Jin-Soo Kim
(*jinsoo.kim@snu.ac.kr*)

Systems Software &
Architecture Lab.

Seoul National University

Spring 2023

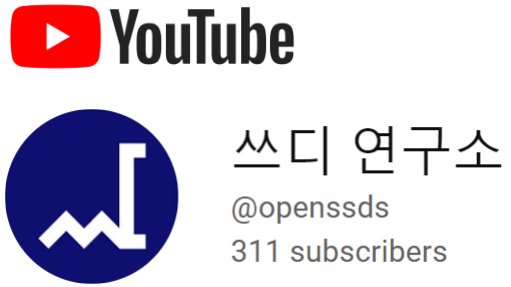
4190.103A: Programming Practice

Course Information

- Schedule
  - Lecture: 14:00 – 15:50 (Tuesday) @ Engineering Bldg. #302-311-1
  - Practice: 14:00 – 15:50 (Thursday) @ Engineering Bldg. #302-311-1
- 3 credits with the S/U grading
- Official language: Korean

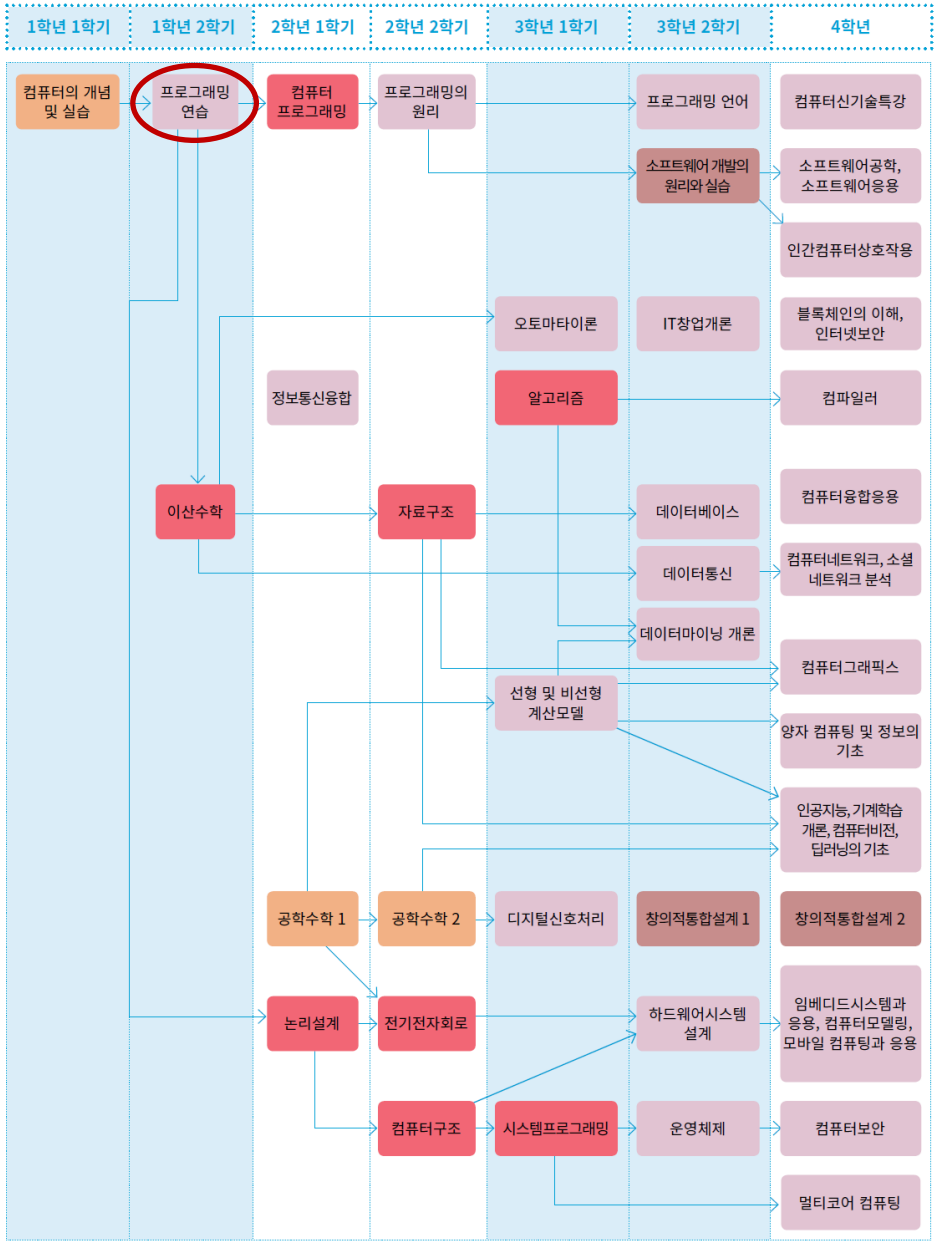
- TAs: 최재원, 박큰산, 이하은 (snucsl.ta@gmail)
- SNU eTL system for assignments and exam scores
- http://csl.snu.ac.kr/courses/4190.103/2023-1/ for announcements, lecture slides, and practice materials

About Me



- Jin-Soo Kim (김진수)
  - Professor @ CSE Dept.
  - Systems Software & Architecture Laboratory
  - Operating systems, storage systems, parallel and distributed computing, embedded systems, …

- E-mail: jinsoo.kim@snu.ac.kr
- Tel: 02-880-7302
- Office: Engineering Bldg. #301-504
- The best way to contact me is by email



	1학기		2학기	
1학년			4190.101*	이산수학
			4190.103A	프로그래밍연습
2학년	M1522.000600*	컴퓨터프로그래밍	4190.206A*	전기전자회로
	M1522.000700*	논리설계	4190.209	컴퓨터공학세미나
	400.021	정보통신융합	4190.210	프로그래밍의 원리
			4190.308*	컴퓨터구조
			M1522.000900*	자료구조
			( )*	공과대학 공통교과목
3학년	4190.306	오토마타이론	4190.307	운영체제
	4190.313	선형 및 비선형 계산모델	4190.309A	하드웨어시스템설계
	4190.407*	알고리즘	4190.310	프로그래밍언어
	4190.416A	디지털신호처리	M1522.000200	창의적통합설계1
	M1522.000800*	시스템프로그래밍	M1522.001400	데이터마이닝 개론
			M1522.001800	데이터베이스
			M1522.002100	데이터통신
			M1522.002400	소프트웨어 개발의 원리와 실습
			M1522.002700	IT창업개론
4학년	4190.303C	임베디드시스템과 응용	4190.403	소프트웨어응용
	4190.402	소프트웨어공학	4190.406B	모바일 컴퓨팅과 응용
	4190.408	인공지능	4190.412	컴퓨터모델링
	4190.409	컴파일러	4190.414A	멀티코어 컴퓨팅
	4190.410	컴퓨터그래픽스	4190.415	컴퓨터보안
	4190.411	컴퓨터네트워크	4190.423	컴퓨터융합응용
	4190.422	IT-리더십세미나	4190.426A	인간컴퓨터상호작용
	4190.427	소셜 네트워크 분석	4190.428	기계학습 개론
	M1522.000300	창의적통합설계2	M1522.001000	컴퓨터비전
	M1522.001100	컴퓨터 시스템 특강	M1522.001200	컴퓨터 신기술 특강
	M1522.002800	블록체인의 이해	M1522.002300	인터넷 보안
	M2177.004300	딥러닝의 기초		

Textbook

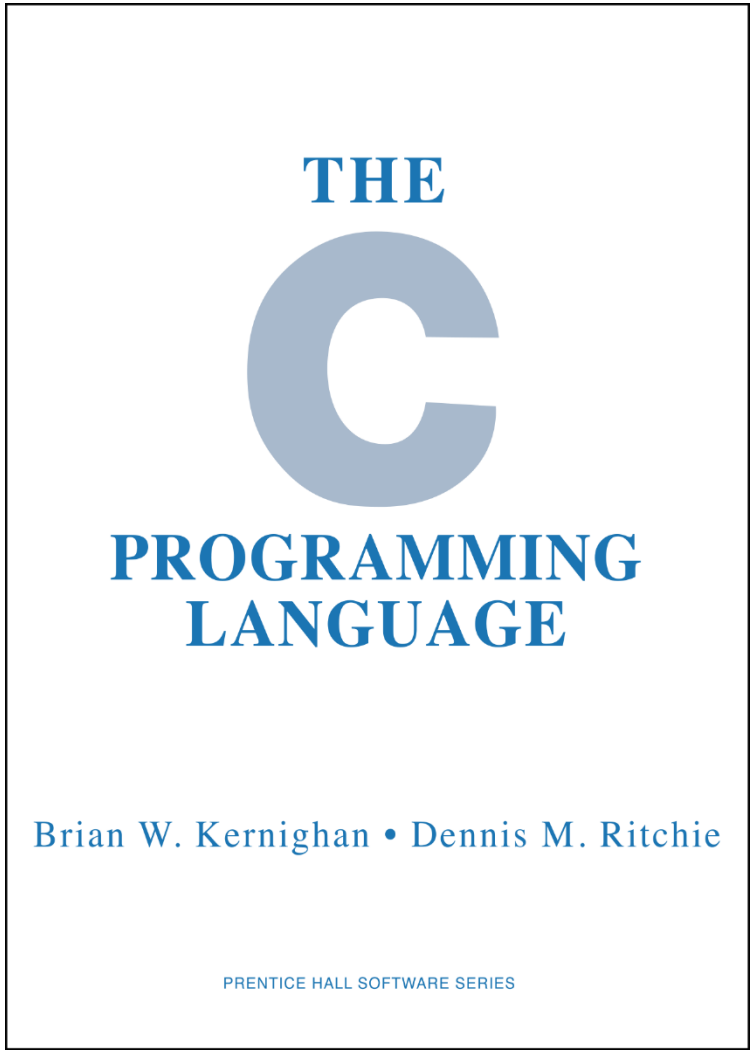
- A Book on C

  - Al Kelley and Ira Pohl
  - Fourth Edition
  - Addison-Wesley, 1998
  - https://users.soe.ucsc.edu/~pohl/abc4.html

  - The Korean version is also OK

Reference

- The C Programming Language (*a.k.a* K&R book)
  - Brian W. Kernighan and Dennis M. Ritchie
  - Second Edition
  - Prentice Hall, 1988



Attendance Policy

- Assigned seating
  - The seat you occupy on March 9th will be designated as your assigned seat for the first half of the semester
- Excused absences
  - If you have a compelling reason, you can request an "excused absence" with appropriate supporting documents
- Some examples of excused absences:
  - COVID-19 vaccination or confirmed positive
  - Other illness or accidents with significant injuries or death of a close family member
  - Official university/department activities
  - Reserve force training, etc.

Grading Policy

- Practice examples (pe): 20%
  - 20% deducted for every single day delay
- Practice assignments (pa): 40%
  - 20% deducted for every single day delay
- Final exam (ex): 40%
  - There will be no midterm exam

Cheating Policy

- What is cheating?
  - Copying another's solution (or one from the Internet) and submitting it as your own
  - Allowing another student to copy your solution (Don't put your solution in Github)
  - It will be decided solely based on code similarity
- What is NOT cheating?
  - Helping others use systems or tools, or debug their code
  - Helping others with high-level design issues
- Penalty for cheating
  - Both copier and source provider will be reported to the CSE dept.'s committee
  - Depending on the committee decision, you can fail this course
  - The result will be shared with your original dept. (if any)

How To Pass This Course

```c
#define ALPHA 0.4
#define BETA  0.4

// cheat: 1 if there is any cheating in practice examples/assignments or final exam, else 0
// ua: the number of unexcused absences
// pe: practice examples score ([0, 20])
// pa: practice assignments score ([0, 40])
// ex: final exam score ([0, 40])

int pass(int cheat, int ua, float pe, float pa, float ex)
{
    if (!cheat && (ua <= 4) && (pe + pa >= ALPHA*60.0) && (ex >= BETA*40.0))
        return 1;    /* PASS: Grade S */
    else
        return 0;    /* FAIL: Grade U */
}
```

- The values of ALPHA and BETA are subject to change

Algorithmic Thinking

- Very diligent
- But, not so smart
- Can do a few of simple operations (instructions)
- Complex operation: a series of simple operations

- Must tell in detail what to do
  - Understandable to computer
  - For all possible cases
- Algorithmic thinking
  - Programs == Recipes

Program? ≈ Recipe!

준비시간 :10분, 조리시간 :10분

재료
라면 1개, 스프 1봉지, 오징어 1/4마리, 호박 1/4개, 양파 1/2개, 양배추 1장, 당근 1/4개, 물 3컵(600cc)

Ingredients ≈ Data

만드는 법
1.오징어는 껍질을 벗기고 깨끗하게 씻어 칼집으로 모양을 낸다.
2.호박, 양파, 양배추는 모두 채썬다.
3.냄비에 물 3컵을 붓고 끓인다.
4.물이 끓으면 스프를 넣고 오징어와 야채를 넣어 충분히 맛이 우러나도록 5분 정도 끓여준다.
5.끓으면 면을 넣어 익힌다.

Directions ≈ Instructions

Source: http://user.chol.com/~yugenie/yo/jjambong.html

Programming Languages

- Algorithms: Developed by people

Programming Languages

High-level languages

Assembly languages

Machine languages

- Computers: Execute algorithms

The C Programming Language

- Developed by Dennis Ritchie in the early 1970s with UNIX

- Small
  - Fewer keywords
- Portable
  - Code written on one machine easily moved to another
- Terse
  - A very powerful set of operators
  - Able to access the machine in the bit level

- The basis for C++ and Java

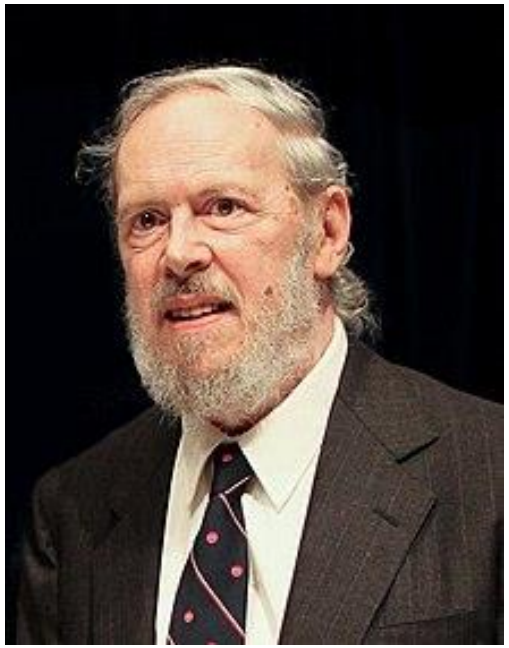


Image source: Wikipedia

History of C

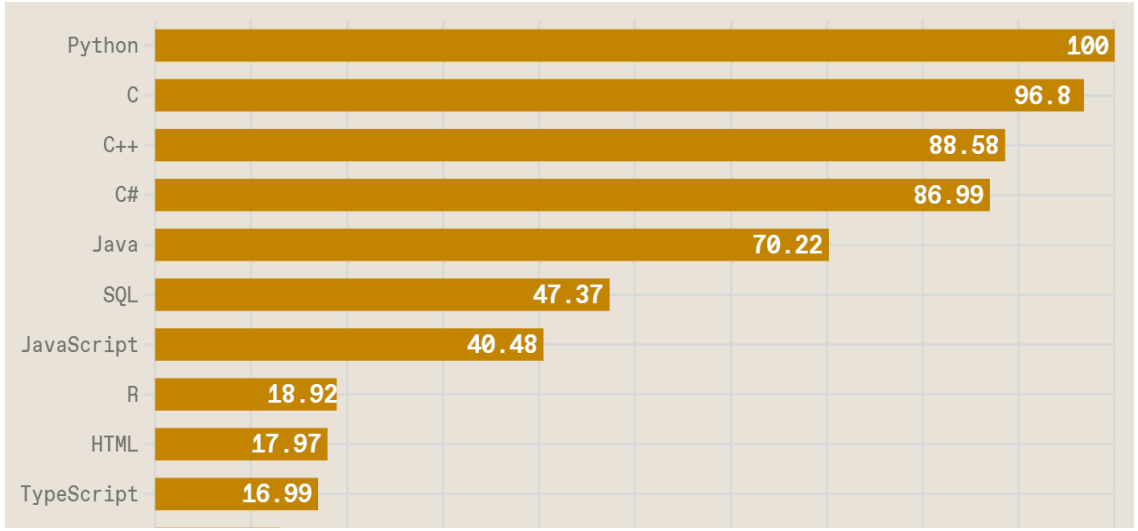
- 1972 Ritchie developed C by improving the B language
- 1973 The UNIX kernel was rewritten in C
- 1978 K&R book published (an informal specification: K&R C or C78)
- 1989 Standardized by ANSI as X3.159-1989 (ANSI C or C89)
- 1990 Adopted by ISO as ISO/IEC 9899:1990 (C90 == C89)

  ↳ We will focus on this version
- 1999 Revised by ISO as ISO/IEC 9899:1999 (C99)

  Added inline functions, new data types, etc. (supported by major C compilers)
- 2011 ISO/IEC 9899:2011 (C11)
- 2017 The most recent standard ISO/IEC 9899:2018 (C17)

Why C?

- Simple, elegant, fast, and powerful
- Widely used
  - The 2nd Top Programming Language (IEEE Spectrum 2022)



- "Relatively low-level"
  - Highly portable (easy to compile and debug)
  - Good for systems software (operating systems and utilities)
  - Good for embedded systems
- Great for understanding how computers work

Why Not C?

- No automatic array bound checking
  - One of the most severe security vulnerabilities
- Manual memory management
  - `malloc()` and `free()`
- No built-in data structures other than arrays
- Complicated syntax
- Multiple use of such symbols as * and =
  - `a = a * p` *vs.* `a = *p` *vs.* `a = **p`
  - `a = b` *vs.* `a == b`
- Not object-oriented

A Vulnerability in GNU C Library Could Allow for Remote Code Execution

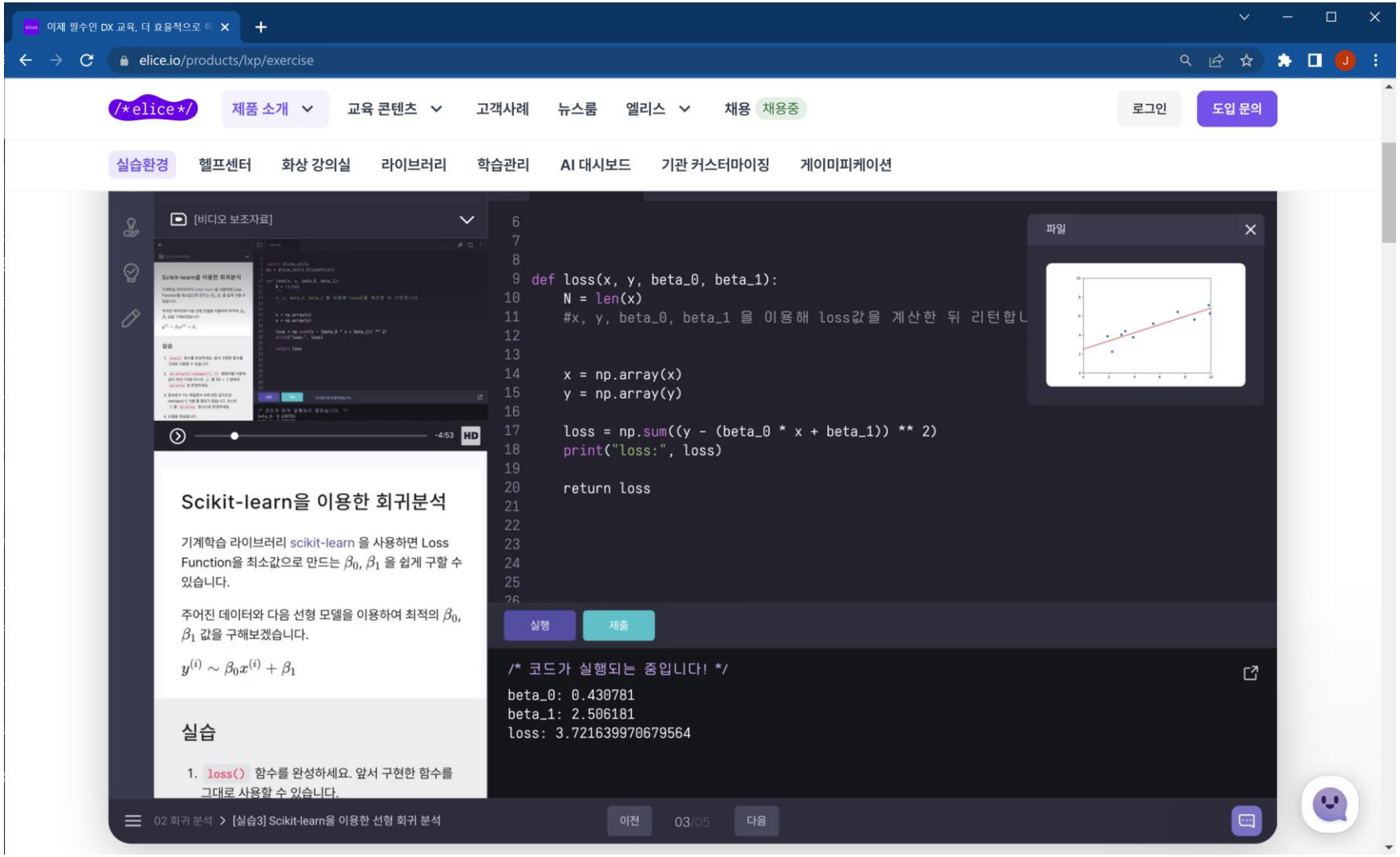
MS-ISAC ADVISORY NUMBER:
2020-105

DATE(S) ISSUED:
08/04/2020

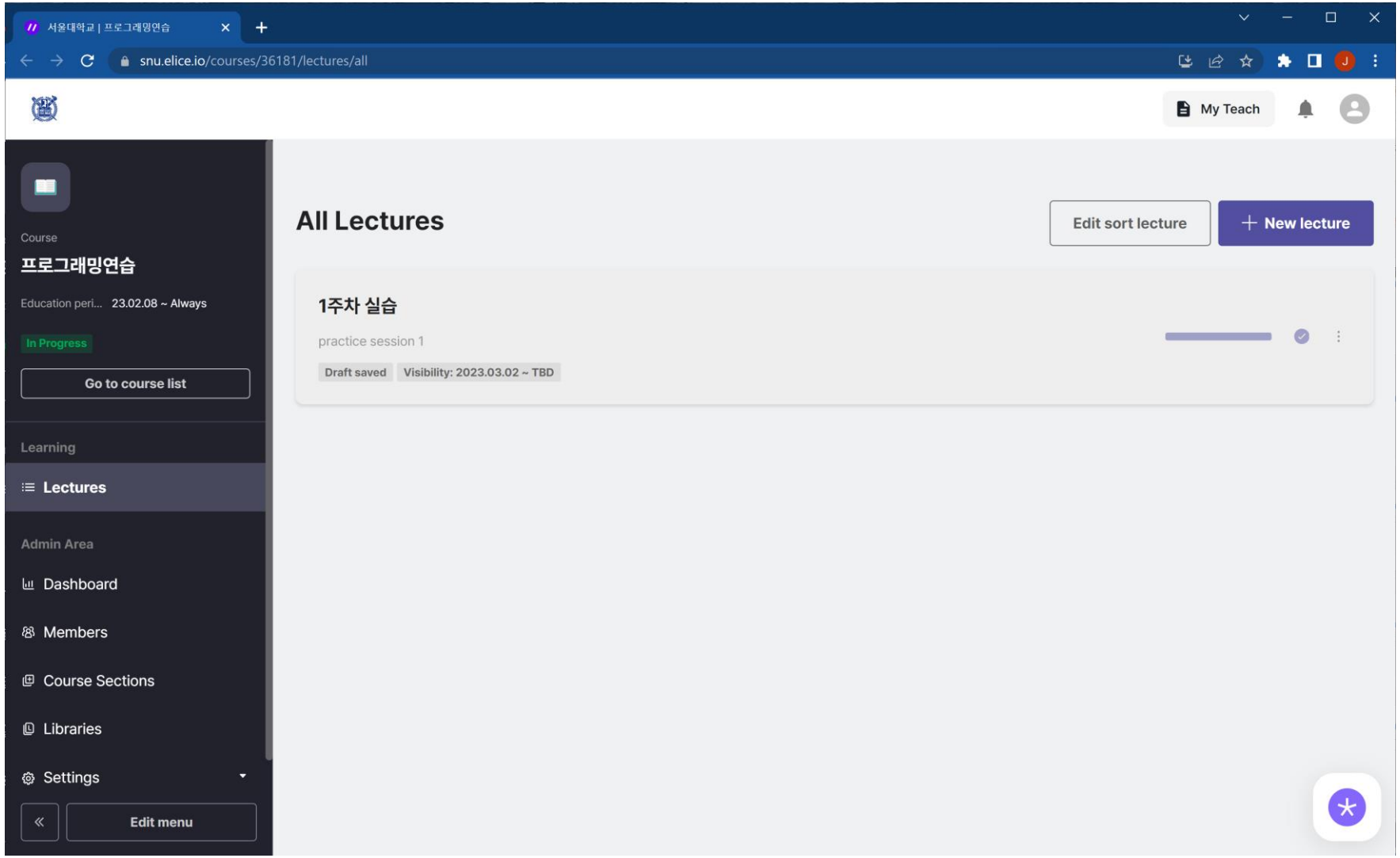
Topics

- Overview (Chap. 1)
- Lexical Elements, Operators, and the C System (Chap. 2)
- Fundamental Data Types (Chap. 3)
- Flow of Control (Chap. 4)
- Functions (Chap. 5)
- Arrays, Pointers, and Strings (Chap. 6)
- Structures and Unions (Chap. 9)
- List Processing (Chap. 10)
- Bitwise Operators and Enumeration Types (Chap. 7)
- Preprocessor (Chap. 8)
- File Input/Output (Chap. 11)

Practice Session with Elice

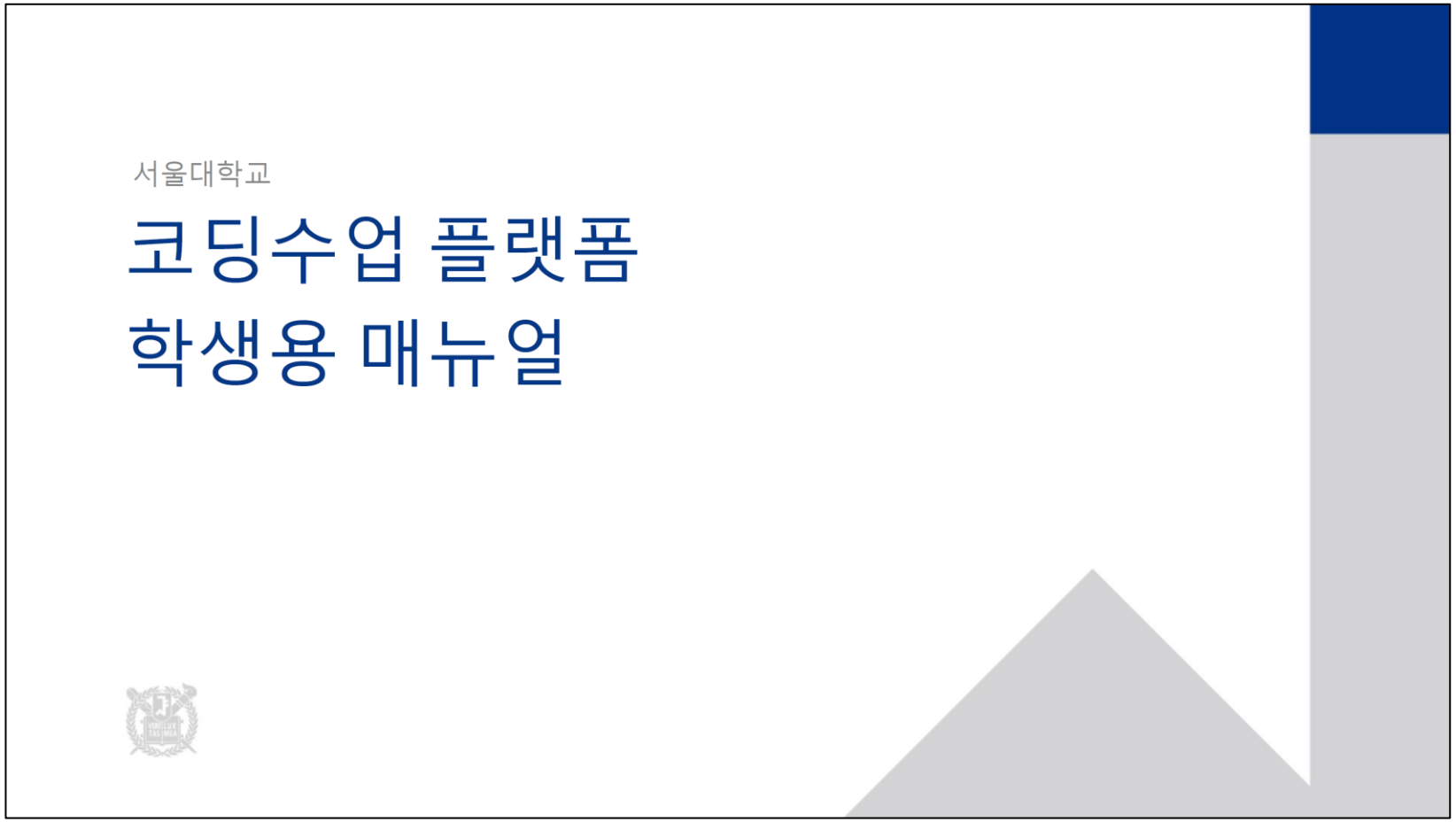


Elice @ eTL


서울대학교 | 프로그래밍연습
snu.elice.io/courses/36181/lectures/all
My Teach
Course
프로그래밍연습
Education peri... 23.02.08 ~ Always
In Progress
Go to course list
Learning
Lectures
Admin Area
Dashboard
Members
Course Sections
Libraries
Settings
Edit menu
All Lectures
Edit sort lecture
New lecture
1주차 실습
practice session 1
Draft saved
Visibility: 2023.03.02 ~ TBD

Elice Manual

- Click to download!



How to Learn Programming

- Learn by doing
  - Do exercises/practices
  - Lectures will give you basic tools only

- In the lectures, you will learn:
  - Language syntax
  - Algorithmic thinking

- Read "An Overview of C (Chapter 1)" & try by yourself

Warning!!!

- Lectures
  - Seem easy
- Textbook examples
  - Seems that you understand well
- Programming assignments
  - More difficult than it seems
- Expect many bugs in your programs

Programming maturity comes with p.r.a.c.t.i.c.e.!!