**Introduction to Programming for the Life Sciences, Fall 2023**

**Instructor:** Seth Syberg

**Contact:** ssyberg@rockefeller.edu

 **Teaching Assistant:** TBD

**Contact:** TBD

 **Meeting Times**: 9 meetings (a mixed schedule of virtual and in-person), roughly every two weeks starting September 13th until late January 2024, 3pm – 5pm during the Fall, there may be a schedule adjustment in January.

Tentative Meeting Schedule:

|  |  |
| --- | --- |
| 9/13/2023 | Remote |
| 9/27/2023 | Remote |
| 10/18/2023 | In Person |
| 11/1/2023 | In Person |
| 11/15/2023 | In Person |
| 11/29/2023 | In Person |
| 12/13/2023 | In Person |
| 1/3/2024 | In Person |
| 1/17/2024 | In Person |

**Room:** Weiss 301

**Office Hours:** Every other week, details TBD

**Pre-course Assignment.** An assignment will be due *before the first class meeting*. Details will be emailed to you shortly after enrollment.

**This is a *very interactive course.*** Core to all software development is iterative and incremental progress. You will get stuck and encounter bugs that require outside help, you will receive feedback requiring timely changes to your work, and sometimes assignments will change mid-stream\*. This means frequent (sometimes *more than daily)* interactions with the course will be necessary.

**Approximate Time/Communication Commitment**: cumulatively 4 to 8 / week (though as mentioned above, often spread out across the week). I highly suggest only taking this course if you have **both** the time to commit to it and a schedule that allows for frequent/daily interactions.

 *\*As is the nature of programming, the course is always evolving, bugs in the labs and assignments are being found and solved, hints and changes are common, so staying abreast of updates and announcements are crucial for success.*

**Computer Requirements**: It’s strongly suggested you have a relatively new / well-functioning computer. Some of the work we do in this class is processor intensive and we will be using recent versions of software that may require recent versions of your operating system. Rule of thumb, if your computer is *more than 5 or 6 years old* you may struggle in this course. Most instructions will be given on Mac OS, but you should be fine using a recent version of Windows or Linux.

**Acknowledgements:** A HUGE thank you to my dear friend Dr. Josh Hug (the instructor for this course in 2013 and 2014) for providing us with his course materials that can be found, modified by myself, throughout this site!

**Course Blurb.** Here's a neat fact you may not know: Once you know how to write code in a single programming language, you can automate any task in the universe that is possible to automate (including perhaps even conscious thought, though that level of program will be beyond the scope of our course).

You may already perform such tasks manually, or perhaps with the assistance of other people's special purpose tools that might not do exactly what you want. These tasks include things like massaging data in spreadsheets into a more useful format, making data plots, or searching for interesting patterns in data (sequence alignment, cell identification in microscopy images, etc.).

In this course, we'll learn how to build tools to perform such tasks from scratch, as well as how to modify existing open source tools. This course will focus primarily on learning to write general purpose programs in the Python programming language.

**Prerequisites.** I'll be assuming no prior programming knowledge, but completion of the pre-course assignment is required.

**Course Format.** The course will consist of the following main components:

1. **One two-hour meeting every other week:** I attempt to minimize lecturing and focus more on live coding/demos and hands-on programming exercises.
	1. **Office hours (1-2 hours) on weeks we don’t meet:** For those who need additional personalized guidance.
2. **Ongoing programming challenges (labs & assignments):** To be completed outside of class.
	1. **Labs:** One or more sets of practice questions per week, reinforcing and extending concepts from lectures.
	2. **Assignments:** Larger questions requiring one to synthesize, apply, and extend concepts from lectures and labs.
3. **Online discussion / live help**: For assistance with challenges and general questions, we will make extensive use of slack (a “chat” program) where students will post questions, help fellow students, and engage in discussion.
4. **Final Assignment**: You will create a project that is something you can use in your own research, utilizing nearly every major concept learned in class. Due at the end of the course.

**Textbooks.** There are no required books for this course, though we will make use of many online references which will be posted in slack, labs, and assignments throughout the course.

**Course Website:** For this class, we will be relying very heavily on the class website offered through the Coursemology platform. This is where all materials, labs, assignments, and announcements will be posted, and where many labs will be completed. Additionally, we will be using Github for all assignment submissions and feedback.

**Slack**. With very little exception, most questions in this course should be directed to our shared slack workspace in a ***public channel*** where they will be answered by me or your fellow classmates. More often than not, a given student's question is one I get from multiple students, and the answer may prove useful to everyone.

If you direct message (DM) me on slack or email me directly I will often refer you to the shared slack channels before answering your question (if I deem it appropriate).

**Do not, however, post your solutions (or partial solutions) for any assignments to slack *in answer to another students question*.** To avoid posting solutions to the slack you may produce an unrelated example that exemplifies your answer for safely posting in slack.

Guidelines for Getting Help on Slack

Before posting to the slack - please think about what me and your fellow students will need to help you! For example:

* If you are posting about a failing test (test suite or Coursemology) always include any/all output you are getting
* If you are getting a python error - post the error
* If you are having a file system issue - post a screen shot
* The more information the better!

**Final Project.** Create a project that is something you can use in your research utilizing nearly every major concept learned in class. If you're unable to come up with something for research, then something that is directly in your field of study/specialty.

Components: (***No late work will be accepted!)***

1. Students will first submit aproposal detailing the problem or set of problems they’d like to solve. At minimum, it should include all program inputs, outputs, and a summary/description of the project.
2. The completed project will include your project code, as well as test client code, and a video verifying the performance of the project code.
3. A short report will be submitted with the completed project, describing the current state of your project, the outputs of the program, and overall conclusions.

Minimum Requirements:

Your project must be substantial (i.e. hard). 30 lines of code that meet all the technical requirements *is not sufficient to pass.*

Your project must also make use of nearly every major construct learned throughout the course:

* Loops
* Conditionals
* Advanced data structures (lists, dictionaries, classes)
* Plotting or an equivalent visual component
* External files / external data of some kind

**Collaboration Policy.** Students are welcome to discuss labs and assignments but should do their own work.

**Grading and Late Work.** This is a pass/fail course generally based on a 75% or higher in the course. A crucial component to the learning process in programming is reviewing well written code, and as such, it’s important to release solutions in a timely fashion. This makes accepting late work difficult or impossible and *missing deadlines can quickly lead to a failing grade*.

Additionally, satisfactory completion of a final project is **also required** for passing the course.

**Software.** We will also be making extensive use of Visual Studio Code IDE (integrated development environments) to complete all coursework.

**Auditing.**  This course is also available for audit (non-credit). Auditors have two participation options a "full audit" and a "normal audit". In order to receive help or feedback during the course, "full" auditors are required to participate on the same level as fully enrolled students (attending lecture and consistently completing assignments and labs). Once an auditor stops participating at this level they are switched to a "normal audit" and are welcome to sit in on lectures, but we no longer assess their work or provide help.

**Instructor Bio:** Seth Syberg has been a computer science and information systems lecturer for a number of years, working at many community colleges and universities, most recently at the University of Washington in Tacoma. His breadth of teaching spans from introductory programming up through graduate level computer science courses. In addition to his work as an instructor, Seth has over 20 years of experience working in the tech industry and is currently the Technical Director at Type/Code. He completed his Master’s of Computer Science at Brown University in 2007 and began teaching at Rockefeller University in Fall 2015.