Lecture 0 – Course Overview

COSE212: Programming Languages

Jihyeok Park

2026 Fall

Course Information

- **Instructor:** Jihyeok Park (박지혁)
  - **Position:** Assistant Professor in CS, Korea University
  - **Expertise:** Programming Languages, Software Analysis
  - **Office hours:** By appointment via e-mail
  - **Office:** 507, Jung Woonoh IT & General Education Center
  - **Email:** jihyeok_park@korea.ac.kr

Course Information

- **Instructor:** Jihyeok Park (박지혁)
  - **Position:** Assistant Professor in CS, Korea University
  - **Expertise:** Programming Languages, Software Analysis
  - **Office hours:** By appointment via e-mail
  - **Office:** 507, Jung Woonoh IT & General Education Center
  - **Email:** jihyeok_park@korea.ac.kr

- **Class:** COSE212 - 02 (English)
  - 13:30–14:45, Mondays and Wednesdays
  - 610, Jung Woonoh IT & General Education Center

Course Information

- **Instructor:** Jihyeok Park (박지혁)
  - **Position:** Assistant Professor in CS, Korea University
  - **Expertise:** Programming Languages, Software Analysis
  - **Office hours:** By appointment via e-mail
  - **Office:** 507, Jung Woonoh IT & General Education Center
  - **Email:** jihyeok_park@korea.ac.kr

- **Class:** COSE212 - 02 (English)
  - 13:30–14:45, Mondays and Wednesdays
  - 610, Jung Woonoh IT & General Education Center

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

Course Information

- **Instructor:** Jihyeok Park (박지혁)
  - **Position:** Assistant Professor in CS, Korea University
  - **Expertise:** Programming Languages, Software Analysis
  - **Office hours:** By appointment via e-mail
  - **Office:** 507, Jung Woonoh IT & General Education Center
  - **Email:** jihyeok_park@korea.ac.kr

- **Class:** COSE212 - 02 (English)
  - 13:30–14:45, Mondays and Wednesdays
  - 610, Jung Woonoh IT & General Education Center

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

- **LMS:** https://lms.korea.ac.kr/
  - Please use the **Board** > **Q&A** section for questions.

Course Information

- **Instructor:** Jihyeok Park (박지혁)
  - **Position:** Assistant Professor in CS, Korea University
  - **Expertise:** Programming Languages, Software Analysis
  - **Office hours:** By appointment via e-mail
  - **Office:** 507, Jung Woonoh IT & General Education Center
  - **Email:** jihyeok_park@korea.ac.kr
- **Class:** COSE212 - 02 (English)
  - 13:30–14:45, Mondays and Wednesdays
  - 610, Jung Woonoh IT & General Education Center
- **Homepage:** https://plrg.korea.ac.kr/courses/cose212/
- **LMS:** https://lms.korea.ac.kr/
  - Please use the **Board** > **Q&A** section for questions.
- **Teaching Assistants:** cose212@googlegroups.com
  - Please contact the TAs via the group e-mail above.

Why Programming Languages in the Era of AI?

If **AI** writes the code, why learn the **language**?

Why Programming Languages in the Era of AI?

If **AI** writes the code, why learn the **language**?

- **AI Writes Code; We Read & Verify**
  To check that generated code captures our **intent**, we must know precisely what it **means** (its **semantics**).

Why Programming Languages in the Era of AI?

If **AI** writes the code, why learn the **language**?

- **AI Writes Code; We Read & Verify**
  To check that generated code captures our **intent**, we must know precisely what it **means** (its **semantics**).

- **The Need for Formal Semantics**
  AI is **probabilistic**, so the guarantee must come from the language itself – defined **mathematically**, not by intuition.

Why Programming Languages in the Era of AI?

If **AI** writes the code, why learn the **language**?

- **AI Writes Code; We Read & Verify**
  To check that generated code captures our **intent**, we must know precisely what it **means** (its **semantics**).

- **The Need for Formal Semantics**
  AI is **probabilistic**, so the guarantee must come from the language itself – defined **mathematically**, not by intuition.

- **Treating Programs as Data**
  Understanding a language lets us handle programs as **data** – so we can **analyze**, **verify**, **modify**, and **compile** them.

Goal of This Course

PLRG

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

Goal of This Course

PLRG

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

- **What you will achieve:**
  - **Read** any program and say precisely what it **means**.
  - **Evaluate** and pick the right language for a task, or **design** your own.

Goal of This Course

PLRG

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

- **What you will achieve:**
  - **Read** any program and say precisely what it **means**.
  - **Evaluate** and pick the right language for a task, or **design** your own.
- **How we will do it:**
  - **Design** languages with their **formal semantics**.
  - **Implement** their **interpreters** in Scala, treating programs as data.

Goal of This Course

PLRG

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

- **What you will achieve:**
  - **Read** any program and say precisely what it **means**.
  - **Evaluate** and pick the right language for a task, or **design** your own.
- **How we will do it:**
  - **Design** languages with their **formal semantics**.
  - **Implement** their **interpreters** in Scala, treating programs as data.
- **What this course is NOT:**
  - You will **NOT** learn **specific languages**.
  - This is **NOT** an introductory course; it assumes discrete mathematics, data structures, and theory of computation.

Interpreters vs Compilers

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

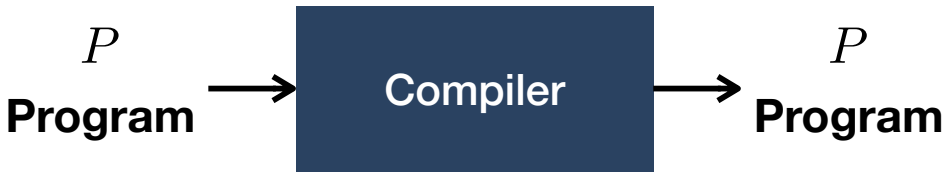
  - Good for **understanding** program behavior, easy to **implement**.
  - For example, `scala`, `python`, `bash`, desktop calculator, etc.
  - You will implement interpreters of various languages in this course.

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.
  - 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**.

Roadmap: Growing a Language

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

Roadmap: Growing a Language

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

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

Grading

- **3 Projects: 0%**
  - You will implement **three** different mini languages: MiniFSharp, MiniPython, and MiniScala.
  - They are **NOT graded** and need **no submission**.
  - However, some exam questions might be related to them, so we **strongly recommend** solving them by yourself.

Grading

- **3 Projects: 0%**
  - You will implement **three** different mini languages: MiniFSharp, MiniPython, and MiniScala.
  - They are **NOT graded** and need **no submission**.
  - However, some exam questions might be related to them, so we **strongly recommend** solving them by yourself.
- **Midterm exam: 45%**
  - October 21 (Wed.) 18:30 – 21:00 (150 min.)
  - In classroom, closed book, closed notes
- **Final exam: 45%**
  - December 16 (Wed.) 18:30 – 21:00 (150 min.)
  - In classroom, closed book, closed notes

Grading

- **3 Projects: 0%**
  - You will implement **three** different mini languages: MiniFSharp, MiniPython, and MiniScala.
  - They are **NOT graded** and need **no submission**.
  - However, some exam questions might be related to them, so we **strongly recommend** solving them by yourself.
- **Midterm exam: 45%**
  - October 21 (Wed.) 18:30 – 21:00 (150 min.)
  - In classroom, closed book, closed notes
- **Final exam: 45%**
  - December 16 (Wed.) 18:30 – 21:00 (150 min.)
  - In classroom, closed book, closed notes
- **Attendance: 10%**
  - Please use **LMS** to attend the class with the code provided.
  - **All or nothing:** $\geq 2/3$ of the checks $\rightarrow$ **10%**, otherwise **0%**.
  - **No** attendance check in the first week; it starts on **Sep. 7**.

Schedule

Week	Contents	Week	Contents
1	Course Overview	9	Continuations
2	Scala, Syntax and Semantics	10	Continuations, First-Class Cont.
3	Syntax and Semantics, Identifiers	11	Compiling with Cont., Type Systems
4	Identifiers, First-Order Functions	12	Typed Languages, Typing Recursion
5	First-Class Functions, Lambda Calc.	13	Algebraic Data Types
6	Recursion, Mutable Data Structures	14	Parametric and Subtype Polymorphism
7	Mutable Variables, Garbage Collection	15	Type Inference
8	Lazy Evaluation, **Midterm (Oct. 21)**	16	Course Review, **Final (Dec. 16)**

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

- Oct. 5 (Mon.) / 7 (Wed.) – International Conference
- Oct. 12 (Mon.) / 14 (Wed.) – International Conference

Instead, lecture videos will be uploaded to **LMS**, and you don't need to check the attendance on these days.

Course Materials

- **Self-contained lecture notes.**

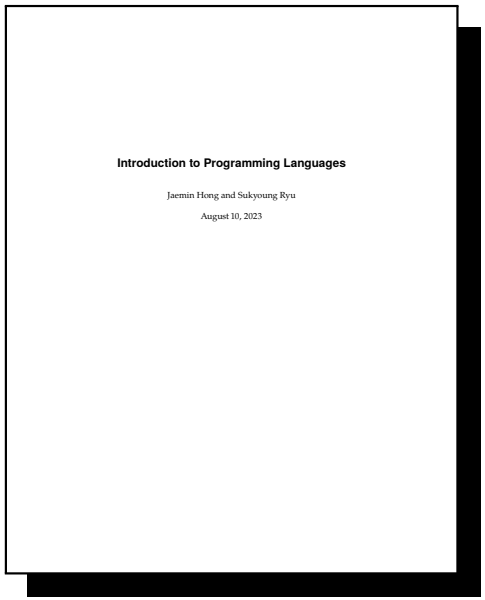
  https://plrg.korea.ac.kr/courses/cose212/

Course Materials

- **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

Next Lecture

- Basic Introduction of Scala

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