COSE212: Programming Languages

Lecture 0 — Course Overview

Hakjoo Oh
2019 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 (by appointment)

TAs:

- Dowon Song (`dowon_song@korea.ac.kr`)
- Sunho Lee (`seonho_lee@korea.ac.kr`)
- Jisuk Byun (`grayplc@korea.ac.kr`)

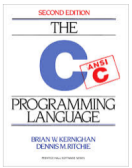
Course Website:

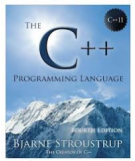
- `http://prl.korea.ac.kr/~pronto/home/courses/cose212/2019/`
- Course materials will be available here.

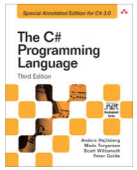
About This Course

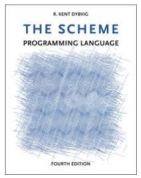
This is not an introductory course on programming.

- You will not learn particular programming languages










- You will not learn how to write programs in those languages

Instead, in this course you will learn

- how programming languages are designed and implemented
- fundamental principles of modern programming languages
- 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

You 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:

$$\text{Program} \longrightarrow \boxed{\text{Interpreter}} \longrightarrow \text{Result}$$

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

$$\text{Program} \longrightarrow \boxed{\text{Type Checker}} \longrightarrow \text{Safe/Unsafe}$$

Functional Programming

The secondary goal of this course is to be familiarized with functional programming, which encourages using pure functions rather than making side effects.

- Functional programming is one of the major programming paradigms adopted in modern programming languages such as Python, JavaScript, C++, Java8, Scala, Go, etc.
- In this course, you will learn functional programming with OCaml[1] and use it to implement programming languages.

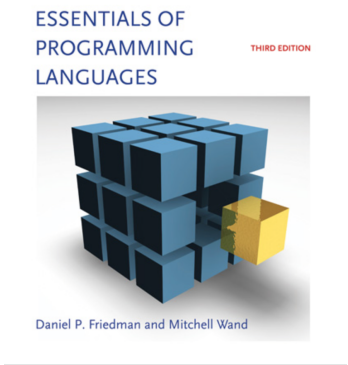
[1] `https://ocaml.org`

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

- Self-contained slides will be provided.
  - You are required to attend every class (otherwise, it'd be difficult to catch up)
- (Supplementary) Essentials of Programming Languages (Third Edition) by Daniel P. Friedman and Mitchell Wand. MIT Press.



Grading

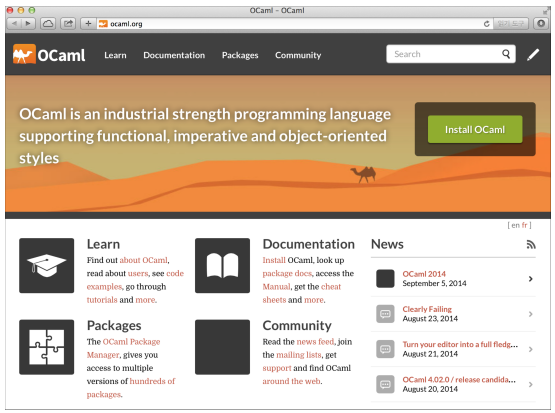
- Homework – 60%
  - 5–7 programming assignments
  - No late submissions will be accepted.
- Final exam – 35%
- Attendance – 5%

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

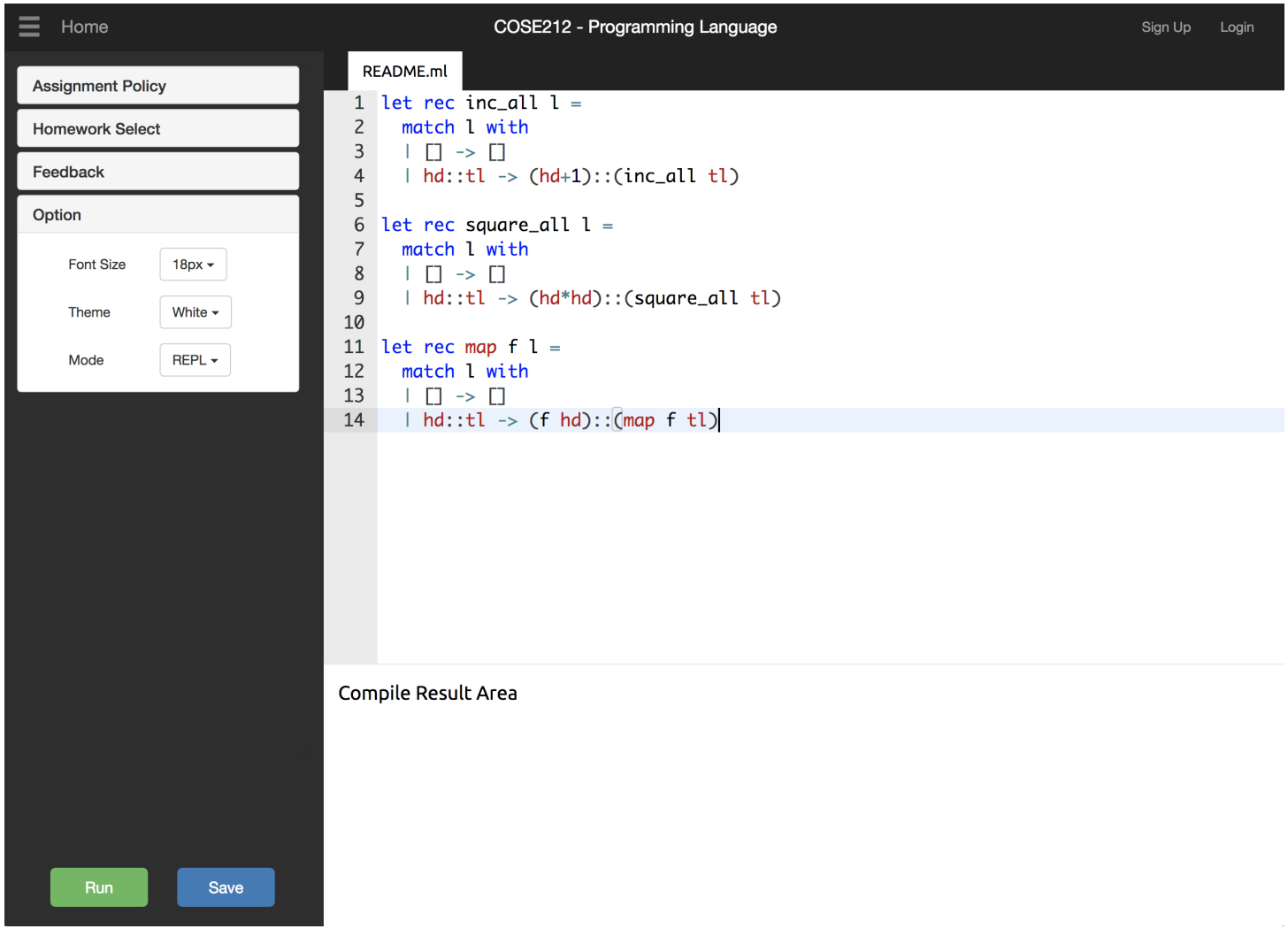
- ML is a general-purpose programming language, reflecting the core research achievements in the field of programming languages.
  - higher-order functions
  - static typing and automatic type inference
  - parametric polymorphism
  - algebraic data types and pattern matching
  - automatic garbage collection
- 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.



Questions?