Syllabus, CSCI312, Programming Languages

Winter 2023

Instructor: Dr. Geoffrey Matthews, Parmly 407A, gmatthews <at> wlu <dot> edu

Web page: https://github.com/geofmatthews/csci312

Office hours: MWF 1:30-2:30

Lectures: MWF 12:15 - 1:15, Parmly 405

Goals. This class is an introduction to the semantics of programming languages. We will study how things work: functions, scoping, objects, continuations, and how these and other features determine the character and style of various programming languages.

Rather than study semantics abstractly, we will study these features using **interpreter semantics**: to understand or explain how a feature works, build an interpreter for it. This method gives immediate feedback, allowing us to program using this feature in simple languages, and evaluate its usefulness for various tasks.

Although we will do some simple programming in a variety of programming languages, most of our programming will be in Scheme, http://www.schemers.org/, an advanced functional programming language. Functional programming is a methodology, style, and suite of programming language features that facilitates rapid programming and algorithmic design. Some required reading on the usefulness of functional programming: *Beating the Averages*, http://www.paulgraham.com/paulgraham/avg.html. Although we will program in a single language, we will build interpreters for a wide variety of languages. Each language you encounter in your career will partake of some of these features; understanding how they are implemented, and what their runtime support entails, will give you a deep understanding of any programming language you come across.

Texts:

- *Programming Languages, Application and Interpretation*, Shriram Krishnamurthi,
  https://cs.brown.edu/~sk/Publications/Books/ProgLangs/2007-04-26/plai-2007-04-26.pdf
  This will be our textbook. We will use the **first** edition.

  Note that the text is pedagogical; frequently he will start with the obvious (but wrong) way to solve a problem, and then work his way to the right solution 5 or 10 pages later. If you don't spend the time reading, and rereading, you may get confused. However, if you put in the time, you will understand at a much deeper level.

- Scheme introductions:
  - *Teach Yourself Scheme in Fixnum Days*,
    https://ds26gte.github.io/tyscheme/
    Best introductory Scheme tutorial I've found. There are many more online.
    Note: skip Chapter 1, it deals with a different Scheme system. Instead, run through the **Quick** tutorial found here: https://docs.racket-lang.org/ or in the Racket help desk available from DrRacket.
  - *The Scheme Programming Language*, (4/e),
    http://www.scheme.com/tspl4/
    Good reference manual for Scheme.
  - *How to Design Programs*, (2/e),
    https://htdp.org/
    Excellent introduction to programming in general, and the functional style or programming in Scheme.

Software:

- *The Racket programming language*, https://racket-lang.org/

Grading: Homework: 50%; Midterm: 20%; Final: 30%; Extra Credit: 10%.

Letter grades: A $\geq$ 90% > B $\geq$ 80% > C $\geq$ 70% > D $\geq$ 60% > F

Homework: Homework assignments will be awarded as they come up, to match timing with the lectures. They will be announced in class and on canvas.

Homework assignments will be available on the github site. Several are there already, so you can look at them in advance, but I reserve the right to revise them at any time prior to their announcement in class and the setting of a due date.

Points will be awarded according to the following.

Factors to consider:	**Points**
Outstanding work. Well formatted, modular, well commented, well designed. Clear, self-documenting identifiers: variables, functions, class names. Good, consistent docstrings. Extra work on optional problems or extensions of the required work. Error checking. Comprehensive unit tests. Innovative solutions. Extensive documentation on design decisions and results.	5
Good work. The problem is solved completely and without errors. Adequate documentation.	4
Adequate work. Most, but not all of the problem is solved. Poor documentation.	3
Incomplete work. Some progress was made, but no complete solution. Nonexistent documentation.	2
Poor work. Little or none of the problem is solved. Random bits of code copied from lectures or the problem description without showing any real coherence or understanding of an approach to the problem.	1
Unacceptable work. Syntax errors. Not turned in on time. Did not follow instructions.	0

Midterms: There will be two midterms, each worth 10%, as in the class schedule below. They is open book and open notes, but you may not consult with any classmates or the internet or other resources.

Final exam: The final exam is comprehensive. It is open book and open notes, but you may not consult with any classmates or the internet or other resources.

Extra credit: A maximum of 10 percentage points of extra credit may be given for a special project. Special projects must: (a) be proposed to the instructor, in writing, three weeks before the last lecture, (b) be approved by the instructor at that time, (c) must include a substantive software component, written in Scheme, (d) must include a writeup with an introduction outlining the purpose of the project, a section discussing the results, and a conclusion, (e) must include well annotated source code, a user's guide, and a programmer's guide to the software, and (f) must be complete and turned in before the last lecture of class.

Schedule:

Su	Mo	Tu	We	Th	Fr	Sa	
							January 2023
8	9	10	11	12	13	14	Python AST, Scheme intro
15	16	17	18	19	20	21	PLAI 1,2,3
22	23	24	25	26	27	28	PLAI 4,5,6
29	30	31					Review
							February 2023
			1	2	3	4	Review; Midterm 1
5	6	7	8	9	10	11	PLAI 7,8,9
12	13	14	15	16	17	18	PLAI 10,11,12
19	20	21	22	23	24	25	holiday
26	27	28					PLAI 12
							March 2023
			1	2	3	4	PLAI 13,14
5	6	7	8	9	10	11	Review; Midterm 2
12	13	14	15	16	17	18	PLAI 15,16,17
19	20	21	22	23	24	25	PLAI 18,19,20
26	27	28	29	30	31		PLAI 24,25,26
							April 2023
						1	
2	3	4	5	6	7	8	Review
9	10	11	12	13	14	15	Final