

Lecture 0 – Course Overview

COSE212: Programming Languages

Jihyeok Park



2025 Fall

- **Instructor:** Jihyeok Park (박지혁)
 - **Position:** Assistant Professor in CS, Korea University
 - **Expertise:** Programming Languages, Software Analysis
 - **Office hours:** 14:00–16:00, Tuesdays (appointment by e-mail)
 - **Office:** 507, Jung Woonoh IT & General Education Center
 - **Email:** jihyeok_park@korea.ac.kr
- **Class:** COSE212 - 01 (English)
- **Homepage:** <https://plrg.korea.ac.kr/courses/cose212/>
- **LMS:** <https://lms.korea.ac.kr/>
 - Please use the **Board** > **Q&A** section for questions.
- **Teaching Assistant:** cose212@googlegroups.com
 - Seongmin Ko (고성민)
 - Hyunjoon Kim (김현준)
 - Minseok Choe (최민석)

- **4 Homework Assignments: 30%**

- Programming assignments in Scala (submission in [LMS](#))
- Please check the course website for the detailed [policy on academic integrity](#). Some highlights are:
 - Do not share your code with others / Do not see others' code.
 - It's your code only if you can explain all the details of the code.
- If violated, you get a **zero** for the assignment or an **F** for the course.

- **Midterm exam: 30%**

- October 22 (Wed.) 18:30 – 21:00 (150 min.)
- In classroom, closed book, closed notes

- **Final exam: 30%**

- December 17 (Wed.) 18:30 – 21:00 (150 min.)
- In classroom, closed book, closed notes

- **Attendance: 10%** (From second week)

- Please use [LMS](#) to attend the class **by yourself**.

Weak	Contents	Weak	Contents
1	Introduction	9	Continuations
2	Syntax and Semantics	10	First-Class Continuations
3	Identifiers and First-Order Functions	11	Type Systems
4	First-Class Functions and Recursion	12	Algebraic Data Types
5	Mutable Variables	13	Parametric Polymorphism
6	Garbage Collection	14	Subtype Polymorphism
7	Lazy Evaluation	15	Type Inference
8	Midterm Exam (Oct. 22 - Wed.)	16	Final Exam (Dec. 17 - Wed.)

On the four days listed below, there will be no offline lectures.

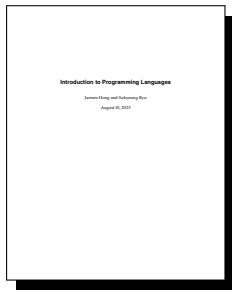
- Oct. 6 (Mon.) / 8 (Wed.) – 추석
- Nov. 17 (Mon.) / 19 (Wed.) – International Conference

Instead, lecture videos will be uploaded to [LMS](#).

- **Self-contained lecture notes.**

<https://plrg.korea.ac.kr/courses/cose212/>

- **Reference: “Introduction to Programming Languages”** written by Jaemin Hong and Sukyoung Ryu



<https://hjaem.info/itpl>

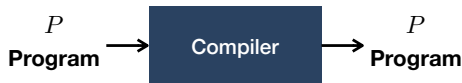
To learn **essential concepts** of **programming languages**

- Why? After this course, you will be able to:
 - **learn new programming languages** quickly.
 - **evaluate** and pick the best language for a given task.
 - **design** your own **specialized languages** for specific tasks.
- How? You will learn how to:
 - **design** programming languages in a **mathematical** way.
 - **implement** their **interpreters** using **Scala**.
- However, note that:
 - You will **NOT learn** particular programming languages.
 - You will **NOT learn** how to write programs in those languages.
 - This is **NOT** an introductory course. You should have a **strong understanding** of introductory computer science courses. (i.e., theory of computation, discrete mathematics, and data structures)

- An **interpreter** takes and executes a program to produce the result.



- Good for **understanding** program behavior, easy to **implement**.
 - For example, scala, python, bash, desktop calculator, etc.
 - You will implement interpreters of various languages in this course.
- A **compiler** takes a program and produces another program.



- Good for **speed**, but more **complex**.
- For example, scalac, gcc, javac, etc.
- If you're interested in compilers, take **COSE312: Compilers**.

We will grow a language step by step from a simple arithmetic language to a complex language with various features.

- **Part 1: Untyped Languages**

- Syntax, Semantics, Identifiers
- **Functional** – Functions, Closures, Recursion
- **Imperative** – Mutation, Sequences, Garbage Collection
- **Advanced** – Lazy Evaluation, Continuations

- **Part 2: Typed Languages**

- **Type Systems** – Types, Typing Rules, Typed Languages
- **Algebraic Data Types** – Variants, Pattern Matching
- **Polymorphism** – Parametric Polymorphism, Subtype Polymorphism
- **Type Inference** – Type Variables, Type Unification

- Basic Introduction of Scala

Jihyeok Park

jihyeok_park@korea.ac.kr

<https://plrg.korea.ac.kr>