ECE 590D– Programming and Data Structures for Machine Learning

Course Information

Instructor: Javier Pastorino (*website*) **Email**: javier.pastorino@duke.edu

Office: 211 Hudson Hall **Phone**: 919-681-5515

Office Hours: times posted on Canvas. Appointments are required. Schedule through Calendly.

Teaching Assistants (TA):

Course Description

This course introduces programming and data structures using Python, as well as libraries used for data analysis (like *numpy*, and *pandas*). We discuss the principles of object-oriented programming and design and how to test programs. We also introduce machine learning applications by visiting the most common libraries (such as *scikit-learn*, and *TensorFlow*) while designing small-sized pipelines.

Course Objectives

Designing and developing software has many aspects. Many of these aspects are best approached without a specific language in mind. The concepts behind most Software Engineering, Program Design, and Program Construction techniques are language-independent.
This course introduces Python as a programming language. Python provides versatility in different programming paradigms, such as structured, object-oriented, and functional programming. We also aim to introduce the basic techniques of data structures needed to understand data representations, including abstraction and object-oriented programming.

Course Format

This course is offered as an in-person course, meeting twice a week on Mondays and Wednesday (75-minute sessions) and on Fridays for discussion sessions.
The course uses a mix of flipped-classroom and lecturing. Students are required to complete reading before class sessions. During sessions, a small review lecture will cover the most relevant topics, followed by practice exercises and discussions. Students will have the opportunity to work on teams during sessions, enhancing the learning experience. Discussion sessions dive deeper into the materials or cover other interesting topics in a reduced discussion space led by teaching assistants.

Prerequisites

This course is reserved for ECE graduate students and assumes no previous programming experience.

Learning Objectives

The course learning objectives are for the student to

- Comprehend the basis for program construction,
- Read and write code in Python,
- Analyze a problem and create a computer program to solve it,
- Use top-down design and abstraction to write clean, readable, fixable code,
- Design high-quality object-oriented programs by conducting program testing and debugging,
- Use Software Configuration Management (SCM) tools to version software,
- Understand data structures and explain the time complexity of operations,
- Understand different aspects of machine learning and outline simple machine learning pipelines discussing simple algorithms.

Textbooks

We will use two textbooks for this course. *Both textbooks are available digitally at Duke Library*.

- **Intro to Python for Computer Science and Data Science.** 1st Edition. By P. Deitel and H. Deitel. Pearson. *ISBN 13*: 978-0-13-540467-6. (**Required**)
- **Data structures & Algorithms in Python.** 1st Edition. By J. Canning, A. Broder and R. Lafore. Pearson. *ISBN 13*: 978-0-19-485568-4. (**Required**)

Additional readings may be provided during the course.

Required Tools

- Computer with a minimum of 8GB of RAM (16G recommended).
- Internet connection (high speed recommended)

Topics

- Introduction to Computers and Programs.
- Program constructs. Variables, assignments, arithmetic. Input and output. Primitive data types.
- Control statements. Conditionals and repetitions.
- Functions.
- Basic data structures. Lists, tuples, Dictionaries, and Sets
- Array-oriented programming.
- Strings, Files, and Exceptions.
- Testing and Debugging programs.
- Object-Oriented Programming. Classes,
- Recursion and complexity analysis (Big O).
- Introduction to Machine Learning and Scikit-Learn
- Introduction to Deep Learning and Keras over Tensorflow
- Implementing Data Structures. Stacks, Queues, Linked Lists, and Trees.

Course Schedule:

Week	Date	Topic	Reading	Assessment
1	Aug 26 (Mon)	Course Introduction		
	Aug 28 (Wed)	Developing Programs	Ch 1	
	Aug 30 (Fri)	*[Rec] Writing Pseudocode*		
2	Sep 02 (Mon)	***Labor Day - No Class***		
	Sep 04 (Wed)	Introduction to Python	Ch 2	
	Sep 06 (Fri)	*[Rec] Setting up the Environment*		
3	Sep 09 (Mon)	Control Statements	Ch 3	
	Sep 11 (Wed)	Functions	Ch 4	
	Sep 13 (Fri)	*[Rec] Writing programs with functions*		
4	Sep 16 (Mon)	Sequences. Lists and Tuples	Ch 5	
	Sep 18 (Wed)	Dictionaries and Sets	Ch 6	
	Sep 20 (Fri)	***[Rec] Test 1 -- Ch 1-4***		**Test 1**
5	Sep 23 (Mon)	Arrays - Numpy	Ch 7	
	Sep 25 (Wed)	Data Analysis - Pandas	Ch 7	
	Sep 27 (Fri)	*[Rec] Processing and Plotting*		PA 1 Due
6	Sep 30 (Mon)	Strings	Ch 8	
	Oct 02 (Wed)	Files and Exceptions	Ch 9	
	Oct 04 (Fri)	*[Rec] Searching data*		
7	Oct 07 (Mon)	Files - CSV	Ch 9	
	Oct 09 (Wed)	Testing and Debugging	TBA	
	Oct 11 (Fri)	*[Rec] Reading and processing data*		
8	Oct 14 (Mon)	***Fall Break - No Class***		
	Oct 16 (Wed)	Testing and Debugging	TBA	
	Oct 18 (Fri)	*[Rec] Testing and Debugging*		PA 2 Due
9	Oct 21 (Mon)	Object Oriented Programming	Ch 10	
	Oct 23 (Wed)	Object Oriented Programming	Ch 10	
	Oct 25 (Fri)	***[Rec] Test 2 -- Ch 5-9 & Testing***		**Test 2**
10	Oct 28 (Mon)	Introduction to Machine Learning	Ch 15	
	Oct 30 (Wed)	Introduction to Machine Learning		
	Nov 01 (Fri)	*[Rec] Running a pipeline from zero to hero*		
11	Nov 04 (Mon)	Introduction to Deep Learning	Ch 16	
	Nov 06 (Wed)	Introduction to Deep Learning		
	Nov 08 (Fri)	*[Rec] Deep learning*		PA 3 Due
12	Nov 11 (Mon)	Data Structures - Introduction to Abstraction		Project Posted
	Nov 13 (Wed)	Data Structures - Stack and Queue	DS Ch 4	
	Nov 15 (Fri)	*[Rec] Complexity Analysis*		
13	Nov 18 (Mon)	Data Structures - Linked List	DS Ch 5	
	Nov 20 (Wed)	Data Structures - Trees	DS Ch 8	
	Nov 22 (Fri)	*[Rec] Implementing Data Structures*		
14	Nov 25 (Mon)	Recursion and Complexity		
	Nov 27 (Wed)	***Thanksgiving Break - No Class***		
	Nov 29 (Fri)			
15	Dec 02 (Mon)	*No Class*		Project Due
	Dec 04 (Wed)	***Project Presentation***		
	Dec 06 (Fri)	***Project Presentation***		
16	Dec 09 (Mon)	***Project Presentation***		
	Dec 11 (Wed)	*Finals Week*		**Final Exam**
	Dec 13 (Fri)			
	Dec 16 (Mon)			

The previous schedule is tentative, and it may change. So please check the current one on Canvas.

Assessments

This course uses a mixture of formative and evaluative assessments. Formative assessments are more flexible and designed to help the student master one particular topic (e.g., loops). Evaluative assessments are less flexible and are designed to assess the knowledge the student gains on multiple concepts.
Formative assessments (homework) will be presented as small programming tasks 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 meet with TAs or the instructor as needed.
Evaluative assessments will include larger programming assignments (also automatically tested), a mini-project, and exams.

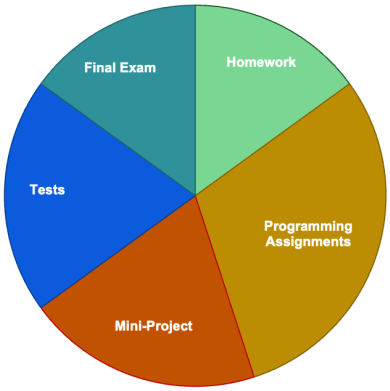
- **Homework**: small-sized programming tasks that will help students to apply the learned concepts. **Homework will be automatically graded**. The student will have access to a platform to test their homework; this system will test the submission and list any errors and the number of tests passed. Each homework may be graded as many times as needed to improve performance. ***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. Grading your homework after its due date will be penalized by subtracting points from your homework grade. 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 1-day 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. **Homework is to be completed individually.**
- **Programming Assignments (PA)**: Programming assignments will be similar to homework, with the main differences being the size of the task and a hard deadline. **PAs are to be completed individually.**
- **Exams**: There are three exams (two mid-term tests and a final exam). All exams will be on paper and include problem-solving and small coding exercises.
- **Team mini-Project**: teams of two students will propose and implement a mini-project that will target the implementation of a machine learning (ml) pipeline. This project will focus on coding a pipeline that gathers data and trains and evaluates the pipeline. Students will implement this mini project over a 3-week period. The project outcomes will be in the form of a poster and a short presentation to the rest of the class before the final exam.
- **Extra Credit**: There will be no extra credit assignments. However, as I consider attending and actively participating during class an important component of learning, I will provide up to 8% extra credit to those students who a) actively participate during class and b) do not miss more than three class sessions.

Extensions: I may allow extensions of evaluative assessments under special circumstances. The student must meet with me before the assessment is due.

Grading Policy

- The Final Grade will be distributed among the assessments following the table below.

Assessment Group	Grade
Homework	15%
Programming Assignments	30%
Team Mini-Project	20%
Tests (x2)	20%
Final Exam	15%



- The final Letter Grade will be converted using the following scale:

Communications

Announcements

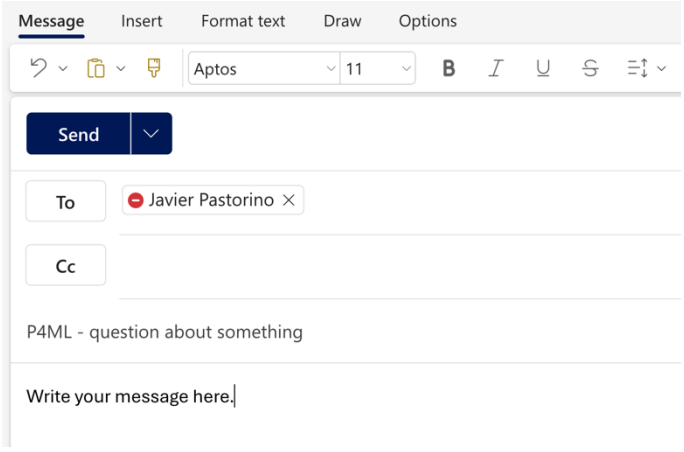
The course staff (instructors or TAs) will post all announcements in ED Discussions. Please make sure you have notifications on so you don't miss them.

Q&A with TAs/Instructors

We will use ED Discussion for questions and answers with TAs. Please mark questions that relate to your particular solutions as **private** so only you and the course staff. All other questions will be public to the course students; you may want to select "Anonymous" to hide your name from other students (course staff will still know who you are). This should be the only medium to reach out to your TAs for help outside office hours.

Contact your Instructor

I only use the **University Email system** (Outlook). If you need to contact me, please **email me**. *Do NOT use Canvas messages*. I usually reply to emails within 24-48 hours. My email is listed at the beginning of this document.
Prefix your email subject line with **`P4ML`** for a quick response.
However, to discuss exercises/problems, please schedule an appointment during my office hours, as I will usually not be able to answer those questions over email.



Academic Integrity

All members of the Duke community are expected to abide by the Duke Community Standard (DCS).

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.*

I encourage you to discuss the material, even form a study group, compare ideas for projects and other assignments, and work on problems you encounter. It is a characteristic of computing that discussions often help clarify problems and resolve difficulties —feel free to take advantage of this to improve your understanding of the material and to complete projects, but make sure you **create your own work**. It's important that you go through the program design, coding, and debugging processes yourself, or you will not be developing your programming skills and understanding.
Submitting someone else's intellectual work product, or generative AI's work product, as your own is unacceptable. This includes broadly available solutions online, from websites (e.g., coursehero.com or stackoverflow.com), online services (e.g., 24HourAnswers.com), tutors (e.g., Chegg.com), generative AI services (e.g., ChatGPT, Copilot or other LLM-based systems), online repositories (e.g., a shared Google Drive, GitHub, etc.), or other materials found online or otherwise not generated by you.

When working in teams, the phrase "Working Together" does not mean that one student does most of the work and others put their names on it! *That's technically plagiarism*.

If you have any questions about whether something you are doing may violate these policies, please come and talk to me before doing it.

We reserve the right to use automated similarity metrics to detect plagiarism in this course.

All students must create their work on their own!

Suspected violations of the DCS will be forwarded to the appropriate office depending on the program in which the student is enrolled –*Office of Student Conduct and Community Standards for undergraduates, graduate program administration for graduate students*– where they will be given an independent third-party review and resolved in accordance with university policies

Remember, the use of AI (ChatGPT, Copilot, other LLM-based systems, etc.) is not allowed unless explicitly stated in the assignment. Further using it will diminish your learning, which will have a direct impact on your tests.