PY101: Intro to Programming in Python
Section 1— 2021

Instructor Information

Name: James Murphy, Ph.D.
Email: See contact page
Office Hours: 2 hrs/wk time TBD

Class Information

Dates: Saturdays & Sundays from January 9, 2021 to March 7, 2021 (9 weeks, 18 classes)
Time: 12:00PM (noon) to 1:30PM, US Central Time
Classroom: Mix of Jitsi (video conferencing) and Slack (chat client)

Course Description

This course is a project-based introduction to programming in the Python programming language focused on real-world practicality. Students will learn how to program in the latest version of Python (3.9). Students will learn fundamental programming concepts such as branching, iteration, abstraction, functions, mutability, hashing, ordering, recursion, and object oriented programming. Additionally, students will learn how to use tools and services to enable themselves to develop and collaborate with others. These include learning to use version control with git and github, interactive debuggers, virtual environments and pip, type checkers, profilers, testing frameworks, and popular Python packages. The course may include some theoretical components, such as boolean logic and some basic algorithms, but only to the extent required for practical usage in code or as requested by the students.

Prerequisites

There are no formal prerequisites for this course, but students are expected to use their own computer with a stable internet connection suitable for video conferencing, and to have a general familiarity with using the internet, email, installing programs, and managing files and folders on their computer. Any previous math or programming experience will certainly give students an advantage, though such experience is not necessary.

Course Objectives

After this course, you should be able to...

- To read and understand most Python code.
- To read and understand Python documentation and APIs.
- To know how and where to find answers to Python questions you have to further your own understanding.

- To write small to medium sized programs in Python for your own personal projects, objectives, or business needs.
- To install and use other packages or projects in your own projects.
- To use version control to track changes as you develop a codebase further and further.
- To collaborate with other developers using github or other git-based frontends, including the ability to track issues and make releases.
- To use a debugger to find, understand, and correct errors in your code.
- To write tests for your code.
- To use a static type-checker to type-check your code.
- To use a profiler to measure code performance and understand where your code spends the most time.
- To use Python to analyze basic datasets.

Video Recordings of Classes

All lectures will be video-recorded and made available to students through a private link for the duration of the course, unless technical issues prevent the recording process. Students consent to the fact that that their image and audio may appear in the video recordings if their camera or microphone is turned on. Students are not permitted to redistribute or otherwise make available the class video recordings. The video recordings are intended for educational use only and will never be shared with third parties unless compelled by law.

Class Attendance and Participation

Attending each lecture is essential for keeping up in the course. You are expected to attend every lecture, and to participate and ask questions when you have them. If for some reason you are unable to attend a lecture, then you are expected to watch the lecture video. Note, however, that watching the lecture videos is not in any way a substitute to attending the lectures, as you will not be able to ask questions to the video. Please let the instructor know if you missed or plan to miss a lecture so that we can make sure that you don't fall behind.

Grades (or lack therof)

There are no grades in this course. You will complete your projects and homeworks because doing so will improve your understanding and make you a better programmer. Having interesting projects to show on your resume, or having a useful tool for your business will do much more for you than a grade-point average. Conversely, worrying about grades causes undue stress on students without necessarily improving their understanding or performance.

So how will I measure how well I am doing?

All assignments and projects will be split up into subparts that are rated based on their difficulty

and time consumption: level 1, level 2, and level 3. Depending on how much time and effort you wish to put into this course, you will decide what percentage of level 1, level 2, and level 3 tasks to complete. You can then measure how well you're doing by comparing to your goal. There are no wrong answers on what percentages to choose, however the more time and effort you put in the better programmer and problem-solver you will become.

You can always ask the instructor for guidance, but here are some hypothetical example students and their percentages:

- Alice is highly motivated but does not have a great background in math, so she chooses to do 100% of the level 1 tasks, 75% of the level 2 tasks, and 20% of the level 3 tasks.
- Bob has a decent math background but is also balancing a full-time job, so he chooses to do 50% of all three levels of tasks.
- Charlie already knows Java and is thinking of making Python programming into a career choice, so he chooses to do 100% of the level 1 tasks, 100% of the level 2 tasks, and 90% of the level 3 tasks.

Academic Integrity

Academic integrity is of the utmost importance. All work turned in by a student must be the student's own work, except for portions contributed by other students during a group project. Outside of group work, students should generally not be reading each other's code. However, discussing code, homework, and the fundamental ideas behind them is actively encouraged.

We also understand that in programming, using other people's code, projects, or libraries within your own is not automatically dishonest. In fact, many authors encourage others to use, modify, or distribute their code. Maintaining academic integrity in this kind of open source world usually comes down to complying with the licenses of the original project, and giving proper attribution. If you are unsure, you are always welcome to ask the instructor whether a particular use of someone else's code within your project would be acceptable.

Communications Policy

Email is the primary vehicle for official communication with the instructor. All email communication in this course should be done using the email account that you signed up to the website with. Please put PY101 in the subject line of your emails.

Due to the online nature of the course, classroom communications through Slack, Jitsi, and Github are also acceptable for non-performance and non-personal-information related communications.

Respect Policy

The classroom should be an open and welcoming environment for all students. Discrimination of any kind will not be tolerated. The instructor will come prepared to teach and help the students understand the course material. The instructor will answer students' questions thoughtfully and without judgment. The instructor will be open to feedback. The students will be on time and pay attention during class. The students will come prepared, having done their homework and other projects. The students will not be disruptive, and will take any calls or texts after class is over or off-camera and away from their microphone. The students will ask questions to improve their

understanding. The students will work with each other and help each other and allow each other to make mistakes so that they can learn from each other.

Tentative Schedule

The following is a *tentative* schedule for the course.

- Class 1, 1/9: The IDE, running and debugging programs, variables, basic datatypes and operations, writing tests
- Class 2, 1/10: User input, functions, scope, branching, type annotations, git and project setup
- Class 3, 1/16: Iterables and iteration, loops, more on debugging
- Class 4, 1/17: More iterables, generators, comprehensions
- Class 5, 1/23: Making your own data types, classes and methods
- Class 6, 1/24: More on classes and dataclasses, first graphical program
- Class 7, 1/30: Graphics events, time and randomness
- Class 8, 1/31: Graphics animation, timers, tick-based movement, real-time movement
- Class 9, 2/6: Reserved for review and questions
- Class 10, 2/7: Object oriented programming
- Class 11, 2/13: More about object oriented programming
- Class 12, 2/14: Mutability, order, hashing, sorting, profiling
- Class 13, 2/20: Recursion
- Class 14, 2/21: Data science, numpy, scipy, matplotlib
- Class 15, 2/27: More on data science, pandas
- Class 16, 2/28: Web scraping, requests
- Class 17 & 18, 3/6 & 3/7: Review or special topics based on class interest (e.g. flask, keras, or more datastructures and algorithms)