Syllabus
CMSC 280: Programming in a second language: Python

Fall 2019

Time:	MWF 2:00pm
Room:	Stevens 118
Website:	`http://cs.longwood.edu/courses/cmsc280`

A programming-intensive course that gives students a comprehensive introduction to a language not taught in the introductory sequence. Examples of languages which may be taught are Ada, Fortran, and Java. May be repeated for credit when language changes. 3 credits. Prerequisite: CMSC 160 or consent of the instructor.

Professor:	Don Blaheta
Office:	Ruffner 334
Phone:	x2191
Email:	`blahetadp@blahedo.org` or `blahetadp@longwood.edu`
Office hours:	Mondays 3–4pm; Tuesdays 2:30–3:30pm; Thursdays 11am–noon; Fridays 1–2pm

Overview

In other classes where you've gotten programming experience, the particular language (C++, mostly) was taught in service of some more primary conceptual understanding—of structured programming, of abstraction, or whatever. This course reverses that; here we primarily focus on learning a new language (in this case, Python), and although we may cover some concepts that are new to you, they will be in service of learning current, idiomatic Python, and building your ability to pick up additional new programming languages, as you'll have to do many times over the course of your career.

Textbook and resources

The required book for this class is Punch and Enbody, *The practice of computing using Python*, 3e (ISBN 978-0-13-437976-0).

The other main resource is provided by us: you'll be given an account on the department Linux machines (if you don't already have one), and you'll do your programming work there.

Course Student Learning Outcomes

At the end of this course, the successful student will be able to:

- write moderately complex Python programs, comprised of multiple components and making use of the Python libraries;
- use language documentation to discover new details about a language or library; and
- learn on their own how to program for particular libraries or technologies.

You may notice that the latter two objectives are explicit about placing responsibility on the student; while one goal of the course is for the students to learn Python, another major goal—in many ways the more important one—is to learn *how to learn* a new programming language.

Time

I figure that I have on average about 9 hours of your time every week, including class time as well as reading, practice, and projects. If you find you're regularly spending substantially more time than this, please do come discuss it with me, so that we can ensure you're making the most effective use of your time.

Projects

This course is centrally focused on the activity of learning how to program in a new language (and learning how to learn how to program in a new

language), so the fundamental measure of assessment will be based on writing programs. There are no exams in the course; the overwhelming bulk of your grade will be based on programming projects.

The details will differ for each project, but the top-level scoring breakdown for each project will be out of 100 points: 15 for an early checkpoint task, 20 for documentation on the final version, and the remaining 65 for quality/correctness of the final version.

The course is broadly divided into four main units, each of which corresponds to a programming project:

Unit 1: Basics. First you have to pick up the syntactic and stylistic differences between Python and C++ (and any other languages you know), so the first project will be fairly small and analogous to the sorts of programs you wrote in CMSC 160—just in a new language. The grade on this project counts for 10% of the final course grade.

Unit 2: Objects. You may have seen classes and objects before if you've taken 162, but Python takes a rather different approach to managing classes and objects than is typical in a C++ program (especially as taught in our 162 course). The second project will be a bit bigger and will give you practice with object-oriented design in the Python style, as well as with reading the documentation of the Python libraries to understand how to use them. This project will be worth 20% of the final course grade.

Unit 3: Graphics and GUIs. Designing and writing programs for a graphical user interface (GUI) is substantially different from writing command-line programs; principles of OO design provide a well-structured way to think about GUIs, and Python has a number of libraries to help you build them, so for this unit you'll learn how graphics and GUIs work in a Python context and write a program that makes use of them. This one will also be worth 20% of the course grade.

Unit 4: You figure it out! Having learned the basics of Python, how to read its documentation, and how to pick up a new technology within that framework, the rubber really hits the road on the final project, where you'll do all of that on a project of your own choosing. You'll work with a partner to write a program that does... something... and makes use of a Python library or technology that was not otherwise covered in class.

During our allocated final exam period, instead of an exam, you and your partner will give a demonstration of your project for the class.

The handed-in project is worth 30% of the grade; in addition, 10% of the grade is allocated to the demo presentation.

Other graded work

The remaining 10% of the grade will be allocated to general attendance and preparedness, to participation when appropriate, to quizzes if I give any, and to minor incidental assignments outside of the project work.

Grading scale

I tend to grade hard on individual assignments, but compensate for this in the final grades. The grading scale will be approximately as follows:

A−	[85, 90)	A	[90, 95)	A+	[95, 100]
B−	[70, 75)	B	[75, 80)	B+	[80, 85)
C−	[55, 60)	C	[60, 65)	C+	[65, 70)
D−	[40, 45)	D	[45, 50)	D+	[50, 55)

While there will be no "curve" in the statistical sense, I may slightly adjust the scale at the end of the term if it turns out some of the assignments were too difficult. Final grades of A+ are recorded as an A in the grading system. Final grades below the minimum for D− are recorded as an F.

Content

Wk	M	W	F
	August		
1	**26** — Introduction Basic Python syntax Ch 1–2, 4–5	**28** Ch. 6 Files Exceptions	**30** — Unit testing `pytest`, `unittest`
	September		
2	[Labor Day **no class**]	* **4** §§7.1–7.5 Lists	**6** §§7.6–7.9 Mutability, references Tuples
3	**9** §7.11 List comprehensions	**11** §8.1 Functions again Scope Comparing to C++	**13** §§8.2–8.5 Functions continued
4	**16** §§9.1–9.3 Dicts **Project 1 due**	**18** §§9.4–9.8 Sets Scope again	**20** §§11.1–11.4 Classes as structs
5	**23** §§11.5–11.7 Methods	**25** §§11.8–11.9 OOP principles public/private	**27** §§12.1–12.3 Operator overloading
		October	
6	**30** §§12.4–12.5 Case study: Rational	**2** §§12.6 Inheritance	**4** ** — Project 2 design and work day
7	**7** §§13.1–13.3 Case study: predator/prey	**9** — Learning by reading code	**11** §§13.4-13.5 Case study continued

* **3 September**: Deadline to add/drop classes (5pm)

** **4 October**: Deadline to elect pass/fail option (5pm)

Wk	M	W	F
	October		
8	[Fall Break **no class**]	**16** §§1.10, 2.3 Simple graphics: Turtle, plot **Project 2 due**	**18** TBA Events
9	**21** — GUI Graphics	**23** — GUI Graphics cont'd	**25** — GUI Graphics cont'd
			November
10	**28** §§14.1–14.2 Files again Processing CSV	**30** — Project 3 design and work day	**1** §§14.4–14.6 Exceptions again
11	**4** §16.1 Numbers	**6** * §§16.2.1–16.2.2 Function arguments Generators, `yield`	**8** §§16.2.3–16.2.4 map, filter, reduce lambda
12	**11** — Final project overview; Design and planning; **Project 3 due**	**13** §16.2.5 Decorators	**15** — Work day
13	**18** §14.3 `os`	**20** — Meetings with groups	**22** — Initial demos Resource sharing
14	**25** — Work day	[Thanksgiving **no class**]	[Thanksgiving **no class**]
	December		
15	**2** — Preliminary demo day	**4** — Work day Meetings with groups	**6** — Work day Meetings with groups

Demo day — Monday 9 Dec — 8–10:30am

* **6 November**: Deadline to withdraw from a class (5pm)

Policies

You can find several university-wide course policies at
`http://www.longwood.edu/academicaffairs/syllabus-statements/` .

Support

I'm in my office a lot (not just during posted office hours). Feel free to come in and ask questions (or just to talk). If you can't catch me in my office, email is probably your best bet.

Honor code policy

Above all, I ask and expect that you will conduct yourself with honesty and integrity—and not to ignore the other ten points of the Honor Code, either. Take pride in what you are capable of, and have the humility to give credit where it is due.

The two main forms of academic dishonesty are "cheating" and "plagiarism". "Cheating" is getting help from someplace you shouldn't, and "plagiarism" is presenting someone else's idea as if it's your own. If you ever find yourself inclined towards either of these, know that there are always other, better options. Persevere! See my website[1] for some discussion and examples of how to steer clear of these problems, and feel free to come talk to me if you need help finding some of those other options (even if it's for another course).

Cheating or plagiarism (on any assignment) will normally receive a *minimum* penalty of a lowered *course* grade, ranging up to an F in the course. Cases will also be turned in to the Honor Board. But: I believe in your potential, and I hope that you will, or will grow to, observe this policy not simply to evade punishment but positively as a matter of character.

Accommodations

If you have any special need that I can accommodate, I'm happy to do so; come speak to me early in the term so we can set things up. If you

[1] `http://cs.longwood.edu/~dblaheta/collab.html`

have a documented disability, you should also contact Longwood's Office of Disability Resources (Brock Hall, x2391) to discuss some of the support the college can offer you. All such conversations are confidential.

Attendance and late policy

Attendance is required, and assignments must be turned in on time. That said, if you have a good reason to miss class or hand something in late, I tend to be fairly liberal with extensions if you ask in advance. (Good reasons do include assignments due for other classes.) (And medical and family emergencies are exempted from the "in advance" part, of course. But contact me ASAP.)

Frequent absence will result in a lowered participation grade; habitual absence may in extreme cases result in a failing grade for the class. *Unexcused* late assignments will normally be given a zero.

Inclement weather policy

I don't plan to cancel class for weather unless the entire college shuts down. If you are commuting or are otherwise significantly affected by a weather event, use your own best judgement; and if you do miss class for this reason, contact me as soon as possible to make up missed work.

Early bird policy

Nobody's perfect, and on occasion an assignment gets written a little unclearly (or, once in a while, with an actual error in it). If you catch one and bring it to my attention early, so that I can issue a clarification or correction to the rest of the class, there'll be some extra credit in it for you.