

Lecture 0 – Course Overview

COSE212: Programming Languages

Jihyeok Park



2023 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:** 609A, Science Library Bldg
 - **Email:** jihyeok_park@korea.ac.kr

- **Class:** COSE212 - 02 (English)

- **Lectures** 13:30–14:45, Mon. & Wed. @ 535 아산이학관

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

- This is **NOT** an introductory course. You should have a **strong understanding** of introductory computer science courses. (i.e., OOP, theory of computation, discrete mathematics, and data structures)

Week	Contents
1	Introduction
2	Syntax and Semantics
3	Identifiers and First-Order Functions
4	First-Class Functions and Recursion
5	Mutable Variables
6	Garbage Collection
7	Lazy Evaluation
8	Midterm Exam (Oct. 25 - Wed.)
9	Continuations
10	First-Class Continuations
11	Type Systems
12	Algebraic Data Types
13	Parametric Polymorphism
14	Subtype Polymorphism
15	Type Inference
16	Final Exam (Dec. 20 - Wed.)

- **2–4 Homework Assignments: 20%**
 - Programming assignments (submission in **Blackboard**)
 - You can utilize or refer to any other materials (e.g., ChatGPT), but you **MUST** write your **OWN** solution.
 - **Cheating is strictly prohibited. Cheating will get you an F.**

- **Midterm exam: 30%**
 - October 25 (Wed.) 13:30 – 14:45 (in class, 75 min.)

- **Final exam: 40%**
 - December 20 (Wed.) 13:30 – 14:45 (in class, 75 min.)

- **Attendance: 10%**
 - Please use **Blackboard** to attend the class **by yourself**.

- **Self-contained lecture notes.**

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

(Special thanks to Prof. **Sukyoung Ryu** @ KAIST)

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



<https://hjaem.info/itpl>

Learn **Essential Concepts** of **Programming Languages**

- Why?
 - To **learn new programming languages** quickly.
 - To **evaluate** and pick the best language for a given task.
 - To design a **specialized language** for a specific task.
- How?

By Implementing **Interpreters** using **Scala**

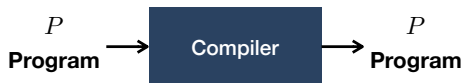
- You will learn how to define **syntax** and **semantics** of target languages.
- You will implement **interpreters** of the target languages.
- You will use **Scala** as an implementation language.

- 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>