# Principles of Programming Languages

CS496

# Teachers


Instructor: Eduardo Bonelli

E-mail: ebonelli@stevens.edu


Office hours: MW 5PM-6:30PM

CAs: Eric Altenburg, John Brummer, John Cinquegrana, Nick Guo, Hamzah Nizami, Jared Pincus, Nikolas Stefanov, Nicholas Szegheo, Mathew Seedhom

# About this Course

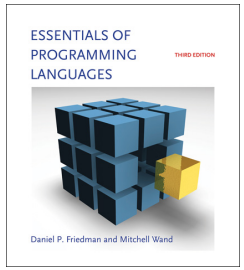
- Programming Languages as objects of study
- We study de principles on which PL are erected
- We do so by implementing these concepts in our own PLs
- This is a hands-on course

# Ask questions!

- Feel free to interrupt and ask questions at any time
  - Your questions also help me better understand the topics
  - It also helps fellow students who might have similar doubts
- Contact me by email
- Come see me during office hours
- See the CAs during their office hours

# Bibliography

- Course notes above all
- Relevant book




- However, we do not use Scheme
- Relevant additional texts
  - Structure and Interpretation of Computer Programs (H. Abelson and G. J. Sussman with J.Sussman, MIT Press, 1984)
  - Types and Programming Languages (B. Pierce, MIT Press, 2002)

# Important Information in the Syllabus – Homework

- Policy for late submissions: 2 points off for every hour past the deadline.
- 0 if code does not compile or submission is empty

# Quizzes

- Typically on Wednesdays
- 0 if absent
- Solved in class immediately after handing it in
- In groups of at most two
- You cannot work with students from other sections
- Lowest quiz dropped at end of semester

# Exams

- Two
  - Midterm
  - Endterm
- Midterm and endterm exam dates are listed in the tentative course schedule available in Canvas.
- Format: considering pencil and paper or short (couple of days) assignment

# Weight of Grading Categories

| | |
|---|---|
| Homework | (35%) |
| Quizzes | (25%) |
| Midterm | (20%) |
| Endterm | (20%) |

# Lectures

- Most of the lectures will involve myself doing coding
- Please make sure you are present

# The Interpreter Approach

- Fundamental concepts in PL studied by
  - Defining a representative language
  - Defining the concepts required to execute a program in this language
  - Defining an interpreter that executes such programs
- Interpreters allow for a deep understanding of PL concepts

# Some Concepts we shall Study

- Foundations of expressions: inductive sets, recursion, induction
- Functional Programming: expression, value, closure, environment, substitution, type checking, type inference
- Imperative Programming: Command, effect, mutable variable, state
- Object Oriented Programming: class, object, class hierarchy, inheritence, dynamic method dispatch, static method dispatch, super, self

# Our Host Language

- Hands-on approach to these concepts
  - We'll write interpreters for simple PLs that build on them
- We'll use a functional language for writing interpeters
  - They are declarative
  - Provide a high-level of abstraction
    - Programs considered "executable specifications"
- Examples:
  - OCaml (we'll use this one)
  - Haskell
  - ML
  - Erlang
  - Scheme
  - F#, etc.

# OCaml

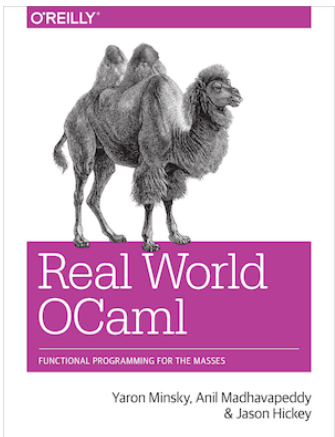
- Industrial-strength, statically-typed functional programming language
- Lightweight, approachable setting for learning about program design

Who else uses OCaml?[1]


[1] Source: `www.seas.upenn.edu/~cis120`

# Bibliography

- Great for beginners to FP: `https://www.cs.cornell.edu/courses/cs3110/2019sp/textbook/`
- Great tutorial notes for beginners to FP: Introduction to Objective Caml, a set of notes by Jason Hickey `courses.cms.caltech.edu/cs134/cs134b/book.pdf`
- A great reference to continue learning (`https://realworldocaml.org`)



# Installing OCaml

- Document will be provided through Canvas