Syllabus

Subject to update during the first week of class.

INLS 560: Programming for Information Professionals

Instructor: Prof. Elliott Hauser

Official Course Description

INLS 560: Programming for Information Professionals (3 credits)

An introduction to computer programming focusing on language fundamentals and programming techniques for library and information science applications. Emphasizes problem-solving through the development of practical applications that include text processing, file handling, user interfaces, and web data access. Offered fall and spring.

Overview

This course is an introduction to Programming as a skill, a discipline, and a profession for graduate students. We'll dive into hands-on programming from day one and progress to evaluating, using, and contributing to open source libraries and frameworks. We'll focus equally on reading and writing code. Students will leave the course with real skills, an ability to learn new programming technologies, and an understanding of how to incorporate open source code into their projects. It will serve as an appropriate foundation for students seeking a career in programming and indispensable background for any information professional needing to evaluate, communicate with, or work with programmers or code.

Objectives

At the end of this course, students should:

- Have the skills required to solve problems by creating and modifying programs and systems, using modern programming tools.
- Have the knowledge of basic programming concepts, their appropriate usage, and how and where to learn more.

- Have an attitude of confidence when reading, writing, or discussing computer code

Students will be prepared to integrate these skills and knowledge with other SILS courses in Databases, Web Development, Text Mining and others. Students will also be well versed in common tools and workflows used by developers and development teams.

Real-world Tools

This class will introduce you to and utilize many of the tools that real development teams use for collaboration, coding, and communication. This will include Github, Jekyll, text editors, videoconferencing, persistent online chat, and screensharing. By the end of the course you should be prepared to collaborate on code effectively with others.

Flipped Design

I'd like to emphasize the flipped design of the course. Almost all of the content consumption (readings, videos, etc) will be pushed outside of the class sessions. During class we'll focus on completing in-class exercises almost every day, which will be due by midnight if you need extra time (see below). I'll help answer questions and provide clarification, but understanding the concepts and techniques in the content I assign you will be essential for completing the exercises in class. **This is especially important in this accelerated Summer session.** Failure to do this will be a serious hindrance to understanding programming, completing the take-home exercises, and doing well in the class.

Office Hours

I will be available for office hours by appointment via video and/or text chat. I will be free an on campus before many of our class sessions. Email me or grab me during class to schedule a time to talk.

Student and Instructor Expectations

Honor Code

I take Honor very seriously. Trusting all students to follow the honor code allows for the extremely open design of my classes. The collaborative learning possibilities of an open classroom are unmatched, and Honor and trust play a central role in enabling this experience for me and my students alike. This also mimics the larger professional context that we all operate in, where personal integrity is fundamental.

All students should familiarize themselves with the University of North Carolina at Chapel Hill Honor Code on the Office of Student Conduct's site if they have not already done so.

For this course in particular, collaboration with other students will be expected in many instances, but work that is assigned to you as an individual should be yours alone. Some of the exercises in the course have solutions or other relevant resources available on the internet. Unless I specify otherwise, students should complete exercises as far as they can without these resources and then acknowledge the resources and how they were used in their exercise writeup. In group projects a combination of git commit histories, writeups and/or project documentation should attribute individual and collaborative work.

If you have any questions about what's allowed, please contact me immediately. Good faith mistakes are no problem; tell me about them and we'll move on. I've never had a problem with student cheating or dishonesty and hope I never do. That said, if I discover bad faith, intential cheating on any assignment I will not hesitate to report the violation and advocate strenuously for the harshest punishements allowed. Such behavior has no place at UNC.

Attendance & Attention

Students should attend every class and give it their undivided attention. Use of non-coding websites or social features is prohibited. There will be many opportunities for interaction, but during class time please pay attention. Students that do not follow this will be asked to leave class.

If you must miss class please let me know as soon as possible, check the class website for any class notes I post that day, and contact a fellow student for a recap. You will still be responsible for any in-class exercises completed that day.

Late Work

In general, lateness is not tolerated in the professional world, and that's the same standard I'm holding you to. Late work without prior approval will not be accepted. Exceptions will be rare. Contact me as soon as you think you may have difficulty meeting a deadline with your rationale for an extension. Turning work in early is good for all involved, for this class and your career beyond, so start early.

Diversity & Inclusiveness

I take the happiness and wellbeing of all students very seriously and share the university's commitment to diversity (available here: http://diversity.unc.edu/our-committment/div-values/). The field of programming has historically struggled with diversity but we have the opportunity to help build a more inclusive future for the profession. I expect students to join me in welcoming different perspectives and backgrounds in the course.

Assignments

Cultural Immersion Activities: Like other cultural products, digital technologies require an immersive experience for effective learning. After a vocabulary of about 300-400 words is developed, students can begin teaching themselves languages, and a roughly similar threshold applies to programming concepts and technologies. Students in INLS 560 will immerse themselves in the culture surrounding programming in these ways:

- Attending **one in-person programming meetup** over the course of the semester. This must be with an off-campus group such as TriPython, our local Python user group.

- Extra Credit: Contributing to **one open source project** in the form of a Github issue or pull request. This should be an active, independent (i.e. not related to this class). See me if you're not sure your project would qualify.

These experiences will combine with class assignments, readings, and in-class exercises to deepen students' interest and facility with the concepts and skills we'll learn. Students who are unable to attend the in-person meetups may propose a comparable virtual community interaction in its place.

Exercises (In-class and Take-Home): Like any skill, frequent practice is key to coding. Almost every class will feature exercises that build on assigned readings. These assignments can be completed and submitted during class, or students can submit them before midnight if they need extra time. Exercises may involve pair programming and I will circle through class to resolve any confusion and share important concepts with the group. In addition to in-class exercises, there will be requent assignments of exercises from our texts and larger project-style assignments. Submission instructions will be included with each assignment. These in-class and take-home exercises are the largest component of your grade.

Readings: Assigned readings, lectures, or exercises should be completed before the indicated class. In most cases their content will be essential to completing the in-class exercises. In technology we'll often encounter terms or project names that we haven't heard of before (myself included). Students will be encouraged to keep track of new terms, research them, and bring them up in class.

Website chat: Our class website has a Gittr chat room available from the upper right of every page. This is available for backchannel Q&A during class. Participation in this chat will count as classroom participation. Students are expected to conduct themselves in this room as they do during class, keeping content appropriate and relevant. Click here to open the chat

Final Project: The semester will culminate in a fully developed program or system of the student's creation. More details on the final project will follow.

Extra Credit: There will be infrequent opportunities to earn small amounts of extra credit. This will usually involve doing things I might normally do, like deploying our class blog, tidying up code I write for this class

for wider consumption, or helping moderate our forums.

Educational Privacy: It is your FERPA right to keep your educational record, including enrollment in any specific class, private. Assignments that require public submission may be completed pseudonymously if you would like to exercise this right. This choice will not affect your grade.

Materials

Primary Textbook: We'll be using a new version of Charles Severance's excellent Python for Informatics. This book is being revised to support Python 3, and I'm actively helping. We'll. I'm working on getting an HTML version up so that you'll always have access to the latest version. Since the textbook is open source you can make contributions or fixes to any errors you find with the Github-based collaboration skills we'll learn in the class.

Primary Video: I recorded a video series with O'Reilly this year and we'll use the videos extensively during a section on making interactive games. The video is available for purchase from O'Reilly or available with a subscription to Safari Books Online. Students will need access to the video lessons throughout the course.

Readings: As often as is possible, materials used in this course will be drawn from freely available resources on the Internet. This mimics the working environment of most programmers.

Software: The course will utilize free software almost exclusively, much of it open source. Most, if not all, exercises will be completed in browser using Trinket.io.

Hardware: Students will need their own laptops able to run Chrome or other modern web browser. iPads are NOT recommended for coding. If you **must** use an iPad, I highly recommend purchasing and external keyboard. Or, since iPad accessories are already pretty expensive, purchase a $150 Chromebook. Coders need keyboards.

Additional Materials: There is a wealth of excellent Python material on the Internet. Here is a selection of some resources students may wish to consult for additional perspectives or information on some of the concepts we'll cover:

- Coursera's Python for Everybody by Dr. Chuck. Yes, the same Dr. Chuck who wrote our textbook. Some of these older videos use Python 2 syntax but regardless they are an excellent option for students who want a more multimedia experience. The Week chunks roughly correspond to the first few chapters. Highly recommended for any time something didn't 'click' in class and you want to hear it again. Free.
- Related, Dr. Chuck has a YouTube playlist with lectures and walkthroughs of much of the course content.
- Automate the Boring Stuff by Al Sweigart. Al's book was a close contender for the text of this course.

Free online, and you can buy the book. Highly recommended if you intend on developing Python proficiency after the course.
- Google's Python Class by Nick Parlante. This is an intensive 2 day course for programmers with experience in a language other than Python. its quick pace means that it's best as a refresher or extender after we've covered something. The link is to a version that I made interactive with Trinket, so it should be quick and easy to pop in and do a section.
- Python's official documentation. You can use this during activities that restrict you to class materials by using a special google search restricted to just this domain.

I'll occasionally pull readings from these sources or refer you to them.

Grading

Weights of course components are as follows:

- Cultural Immersion Activities: 10%
- In-Class and Digital Participation: 15%
- Midterm: 0%
- Exercises: 50%
- Final Project: 25%

Note that the SILS grading policy is based on the University Grading Policy. SILS uses the following graduate and undergraduate grading scales:

Graduate		Undergraduate
H	Clear excellence	A
P	Entirely satisfactory	A-, B+, B, B-
L	Low Passing	C+, C, C-
F	Failed	D+, D, F

For the purposes of this course, satisfactory performance (the 'P' grade) involves completing all of the assignments and showing personal competence with the material. Clear excellence (the 'H' grade) means active participation in our classroom community, mastery and extension of the material, and effective collaboration with other programmers, either inside or outside our class.

Acknowledgements

This course and its content has been deeply influenced by my interactions with and the input of others. I’d like to thank Denise Anthony, David Gotz, Hilmar Lapp, Stephanie Haas, Brian Marks, Greg Wilson, John D. Martin III, Rob Capra, Alan Dipert, Ryan Shaw, Grant McLendon, Al Sweigart, and Charles Severance for a wide range of help.