Introduction to Python

Course Description

Python is a powerful, versatile cross-platform programming language that has a strong presence in diverse software engineering disciplines including web development, information security, network scripting, data science, and embedded systems. While Python itself may be a deceptively simple language, the vast array of frameworks and tools available for use across a variety of specialized fields make it a formidable tool in the arsenal of any technologist with areas of focus from Machine Learning to Cybersecurity.

This course will provide a pragmatic and hands-on introduction to the Python programming language, with a focus on practical applications and projects, rather than theoretical topics. Students will design and build software to solve problems from various disciplines each week using Python. As the course progresses, students will learn to work with packages, data structures, object-oriented programming, and tools for data science and cybersecurity.

Learning Objectives

By the end of this course, students will achieve:

- Familiarity with the Python programming language and associated development tools (IDEs, pip, Jupyter notebooks)
- Hands-on experience using various Python data structures, and reading and writing files in Python
- Experience with Object-Oriented design in Python
- Experience leveraging powerful 3rd party packages for Python, as well as creating your own custom Python packages for reuse and distribution
- Exposure to Python tools and techniques used to solve problems in real fields such as data science and cybersecurity
- A working understanding of when to use Python, as opposed to other programming languages/tools

Environment

We will be using Canvas as our LMS for this course. A few notes to help get you started:

- You can access canvas at https://canvas.cmu.edu
- Go into your personal Notification settings and alter them as you desire. For example, I recommend enabling email notifications when a new Announcement is posted to the course (this does not seem to be the default setting).

- Check out the messaging and discussion forum features - these will be a primary source for contact and asking questions

Textbooks and Resources

There are no required textbooks for this course, but the following recommended books will give you valuable references for Python programming:

- Zed A. Shaw, Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code (**strongly recommended**)
- Mark Lutz, Learning Python, 5th Edition (**recommended**)

Additionally, like any programming language, Python has a wide variety of comprehensive online resources available. A few that are particularly relevant to this course are listed below:

- The Python Language Documentation
- Scikit-learn Documentation
- NLTK Documentation

Lectures

I've designed the content for this course to be distance-first, meaning we won't be relying on on-campus lectures, multiple hours in length. Instead, you will find links to videos covering each requisite topic in the course schedule below, organized by week. All course videos can also be found in this playlist:

https://www.youtube.com/playlist?list=PL4P5xWlkyQGuVJ9WVwH4EpabK6AEddSJa

Additionally, all example/demo files used in these videos are also linked, in either .py or .ipynb format. You are responsible for all material covered in these videos and the accompanying resources. The video content directly ties into the lab exercises, homework assignments, and final projects.

Office Hours

Office hours will be held weekly via Zoom Meetings, on the date and time listed on the course calendar. The link to attend is: http://bit.ly/cois-office

Grading

The only way to learn programming is to do it. As such, this course will focus heavily on writing Python code to solve problems. There will be no exams, only coding assignments. These will include weekly labs, regular homework assignments, and a final project.

Grade Distribution

Weekly Programming Projects	50%
Lab Exercises	20%
Final Project	30%

**One weekly programming project (not lab, not final project) can be up to a week late without penalty. You can only use this extension once, and must notify me when you invoke it. Any other late homework will be penalized 10% per day, and late labs will receive no credit.*

Weekly Programming Projects

Weekly projects are due each week by Sunday night (11:59pm ET). These are programming projects, where you will write a small application to perform a specified function.

Note: Homework **must** be submitted by the deadline posted. No late submissions will be accepted.

Project Grading

Total Points possible for each homework assignment: 10

Correctness / Meeting Specified Requirements: Deductions will be made at the discretion of the grader.

Running without error: **2 point** deduction if the .py file you submit does not execute correctly (i.e. there are errors thrown when the grader execute the main script).

Style: Coding style is very important. Starting with Homework 3, you will receive a deduction of **1 point** if your code does not get 10/10 in pylint.

- Graders also have discretion to deduct up to **1 point** for lack of:
    - consistent coding style
    - appropriate use of variables

- appropriate use of functions
- good commenting
- good choice of variable names

Labs

There are one or more lab exercises each week. All weekly lab exercises will be due at the end of each week (Friday evening, 11:59pm ET). The three OLI modules due during the first two weeks of the course will also count as labs, for grading purposes.

Lab Grading

Labs will be graded as a participation score. Lab activities must be completed and turned in by the date specified to receive credit. There are N lab activities in the course, and your lab grade will be calculated as the percentage of labs you have completed, i.e.

Lab grade = [# of labs you complete]/N.

Academic Integrity and Collaboration

Collaboration is not permitted. All code you submit for this course must be **written by you, and you alone**. You may discuss problems and approaches with classmates, but it is a cheating violation when code is copied or shared. Do not copy and submit code from classmates, former students, github.com, stackoverflow, or any other source.

Second, you must not provide answers or code to other students. As part of this, **do not** post your project code for this course on github or any other public code sharing site.

If a student is caught sharing his or her work with another, a failing grade may be assigned to that student for the course. Likewise, if a student uses another person's work and submits it as his or her own, a failing grade may be assigned for the course. Any case of suspected cheating will be brought to the Dean's attention. At that point, the policies of the Heinz College on cheating will be followed.

Stackoverflow and Other Programming Resources

Let's be crystal clear: You are professional programmers. Professional programmers use the internet (Stackoverflow, blogs, github, etc) to find solutions to programming problems roughly every 5 minutes. I'm not asking you to ignore the perfectly valid and highly beneficial resources at your disposal when programming for this course; I want you to develop programming skills that *include* real-world research techniques. However, I am asking three things:

1. **DO NOT** copy and paste code directly from Stackoverflow, or any other internet resource. This counts as cheating, see above section of the syllabus. Instead, use solutions found on the internet as reference implementations to study and help craft your own solution.

2. **DO** attempt to solve all problems by yourself before looking up external resources - this exercise will engage your analytical mind, and force you to understand the problem better. Ultimately, the value you get from this course is up to you, but I'd like to see every student learn every topic to the best of their ability.

3. **DO** completely understand any code you write. This means that if you are stuck and go to the internet for help, you do not just transcribe code, you study and understand an external solution to the extent that you can implement the same algorithm on your own. Trust me, this approach make a huge difference in improving you as a software engineer. Also, if you ever have a question on a grade for a project, I will ask you to explain your code to me. If you can't, I can't trust that you actually solved the problem.

Again, it’s up to you to get all the value you can from this course. This material is extremely valuable to software engineers in the field, and will serve you well if you give it all you've got.

Schedule

Dates	Topics	Lessons	Additional Resources	Assignments Due
Week 1 [8/26/19 - 8/30/19]	• Welcome (video) • Why Python? (video) • Setting Up Your Python Development Environment (video) • Computational thinking (OLI) • Variables, operators, basic types (OLI) • Basic string operations (OLI) • Functions (OLI) • Importing and using modules (OLI)	[1] Welcome (Links to an external site.) [2] Why Python? (Links to an external site.) [3] Setting Up Your Local Development Environment (Links to an external site.) [4] OLI Module 1: Introduction to Programming with Python	Shaw (textbook), exercises 0-4, 18-19, 21, 27-28 Shaw appendix: Command Line Crash Course	• OLI Module 1 • Lab 0 • Homework 1

Week 2 [9/2/19 - 9/6/19]	• Loops (OLI) • Ranges (OLI) • Lists, nested lists (OLI) • Advanced string manipulation (video) • Format strings (video) • Conditional statements [if/else] (OLI) • Pypi, pip, and packages (video) • Some useful packages (requests, arrow, etc) (video)	[5] OLI Module 2: Iteration [6] OLI Module 3: Making Decisions [7] Advanced String Manipulation (Links to an external site.) [8] Pip and External Packages (Links to an external site.)	Shaw (textbook), exercises 5, 6, 29-37 cli_input.py	• OLI Modules 2 & 3 • Lab 1 • Lab 2 • Homework 2

Week 3 [9/9/19 - 9/13/19]	• coding style • linting with pylint • advanced list usage • sets • tuples • dictionaries • reading data from files	[9] Python Coding Style and Pylint (Links to an external site.) [10] Advanced List Usage (Links to an external site.) [11] Sets and Tuples (Links to an external site.) [12] Dictionaries (Links to an external site.) [13] Reading Files (Links to an external site.)	Shaw (textbook), exercises 15, 20, 38-39 good_style.py bad_style.py reading_dicts.py	• Lab 3 • Lab 4 • Homework 3

Week 4 [9/16/19 - 9/20/19]	• writing data to files • writing your own modues • writing your own packages • the 'requests' library • querying a REST API • review of final projects	[14] Writing Files (Links to an external site.) [15] Creating Your Own Modules (Links to an external site.) [16] Creating Your Own Packages (Links to an external site.) [17] Querying a REST API (Links to an external site.) [18] Final Projects (Links to an external site.)	Shaw (textbook), exercises 16-17 writing_files.py writing_dicts.py stringops.py api_requests.py	• Lab 5 • Lab 6 • Homework 4 **Start Your Final Project**
Week 5 [9/23/19 - 9/27/19]	• object-oriented thinking • classes and objects • exceptions and error handling	[19] Object-Oriented Thinking (Links to an external site.) [20] Classes and Objects (Links to an external site.) [21] Exceptions and Error Handling (Links to an external site.)	Shaw (textbook), exercises 40-43	• Lab 7 • Lab 8 • Homework 5

Week 6 [9/30/19 - 10/4/19]	• class inheritance • duck typing • building web apps with Flask • network (client/ server) programming with python • tcp connections	[22] Class Inheritance and Duck Typing (Links to an external site.) [23] Simple Web Apps with Flask (Links to an external site.) [24] Exploring Network Programming (Links to an external site.)	Shaw (textbook), exercises 44, 50, 51 bird.py duck.py pigeon.py duck_typing_demo.py client.py server.py	• Lab 9 • Keep working on your final project!
Week 7 [10/7/19 - 10/11/19]	• natural language processing • machine learning • scikit-learn • moving on	[25] (Jupyter Notebook) Exploring Natural Language Processing [26] Exploring Machine Learning with scikit-learn (Links to an external site.) [27] Where do I go from here? (Links to an external site.)	Machine Learning 101 (slides) Exploring Machine Learning (Jupyter Notebook)	• Keep working on your final project!
Final Project Due **Wednesday, 10/16/19, 11:59PM ET**				**Final Project Due**