

# Lecture 0 – Course Overview

## COSE212: Programming Languages

Jihyeok Park



2024 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

- **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 - 01 (English)

- **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 - 01 (English)
- **Homepage:** <https://plrg.korea.ac.kr/courses/cose212/>

- **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 - 01 (English)
- **Homepage:** <https://plrg.korea.ac.kr/courses/cose212/>
- **Teaching Assistant:**
  - Jungyeom Kim (김준겸) – kimjg1119@korea.ac.kr
  - Seongmin Ko (고성민) – 2018320221@korea.ac.kr

- **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 - 01 (English)
- **Homepage:** <https://plrg.korea.ac.kr/courses/cose212/>
- **Teaching Assistant:**
  - Jungyeom Kim (김준겸) – kimjg1119@korea.ac.kr
  - Seongmin Ko (고성민) – 2018320221@korea.ac.kr
- **Discussion & Questions:** <https://campuswire.com/c/G2CA06AE4>



Passcode:

- **4 Homework Assignments: 30%**
  - Programming assignments in Scala (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.**

- **4 Homework Assignments: 30%**
  - Programming assignments in Scala (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 23 (Wed.) 18:30 – 21:00 (150 min.)
  - In classroom, closed book, closed notes
- **Final exam: 30%**
  - December 18 (Wed.) 18:30 – 21:00 (150 min.)
  - In classroom, closed book, closed notes



- **4 Homework Assignments: 30%**
  - Programming assignments in Scala (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 23 (Wed.) 18:30 – 21:00 (150 min.)
  - In classroom, closed book, closed notes
- **Final exam: 30%**
  - December 18 (Wed.) 18:30 – 21:00 (150 min.)
  - In classroom, closed book, closed notes
- **Attendance: 10%**
  - Please use [Blackboard](#) to attend the class **by yourself**.

Week	Contents	Week	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	<b>Midterm Exam (Oct. 23 - Wed.)</b>	16	<b>Final Exam (Dec. 18 - Wed.)</b>

On the four days listed below, there will be no offline lectures. Instead, lecture videos will be uploaded to [Blackboard](#).

- Sep. 16 (Mon.) / 18 (Wed.) – 추석
- Oct. 9 (Wed.) – 한글날
- Nov. 20 (Wed.) – External Schedule

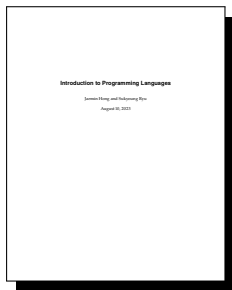
- **Self-contained lecture notes.**

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

- **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**

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

- Why?

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

- Why? After this course, you will be able to:

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.



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?

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**.

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.

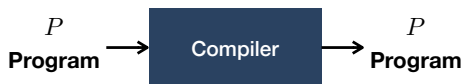


- Good for **understanding** program behavior, easy to **implement**.
- For example, `scala`, `python`, `bash`, desktop calculator, etc.
- You will implement interpreters of various languages in this course.

- 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.

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

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>