LSCI 109: Intro to programming for Language Science

Winter 2025

Lecture time: MWF 10–10:50 am
Lecture location: SSL 155

Lab time: F 11–11:50 am
Lab location: SST 107

Instructor: Connor Mayer
Student hour time: TBD
Student hour location: SSPB 2211
Instructor e-mail: cjmayer@uci.edu
Website: https://canvas.eee.uci.edu/courses/69527

Course Description

This course provides an introduction to the fundamentals of programming in Python, with specific applications for the study of human language. By the end of this course, students should be able to:

- Read and understand basic Python code, including components such as variables, conditionals, loops, functions, and classes.
- Write Python code to solve basic problems.
- Read and write files using Python.
- Debug Python programs using the Python debugger.
- Demonstrate familiarity with the Linux command line
- Use Python libraries to solve more complex problems related to human language.
- Understand how to format, document, and test Python code according to best practices.
- Install and run Python on their own computers.

Course Format

This course consists of three weekly lectures and one weekly lab. The weekly lab will allow students to collaboratively work on the week's homework assignments, with the opportunity to ask questions of the instructor. Attendance at lectures and labs will not be graded. However, you are *strongly* encouraged to attend at the scheduled times so you can ask questions. If you are unable to attend lectures due to illness or other extenuating circumstances, you can access recordings of the lectures on Canvas. Lectures will be recorded live, and the recordings may not be optimal quality.

Prerequisites

Students should have taken LSCI 3 or equivalent. No prior programming knowledge is assumed. You should not take this course if you have taken ICS 31/32, or a similar introduction to Python.

Course Materials

Readings

You do not need to purchase any textbooks for this course. All materials (assignments, notes, readings) will be distributed through the course website on Canvas or GitHub Classrooms, or are available from free online resources. Our readings will be taken from Automate the Boring Stuff with Python (ATBS): A free, online introduction to Python.

Software

Programming examples and assignments will be administered through GitHub Codespaces/Classrooms. The only requirements to use these tools are a computer with a working internet connection and a GitHub account. In Week 10 you will learn how to install Python on your own computer.

Requirements and grading

Component	**Proportion of grade**
Eight weekly assignments	50%
In-person midterm exam	20%
In-person final exam	30%

Weekly assignments

There will be eight weekly assignments that will make up a total of 50% of your final grade. There will not be an assignment due in Week 1 or Week 6 (midterm week). Assignments will consist of programming problems or problems that ask you to read code and answer questions. Assignments will be administered via GitHub Classrooms.

Midterm and final exams

There will be an in-class midterm on Monday of Week 6 and an in-person final exam during exam week. The final exam will be cumulative. These exams will ask you to write Python code or read Python code and answer questions. You will not be allowed to use computers or other electronic tools during the exams.

A note on the use of AI tools

Large language models such as ChatGPT are quite adept at solving the kinds of simple programming questions that the assignments and exams will contain. Additionally, there are AI tools such as GitHub Copilot that can assist with programming. Although these tools can be useful, **I strongly encourage you not to use them in this class**. There are three reasons for this:

- The midterm and final exams will be pen-and-paper, in-person exams. This is intentional. You will not be able to rely on AI tools, and if you are not able to write Python code without their assistance, your grade will suffer.
- If you do not master the material in this course, you will not be able to critically evaluate the code produced by AI tools (e.g. to identify bugs, to determine whether the code does what you requested, etc.). There is no guarantee that code produced by these tools will be correct, and you put yourself at their mercy if you are not able to understand their output.
- It is academically dishonest to take credit for code you did not write.

We will have a lecture in Week 2 that goes over these AI tools, how they can be useful to you, and potential pitfalls you need to look out for.

Grading policies

Letter grades are calculated from numeric grades as follows:

Numeric grade	Letter grade
$\geq$ 97%	A+
$\geq$ 93%	A
$\geq$ 90%	A-
$\geq$ 87%	B+
$\geq$ 83%	B
$\geq$ 80%	B-
$\geq$ 77%	C+
$\geq$ 73%	C
$\geq$ 70%	C-
$\geq$ 67%	D+
$\geq$ 63%	D
$\geq$ 60%	D-
$<$ 60%	F

Grades will only be changed for clerical or arithmetic errors. The exception to this is that I reserve the right to scale final grades if I think it is necessary. I will only scale grades up: that is, your final grade can only *improve* as the result of scaling.

Getting help

- The first place you should seek help is using the discussion board on Canvas. If you have a question, it's likely that someone else has the same question. Posting on the discussion board allows everyone to see the answer. I also strongly encourage you to try to answer your peers' questions on the discussion board. This gives you valuable practice engaging with the course material, utilizing online resources, and synthesizing information, all of which will serve you well down the road.
- The second place you should come for help is my student hours. Please feel free to drop by as frequently as you like, even if you don't have any specific questions and you just want to work on an exercise or chat.
- If neither the discussion board or student hours are viable, you can email me with questions or concerns. I will reply to you within 24 hours.
- In certain circumstances I may be willing to arrange a meeting with you outside of normal class times and student hours. For the sake of my schedule (and yours!), please consider this a last resort, and do your best to seek help using the resources in the previous three points.

Academic integrity

All students are expected to adhere to the UCI Academic Dishonesty Policies (for more information, please visit `https://aisc.uci.edu/students/academic-integrity/index.php`).

Disability

Any student requesting academic accommodations based on a disability is required to apply with Disability Service Center at UCI. For more information, please visit `http://disability.uci.edu/`.

Course Schedule

Week	*Dates*	*Topic*	*Homework*	Readings	*Notes*
1	1/6 – 1/10	Linux and Python basics		ATBS Ch. 1	
2	1/13 – 1/17	Control flow	HW 1	ATBS Ch. 2	
3	1/20 – 1/24	Data types	HW 2	ATBS Chs. 4-6	*No class on 1/20*
4	1/27 – 1/31	Functions and modularization	HW 3	ATBS Ch. 3	
5	2/3 – 2/7	Reading and writing files	HW 4	ATBS Ch. 9	
6	2/10 – 2/14	Exceptions and debugging		**Midterm: 2/10**	
7	2/17 – 2/21	Object oriented programming	HW 5		*No class on 2/17*
8	2/24 – 2/28	Using modules and libraries	HW 6		
9	3/3 – 3/7	Code quality, regular expressions	HW 7	ATBS Ch. 7	
10	3/10 – 3/14	Running Python locally, review	HW 8	ATBS App. A, B	
11	3/17 – 3/21	Exam week		**Final exam: 3/17 10:30 am**	

Acknowledgments

Thanks to Niels Dickson, Nathaniel Imel, Yanting Li, and Weijie Xu for their contributions to the design of this course.