Lecture 0 — Course Overview

COSE212: Programming Languages

Minseok Jeon

2026 Spring

Basic Information

Instructor: Minseok Jeon

- **Position:** Assistant professor in EECS, DGIST
- **Expertise:** Programming Languages
- **Office:** 211 E7 Bldg.
- **Email:** `minseok_jeon@dgist.ac.kr`
- **Office Hours:** 9:00am–11:59am Mondays (by appointment)

TAs:

- Sanggyu Oh (sg549@dgist.ac.kr)

Course slides will be uploaded to:

- LMS
- https://dgistpl.github.io/courses/cse307/2026/

About This Course

This is not an introductory course on programming.

- You will not learn particular programming languages

THE C PROGRAMMING LANGUAGE
THE C++ PROGRAMMING LANGUAGE
Java
The C# Programming Language
python
JavaScript
THE SCHEME PROGRAMMING LANGUAGE
php

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

About This Course

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:

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

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

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

Desirable Achievement

By the end of this course, I hope you are able to:

- **Develop a programming language to address your own problems**
  - Design a language to solve a specific (i.e., your own) problem
- Programming languages are not just tools to use – they are tools you can **create** to solve problems more effectively

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.

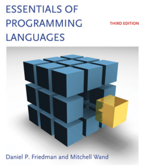
[1] `https://ocaml.org`

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)
- Hakjoo Oh. Introduction to Principles of Programming Languages. (pdf will be provided)
- (Supplementary) Essentials of Programming Languages (Third Edition) by Daniel P. Friedman and Mitchell Wand. MIT Press.

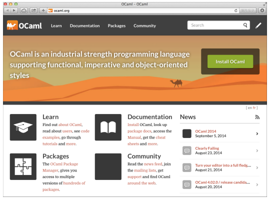

ESSENTIALS OF PROGRAMMING LANGUAGES
THIRD EDITION
Daniel P. Friedman and Mitchell Wand

Grading

- Attendance & Participation – 10%
- Homework assignments – 10%
- Midterm – 40%
- Final – 40%

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:



OCaml is an industrial strength programming language supporting functional, imperative and object-oriented styles

Learn
Find out about OCaml, read about users, see code examples, go through tutorials and more.

Documentation
Install OCaml, look up package docs, access the Manual, get the cheat sheets and more.

Packages
The OCaml Package Manager, gives you access to multiple versions of hundreds of packages.

Community
Read the news feed, join the mailing lists, get support and find OCaml around the web.

News
OCaml 2014
September 5, 2014
Clearly Failing
August 23, 2014
Turn your editor into a full fledg...
August 21, 2014
OCaml 4.02.0 / release candida...
August 20, 2014

`http://ocaml.org`

Questions?

(!) Special Project

Project Goal: Design a programming language that is AI-resistant but human-friendly.

Challenge:

- Existing AI systems (ChatGPT, Claude, etc.) can easily solve programming assignments in conventional languages
- This undermines the learning process and academic integrity

Your Task:

- Design a new programming language where:
  - AI models struggle to write correct solutions
  - Human programmers can still solve assignments effectively
- Consider: syntax design, semantic features, unconventional paradigms
- Deliverable: Language specification + justification of design choices

This is an open-ended research project – be creative!

(!) Special Project

Reward:

- Students who successfully design such a language will receive **a higher score** on the final grade

How to Participate:

- Email the instructor (`minseok_jeon@dgist.ac.kr`) if you want to challenge this project

Good luck!