COMP 360 — Programming Languages — Spring 2023

CRN 23321

Instructor: Phillip Kirlin
Meetings: Tue/Thu 2–3:15pm
Course website: `http://pkirlin.github.io/pl-s23`
Email: `kirlinp@rhodes.edu` (please include "PL" or "Programming Languages" somewhere in the subject)

Course Overview: In this course, we will discuss the fundamental concepts that appear in one form or another in almost every programming language. We will also get some sense of how these concepts "fit together" to provide what programmers need in a language. We will examine different languages to see how they can take complementary approaches to representing these concepts. All of this is intended to make you a better software developer, in any language.

Successful course participants will obtain an accurate understanding of what functional and object-oriented programs mean, develop the skills necessary to learn new programming languages quickly, master specific language concepts such that they can recognize them in strange guises, learn to evaluate the power and elegance of programming languages and their constructs, attain reasonable proficiency in a number of popular programming languages, and, as a byproduct, become more proficient in languages they already know.

Text: There is no required textbook. You may find it useful to consult Abelman and Sussman, *Structure and Interpretation of Computer Programs*, which is freely available online, as well as the documentation for the various programming languages we will use.

Prerequisites: COMP 241 (Data Structures and Algorithms). Knowledge of programming in Python and Java is assumed. Students should have enough confidence in their programming abilities to be able to debug programs effectively, locate and understand documentation for specific language features, and generally program independently.

Course Work:

	Tentative weight	Tentative date
Homework/Projects	55%	
Midterm	20%	Tuesday, February 28, in class
Comprehensive final exam	25%	Saturday, December 9, 5:30pm

Final letter grades of A–, B–, C–, and D– are guaranteed with final course grades of 90%, 80%, 70%, and 60%, respectively. If your final course grade falls near a letter grade boundary, I may take into account class participation, attendance, and/or improvement during the semester.

Any work submitted on paper should be typed or handwritten neatly. Poorly written work will not be graded. All pages of assignments should be stapled together.

Course Topics: (not necessarily in this order)

- Functional programming paradigms
- Static and dynamic typing
- Tail recursion

- First-class functions, closures
- Lexical and dynamic scoping
- Delayed evaluation and thunks
- Lazy evaluation and streams
- Threading
- Interpreter construction

Late Work and Makeup Assignments: In general, late work will not be accepted without arranging an extension in advance with the instructor, and will often come with a late penalty. Please make every effort to submit assignments on time.

If you have a valid reason for a makeup exam, inform your instructor as soon as you know. A valid reason is a medical emergency, a death in the family, religious observation, a college-sponsored off-campus activity, and, quite frankly, very little else. Generally, assignment extensions will only be granted for *unplanned* circumstances (e.g., the first two reasons above).

Office Hours: In addition to regular office hours, am also available immediately after class for short questions. You never need an appointment to see me during regular office hours; you can just come by. Outside of regular office hours, feel free to stop by my office, and if I have time, I'll try to help you. If I don't have time at that moment, we'll set up an appointment for a different time. Don't be shy about coming by my office or sending me email if you can't make my regular office hours. I always set aside time each week for "unscheduled" office hours.

Attendance: Attendance is expected for each class. If your attendance deteriorates, you will be referred to the dean and asked to drop the course. Attendance, participation, and apparent overall improvement trend may be considered in assigning a final grade. Attendance will be checked each class lecture period. After five unexcused absences, each additional absence will reduce the final grade for the course by one letter grade.

Class Conduct:

- I encourage everyone to participate in class. Raise your hand if you have a question or comment. Please don't be shy about this; if you are confused about something, it is likely that someone else is confused as well. Teaching and learning is a partnership between the instructor and the students, and asking questions not only helps you understand the material, it also helps me know what I'm doing right or wrong.
- Do not use your cell phone while in class, and keep the ringer on silent.
- If you cannot make it to class for whatever reason, make sure that you know what happened during the lecture that you missed. It is your responsibility, and nobody else's, to do so. The best way to do this is to ask a classmate.
- If you have to leave a class early, please inform the instructor in advance.

Collaboration: Students should talk to each other about the subject matter of this class and help each other. It is fine to discuss the readings, lectures, and problems and ask questions about them. I encourage such questions in class as well as elsewhere. However, there is a line past which you must not go, e.g., copying a solution from a fellow student, book, website, etc. will cause you to fail the course, or worse. If a significant part of one of your solutions is due to

someone else, or something you've read, then you must acknowledge your source. Failure to do so is a serious academic violation. ***This includes the use of artificial intelligence tools or other software.*** Of course, even after you acknowledge your source you must still understand your solution and write it in your own words. Copying a solution from the web, a book, or classmate will result in failure even if you acknowledge your source, unless you put it in quotation marks and say something like, "Here is Amy's solution, but I don't understand it enough to absorb it and write it in my own words." However, this won't get you much — if any — credit.

Rules for Completing Assignments Independently

- Unless otherwise specified, programming assignments handed in for this course are to be done *independently*.
- Talking to people (faculty, other students in the course, others with programming experience) is one of the best ways to learn. I am always willing to answer your questions or provide hints if you are stuck. But when you ask other people for help, sometimes it is difficult to know what constitutes legitimate assistance and what does not. In general, follow these rules:
  - **Rule 1: Do not look at anyone else's code for the same project, or a different project that solves a similar or identical problem.**
    Details: "Anyone else" here refers to other members of the class, people who have taken the class before, people at other schools enrolled in similar classes, or any code you find online or in print. "Similar or identical problem" here should allow you to look at code that uses techniques applied in different situations that you can then adapt to your project. However, if you find yourself copying-and-pasting code or directly transforming code line by line to fit into your program, then that is considered plagiarism.
    Exception: You may help someone else debug their program, or seek assistance in debugging yours. However, this requires the person writing the code being debugged to have made a good-faith attempt to write the program in the first place, and the goal of the debugging must be to fix one specific problem with the code, not re-write something from scratch.
  - **Rule 2: Do not write code or pseudocode with anyone else.**
    Details: You must make a good faith effort to develop and implement your ideas independently before seeking assistance. Feel free to discuss the project *in general* with anyone else before you begin and as you're developing your program, but when you get to the level of writing code or pseudocode, you should be working independently.

  The underlying idea is that you are entitled to seek assistance in ways which will genuinely help you to learn the material (as opposed to just getting the assignment done). Programming assignments are graded as a benefit to you; they are your chance to show what you have learned under circumstances less stressful than an exam. In return, I ask only that your work fairly reflect your understanding and your effort in the course.

Additional Information: To streamline this syllabus, I have moved a number of policies common to all my classes to a separate document called "Additional Course Policies." Those policies

should be interpreted as a part of this syllabus.