

COSE212: Programming Languages

Lecture 0 — Course Overview

Hakjoo Oh
2018 Fall

Basic Information

Instructor: Hakjoo Oh

- **Position:** Associate professor in CS, Korea University
- **Expertise:** Software Analysis, Programming Languages
- **Office:** 616c, Science Library
- **Email:** hakjoo_oh@korea.ac.kr
- **Office Hours:** 1:00pm–3:00pm Mondays and Wednesdays (by appointment)

TAs:

- Junho Lee (wnsgh1906@korea.ac.kr)
- Downon Song (sdw0316@gmail.com)

Course Website:

- <http://prl.korea.ac.kr/~pronto/home/courses/cose212/2018/>
- Course materials will be available here.

About This Course

This course is *not* about

- to learn particular programming languages



- to improve your “programming skills” (e.g., tools, libraries, etc)

Instead, in this course you will learn

- fundamental principles of modern programming languages
- how programming systems are designed and implemented
- thinking formally and rigorously

To succeed in this course, you must

- have basic programming skills
- be familiar with at least two PLs (e.g., C, Java)
- have taken Theory of Computation, Discrete Math, etc
- be prepared to learn new things

Design and Implementation of Programming Languages

We will learn programming language concepts by designing and implementing our own programming language system.

- We will define a programming language. For example, “factorial” is written in our language as follows:

```
let x = read in
letrec fact(n) =
  if iszero n then 1
  else ((fact (n-1)) * n)
in (fact x)
```

- We will design and implement an interpreter for the language:

Program \rightarrow Interpreter \rightarrow Result

- We will design and implement a type checker for the language:

Program \rightarrow Type Checker \rightarrow Safe/Unsafe

Topics

- **Part 1 (Preliminaries):** inductive definition, basics of functional programming, recursive and higher-order programming
- **Part 2 (Basic concepts):** syntax, semantics, naming, binding, scoping, environment, interpreters, states, side-effects, store, reference, mutable variables, parameter passing
- **Part 3 (Advanced concepts):** type system, typing rules, type checking, soundness/completeness, automatic type inference, polymorphic type system, lambda calculus, program synthesis

Course Materials

- Essentials of Programming Languages (Third Edition) by Daniel P. Friedman and Mitchell Wand. MIT Press.



(Not required but recommended)

- Self-contained slides will be provided.

Grading

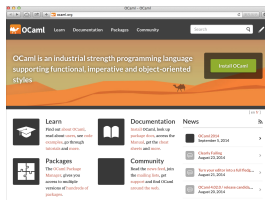
- Homework – 50%
 - ▶ 5–6 programming assignments
 - ▶ No late submissions will be accepted.
- Final exam – 45%
- Attendance – 5%

Assignment Policy / Academic Integrity

- **All assignments must be your own work.**
- Discussion with fellow students is encouraged and you can discuss how to approach the problem. However, your code must be your own.
 - ▶ Discussion must be limited to general discussion and must not involve details of how to write code.
 - ▶ You must write your code by yourself and must not look at someone else's code (including ones on the web).
 - ▶ Do not allow other students to copy your code.
 - ▶ Do not post your code on the public web.
- Cheating (violating above rules) gets you 0 for the *entire* HW score.
 - ▶ We use automatic technology for detecting clones

Programming in ML

- ML is a general-purpose programming language, reflecting the core research achievements in the field of programming languages.
 - ▶ higher-order functions
 - ▶ static typing and automatic type inference
 - ▶ parametric polymorphism
 - ▶ algebraic data types and pattern matching
 - ▶ automatic garbage collection
- ML inspired the design of modern programming languages.
 - ▶ C#, F#, Scala, Java, JavaScript, Haskell, Rust, etc
- We use OCaml, a French dialect of ML:



<http://ocaml.org>

Web-based Programming Environment

We will provide a web-based programming environment, where you do and submit homework assignments.

The screenshot shows a web-based programming environment interface. The top navigation bar includes a hamburger menu, the text "Home", the page title "COSE212 - Programming Language", and links for "Sign Up" and "Login". On the left side, there is a sidebar menu with sections: "Assignment Policy", "Homework Select", "Feedback", and "Option". The "Option" section contains three dropdown menus: "Font Size" (set to "18px"), "Theme" (set to "White"), and "Mode" (set to "REPL"). The main area is a code editor titled "README.ml" containing the following OCaml code:

```
1 let rec inc_all l =
2   match l with
3   | [] -> []
4   | hd::tl -> (hd+1)::(inc_all tl)
5
6 let rec square_all l =
7   match l with
8   | [] -> []
9   | hd::tl -> (hd*hd)::(square_all tl)
10
11 let rec map f l =
12   match l with
13   | [] -> []
14   | hd::tl -> (f hd)::(map f tl)
```

Below the code editor is a "Compile Result Area". At the bottom of the interface, there are two buttons: "Run" (green) and "Save" (blue).

Web-based Programming Environment

The system has a feature of generating feedback automatically.

- Providing personalized feedback on your homework assignments is challenging because of the huge number of students.
- So we have developed an AI system that automatically provides feedback on logical errors in your submissions.

```
          a != b
          ↑
let rec sigma f a b =
  if f a != f b then
    let induction = f b in
      induction + sigma f a (b-1)
  else f b
```

Questions?