programming and Data Structures for ML

ECE 590 - P4ML
Fall 2025

Syllabus and Schedule

INSTRUCTOR

Javier Pastorino, Ph.D.
211 Hudson Hall
javier.pastorino@duke.edu
Make and Appointment.

Teaching Assistants

Jay Ding
jay.ding@duke.edu
Wihang Zhang
weihang.zhang@duke.edu

Office Hours

Instructor and TA office hours will be posted in the course website.

Main Sessions

Monday/Wednesday 10:05AM-11:20AM
Wilkinson 136

Discussions

Friday 10:05AM-11:20AM

- Section 1: 218 Hudson Hall
- Section 2: 222 Hudson Hall

Course Website

https://duke.is/p4ml

This course introduces programming fundamentals and essential data structures using Python, tailored for graduate students pursuing machine learning engineering and machine learning for systems. Students will learn how to implement effective programs and design algorithms and systems that process and utilize data. We will explore core libraries used in data analysis, including NumPy and Pandas, and cover key concepts in object-oriented programming and design. Emphasis will also be placed on writing robust, testable code to support scalable and maintainable ML workflows.

What background do you need to take this course?

This class requires no previous background in programming, as we start from the beginning; however, students without any prior programming experience must work quite hard to keep up with the intense pace of the class: we cover more material in one semester than most undergraduate curriculums do in two semesters. This pace is required for all master's students to be "up to speed" for programming in coming semesters as well as getting them ready for industry internships, regardless of the background they came in with. Accordingly, the workload is high, but not impossible. Many students have become proficient programmers, even with no prior background, when they demonstrate **hard work**, **discipline**, and **focus**.

What will you get out of this course?

By the end of this course, you will be equipped to analyze, design, and implement small to mid-sized software systems that solve real-world problems involving data and user interaction. You will learn how to process and interpret data, interact effectively with users, and create well-structured programs that integrate machine learning models or other computational systems to support decision-making.

In terms of specific learning objectives, you will be able to:

- ***Understand*** core principles of software design and construction that transcend specific programming languages, drawing on concepts from software engineering, program design, and algorithm development.
- ***Comprehend*** the foundational techniques of program construction, including top-down design, abstraction, and modular development.
- ***Develop*** proficiency in Python, a powerful and widely-used programming language that supports structured, object-oriented, and functional paradigms.
- ***Read and write*** clear, idiomatic Python code to build functional, reusable program components.
- ***Apply*** data structures and programming abstractions to effectively represent, manipulate, and reason about data, with a focus on object-oriented design and problem decomposition.
- ***Design and implement*** high-quality object-oriented programs, including rigorous testing, debugging, and documentation practices.
- ***Use*** Software Configuration Management (SCM) tools (e.g., Git) for version control and collaborative software development workflows.
- ***Understand and explain*** the structure and behavior of key data structures (e.g., lists, dictionaries, trees, graphs) and analyze the time complexity of associated operations using Big-O notation.
- ***Understand*** the use of multithreading and multiprocessing, particularly when handling data or interacting with remote components.

How can you be successful in this course?

To succeed in this course, focus on three key practices:

1. **Come to class prepared.** Read the assigned material before class so you are ready to engage in discussions. While class will reinforce concepts, it's structured around exploring—not introducing—them. You don't need to fully grasp everything beforehand, but familiarity will help you follow and contribute to conversations.
2. **Engage consistently.**
   Learning to build systems is about practice, not memorization. Progress comes through iteration. Expect setbacks, and view them as part of the learning process.
3. **Start assignments early.**
   Assignments often raise questions. Begin them early to give yourself time to work through challenges and seek help.
4. **Reach out for help.**
   Be proactive, ask questions. The instructional team is here to support you —make sure you give yourself space to take advantage of that.

Course Information

Textbooks

We will not stick to one unique book, however, there are many references that we will use and will be helpful for you.

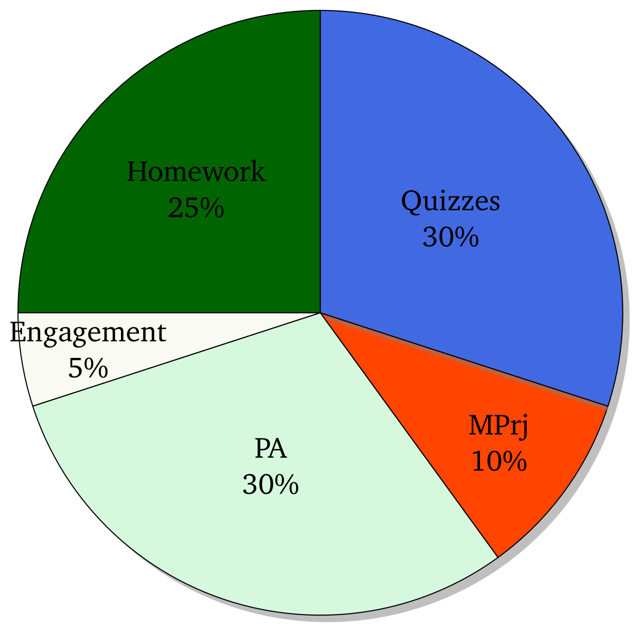
All Of Programming
A. Hilton, A. Bracy.
We will use Chapter 1 only; available online here.

Intro to Python: for Computer Science and Data Science
P. Deitel, H. Deitel.
Pearson
ISBN 13: 978-0-13-540467-6
The full text is available online at Duke Library.

Data structures & Algorithms in Python
J. Canning, A. Broder, R. Lafore
Pearson
ISBN 13: 978-0-19-485568-4
The full text is available online at Duke Library.

Course Grading

Your comprehension and mastery of the concepts we discuss as part of this class will be evaluated two-fold via a mixture of formative and evaluative assessments.



Formative assessments Formative assessments are more flexible and designed to help the student master one particular topic (e.g., linked-list). Formative assessments will be presented as small programming tasks (homework) that will be automatically tested. Feedback for these assessments will come in the form of passing some predefined tests; however, students are welcome and encouraged to discuss them with TAs or the instructor as needed. Other formative assessments may include preparation or in-class activities.

Evaluative assessments Evaluative assessments are less flexible and are designed to assess the knowledge the student gains on multiple concepts. Evaluative assessments will include larger programming assignments (also automatically tested), quizzes and a final exam.

Formative Assessments

Homework (25%) Homework are individual, small-sized programming tasks that will help students to apply the learned concepts. Homework will be automatically graded through ADTG. ADTG is a system that delivers, tests and tracks student homework; this system will test student's submissions and list any errors and success percentage. It uses test cases to verify correctness; *you will learn about this during the course as well!* Each homework may be graded as many times as needed to correct errors and improve score. It is recommended that homework be started ASAP so that there is time to discuss any issues with the TA or instructor. Homework is also chained, meaning you must complete each exercise (with at least 70% correctness) before moving to the next one, some homeworks may be released concurrently, though.

Deadlines for homework are soft –meaning you can still complete your homework after the deadline. However, doing so will have a grade penalty depending on how late your submission is. Your grade is the product of correctness (C) and timeliness (T): $G = C \times T$, where timeliness is computed with a logistic function of the number of hours (H) that your assignment is late: $T = \frac{1}{1+e^{(0.05\times(H-96))}}$. For instance, if your homework is 24-hours late $T = 0.973$, 2 days late $T = 0.917$, 3 days late $T = 0.769$, 4 days late $T = 0.5$, 5 days late $T = 0.231$, etc. After one week late, you will receive no marks for your late assessment however you still need to complete it!

Engagement (5%) As part of engagement, we consider students' attendance and participation in class. This includes participation (and timely submission) of in-class activities, as well as attendance (recorded for all sessions –including discussions). Moreover, some activities include reflections, which are also considered part of the engagement with the class, demonstrating virtues that are key for engineers, such as resilience, humanity, humility, and drive, among others.

Engagement is assessed through attendance, participation, and timely submission of a variety of activities. Attendance is recorded for all sessions (including discussions), using the following guideline: students arriving on time will receive 1 point, when they arrive within the first ten minutes of the session start, they will receive 0.8 points, and students arriving later than ten minutes will receive no points for that day. Engagement activities may include preparation tasks, reading quizzes, reflections, and in–class work, which emphasize key engineering virtues —like resilience, humility and drive. Activities may carry different amounts of points (usually between 1 and 5). Students who complete 90% or more of the available engagement points will earn full marks for the engagement total.

Evaluative Assessments

Quizzes (30%) Quizzes assess understanding of the concepts through small, on-paper coding exercises. Although quizzes are traditionally an evaluative assessment, as we embrace failing as a learning opportunity, we offer retakes for quizzes 1-4 for students who score a grade lower than 86% (*due to scheduling constraints, retakes for Quiz 5 will not be possible*. Students who earn a quiz grade between 86% and 92% may meet with the instructor during office hours (by appointment) to review feedback and discuss how to address it in order to improve their quiz grade. A discussion for Quiz 5 may be arranged for students who score between 70% and 92%. Students should request this as soon as possible as time is of the essence for scheduling such a meeting. Retakes will take place on Fridays early morning (time TBA), the week after the grade is released. Discussions for grade improvement must happen within a week after the grade is posted.

This process is designed not only to assess understanding, but also to foster resilience and tenacity—virtues essential to engineering practice. Students are encouraged to view setbacks as opportunities for growth and to engage actively in the feedback process.

Programming Assignments (30%) Programming Assignments (PA) are similar to homework, with the main differences being the size of the task and a **hard deadline** –*after the deadline you will receive no marks.* PAs are to be completed individually.

Mini-Project (10%) You will apply concepts we discuss in class —especially those that enhance performance, user experience, and connectivity of systems. The mini-project is more open-ended than the programming assignments (though still constrained to ensure demonstration of key concepts) and showcases practical applications of the material covered in class.

You will receive **guidance to formulate a project proposal**, which you should discuss with the instructor and plan to implement by **finals week**. During the **final exam period**, you will **showcase and defend your project results**, reflecting both your technical implementation and your understanding of the underlying concepts.

Final Letter Grade

At the end of the semester, the total weighted grade of each assessment category will be used to compute your final Letter Grade using the following scale (we will round if needed).

Course Policies

Academic Integrity

All members of the Duke community, myself included, are expected to abide by the Duke Community Standard (DCS).
`https://students.duke.edu/get-assistance/community-standard/`

> *Duke University is a community dedicated to scholarship, leadership, and service and to the principles of honesty, fairness, respect, and accountability. Citizens of this community commit to reflect upon and uphold these principles in all academic and nonacademic endeavors, and to protect and promote a culture of integrity.*
>
> *To uphold the Duke Community Standard:*
>
> - *I will not lie, cheat, or steal in my academic endeavors;*
> - *I will conduct myself honorably in all my endeavors; and*
> - *I will act if the Standard is compromised.*

What are my expectations with respect to academic integrity?

I expect that all work you submit for a grade is your own, individual, work. It is **not acceptable** to submit someone else's intellectual work product, **or generative AI's work product**, as your own. In particular, it is not acceptable to obtain, access, consult, or reference solutions to graded assignments prepared by others. This includes, but is not limited to, solutions authored by other current or former P4ML students, textbook solution manuals, solutions provided to other current or previous P4ML sections, materials for other courses at Duke, materials for other courses at other universities, websites (*e.g.*, CourseHero.com), online services (*e.g.*, 24HourAnswers.com), tutors (*e.g.*, Chegg.com), generative AI services (*e.g.*, ChatGPT), online repositories (*e.g.*, a shared Google Drive), or other materials found online or otherwise not generated by or originated by you.

I also expect that you will not share our course materials with others, whether to people as individuals or to groups, organizations, online services, or repositories (*e.g.* CourseHero.com, Chegg.com, a shared Google Drive).

What should you do if you're unsure if an action is allowable?

You may encounter situations where whether or not an action would be considered a violation of the DCS is not entirely clear to you. It is ok to be uncertain; there are a lot of situations that fall into gray zones where perspective can shift the evaluation of "ok" vs "not ok," or "right" vs "wrong".

> **If you are unsure whether an action would be considered a violation of the Duke Community Standard, please talk with me *before* you do it.**

A guiding principle is, if you wouldn't want anyone, or everyone, to know you did it, that is likely a strong indicator that you shouldn't do it.

How can you avoid academically dishonest actions?

Students most often experience lapses in judgment when they find themselves under time pressure to complete assignments. You can help yourself by starting your assignments well in advance of when they are due. It is never to your advantage to take academically dishonest shortcuts, as doing so will invariably lead to an outcome worse than either submitting an incomplete assignment or not submitting it at all.

Collaboration

Collaboration on homework and the mini-project, is permitted as far as you discuss concepts, ideas, and approaches. **You are not permitted to share solutions or code**. As part of discussing concepts, you may discuss which functions, packages, methods, etc. you used. It is not acceptable to share your code; it is not acceptable to receive code.

Absences, Missed Work, and Deadline Extensions

If circumstances prevent you from attending class in-person please reach out to me to discuss options to help you catch up on the missed class.

Options for accommodating circumstances that interfere with submitting an assignment by the due date vary according to the circumstance. In these cases, please make and appointment and let's discuss your case.

Incapacitating Short-term Illness

Please email me as soon as possible. Attach documentation (usually you request a letter of excuse to your doctor). Per university policy, "**it is your responsibility to meet or otherwise communicate with the instructor of the course in question within 48 hours** to discuss your missed work and how it will be treated in accordance with the instructor's policy. Otherwise, the instructor is under no obligation to accommodate your illness and can treat your missed work as unexcused.

Religious Observances

If a religious observance prevents you from submitting an assignment by the due date complete the form with at least two weeks before the date.

Tentative Schedule

The following schedule outlines our plan for this semester. While we aim to follow it closely, adjustments may be made as the course progresses.

Week	Topic
Week 1	Course Introduction Developing Programs
Week 2	Executing Code Writing Algorithms
Week 3	Introduction to Python, Control Statements Coding Algorithms
Week 4	Functions Typehints and Doc Strings
Week 5	Sequences Dictionaries and Sets, Collections
Week 6	Strings and Regular Expressions Files and Exceptions
Week 7	Debugging Testing
Week 8	Other Data Storage Formats Data Processing and Testing
Week 9	Numerical Arrays & Structured Data Processing and Plotting
Week 10	Object Oriented Programming Object Oriented Design
Week 11	OOP: Inheritance and DuckTyping Data Structures & Computational Complexity Analysis
Week 12	Stacks and Queues Linked List
Week 13	Trees - Binary Tree Graphs, BSF, DSF
Week 14	API Thanksgiving Break (2 days)
Week 15	Multi-threading and Multi-processing User Interface
Week 16	Project Workday Project Defense (TBA)