Lecture 0 – Course Overview

COSE212: Programming Languages

Jihyeok Park

2023 Fall

Course Information

PLRG

- **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) - **Only for CS students**
- **Lectures** 13:30–14:45, Mon. & Wed. @ 604 우정정보관
- **Homepage:** https://plrg.korea.ac.kr/courses/cose212/
- Please use **Blackboard** when asking questions, checking the attendance, uploading your homework, etc.

Schedule

PLRG

Weak	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-Order Representation of Continuations
11	Type Systems
12	Type Inference
13	Algebraic Data Types
14	Parametric Polymorphism
15	Subtype Polymorphism
16	**Final Exam (Dec. 20 - Wed.)**

Grading

PLRG

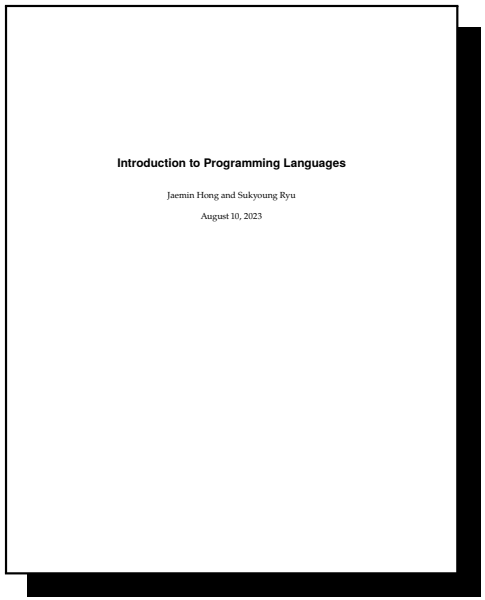
- **2–4 Homework Assignments: 20%**
  - Programming assignments (submission in blackboard)
  - You can utilize or refer to any other materials (e.g., ChatGPT), but you **MUST** write your **OWN** solution.
  - Cheating is strictly prohibited. Cheating will get you an F.
- **Midterm exam: 30%**
  - October 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 and Participation: 10%**
  - Please use **blackboard** to attend the class.

Course Materials

- **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

Goal of This Course

PLRG

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**

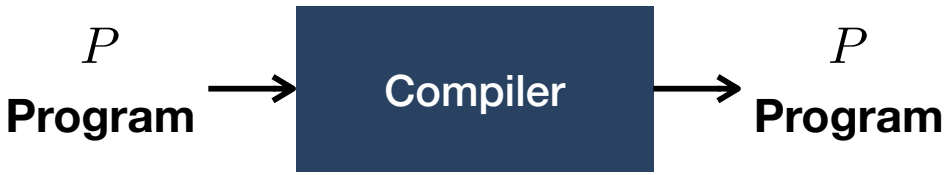
- You will design **syntax** and **semantics** of diverse target languages.
- You will implement **interpreters** of the target languages.
- You will use **Scala** as an implementation language.

Interpreters vs Compilers

- 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.

- A **compiler** takes a program and produces another program.



  - Good for **speed**, but more **complex**.
  - For example, `scalac`, `gcc`, `javac`, etc.

Roadmap: Growing a Language

PLRG

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
  - **Type Inference** – Type Variables, Type Unification
  - **Algebraic Data Types** – Variants, Pattern Matching
  - **Polymorphism** – Parametric Polymorphism, Subtype Polymorphism

Next Lecture

- Basic Introduction of Scala

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr