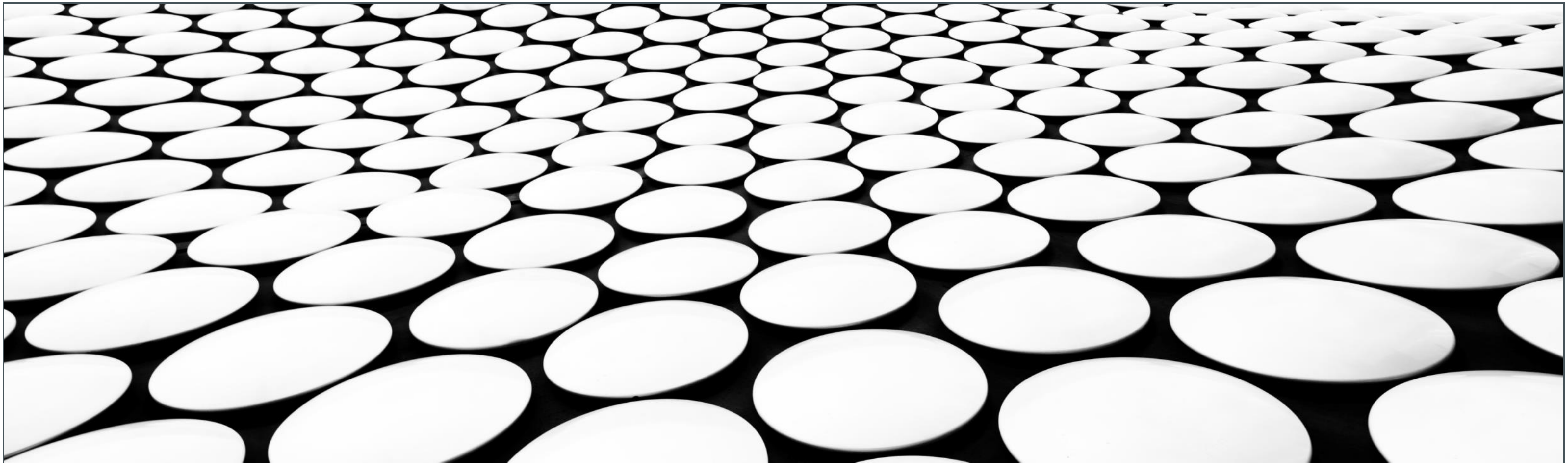
COMPUTATIONAL THINKING

COURSE ORIENTATION



학습 개요

Orientation

1. 강사소개 / 과목 운영 구성원 소개
2. SKKU 컴퓨팅 사고 교양 커리큘럼 소개
3. 수업방법 , 평가방법 (Syllabus 포함)
4. 과제, 평소학습 제출기한
5. 참고 도서 소개

강사 소개

- 컴퓨팅 사고와 SW코딩

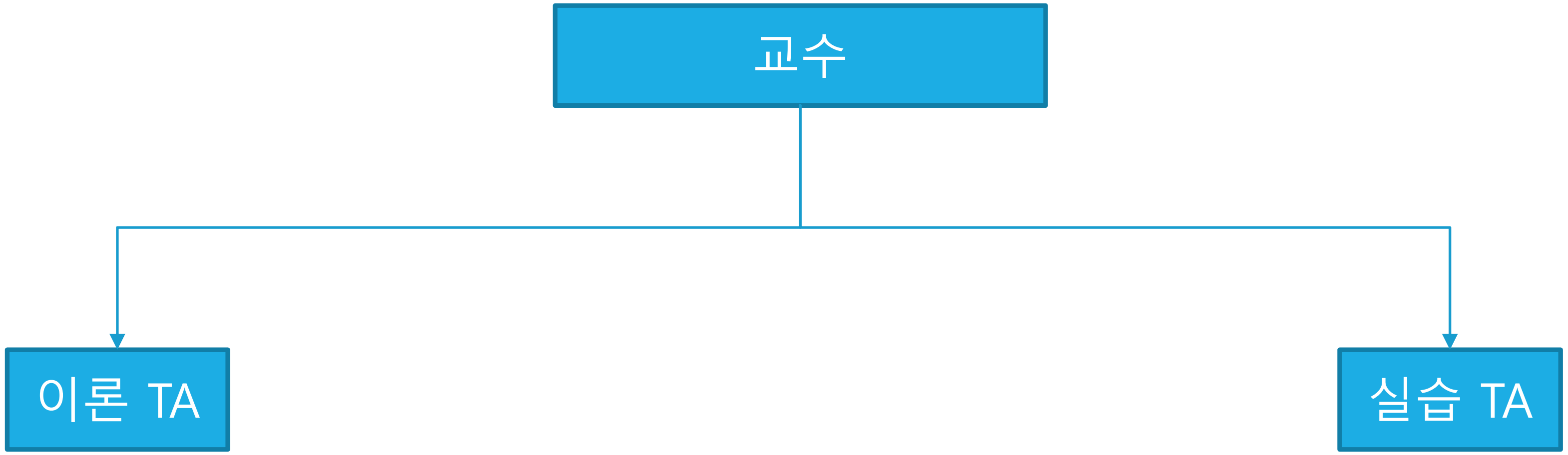
이름 : 황숙희

전공 : 컴퓨터공학 / 인공지능

연구실 : 공학 1관 21539 호

E-mail : mamipapa714@skku.edu

TA (TEACHING ASSISTANCE)



교수

이론 TA

실습 TA

운영중심

중간고사,기말고사 운영

질의응답

과제 보조

파이썬 실습 중심

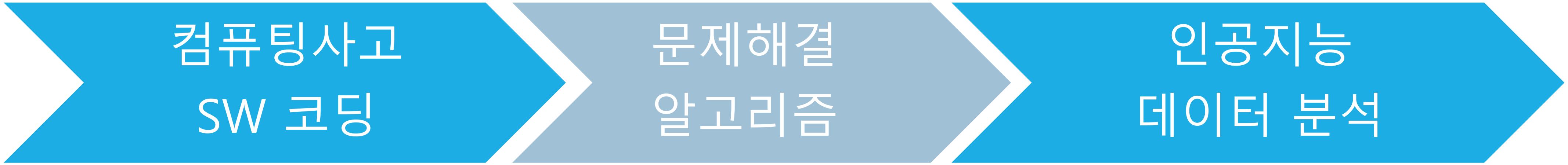
오픈채팅방 운영

질의응답

과제 보조

SKKU 컴퓨팅 사고 교양 커리큘럼

1학년 1학기 | 1학년 2학기 | 2학년



컴퓨팅사고
SW 코딩

문제해결
알고리즘

인공지능
데이터 분석

- 컴퓨팅사고의 개념
- 파이썬 프로그램 언어의 기본 문법 및 연습
- 간단한 문제 해결

- 자료구조와 알고리즘
- 컴퓨터로 문제해결 하기
- 러닝 페어 (문제 제시 및 해결 모델 제시 및 구현)

- 파이썬에 포함된 인공지능 라이브러리를 이용한 프로그래밍 연습
- 파이썬에서 제공하는 데이터분석 라이브러리를 이용한 프로그래밍 연습

컴퓨팅 사고력을 기반으로 문제해결을 할 수 있는 능력 향상 및 인재 양성

수업 방법 및 평가 방법

- 성적평가는 다음의 사항들을 통합적으로 고려하여 산출 **(상대평가)**

항목	비중	설명
출석	20%	- i-campus 시스템 설정에 따른 점수 배정 - 95% 이상 시청 출석인정
과제	10%	- 2회
퀴즈	10%	- 4회
평소학습	25%	- 6회
중간고사	15%	- 7주차 주말 (8주차 강의 없음)
기말고사	20%	- 15주차 주말 (16주차 강의 없음)
합 계	100%	

SYLLABUS

주차	주제	수업내용	수업 유형	학습 활동
1	Course Orientation	▪ Course Orientation ▪ 컴퓨팅 사고	온라인 녹화강의	온라인 동영상
2	이론 : 컴퓨터	▪ 튜링기계 ▪ 컴퓨터의 실현 ▪ 과제1		온라인 동영상 & **과제1**
3	이론 : 소프트웨어 실습 : 파이썬 실습	▪ 소프트웨어 ▪ 파이썬 소개 ▪ 파이썬 설치		온라인 동영상 & **퀴즈1**
4	이론 : 프로그래밍 언어 실습 : 파이썬 실습	▪ 프로그래밍 언어 ▪ 변수		온라인 동영상 & **퀴즈2**
5	이론 : 컴퓨팅 사고의 여러가지요소 실습 : 파이썬 실습	▪ 컴퓨팅사고의 여러가지 요소 ▪ 기본자료형 (수치, 문자, 논리) ▪ 중간고사안내		온라인 동영상 & **퀴즈3**
6	이론 : 컴퓨팅 사고 요소 실습 : 파이썬 실습	▪ Decomposition (분해) ▪ Pattern Recognition (패턴인식) ▪ Abstraction (추상화) ▪ 표준 입출력		온라인 동영상 & **퀴즈4**
7	이론 : 컴퓨팅 사고 요소 실습 : 파이썬 실습	▪ Algorithm (알고리즘) ▪ 과제2		온라인 동영상 & **과제2**
8	중간고사	▪ 중간고사		온라인 실시

SYLLABUS

주차	주제	수업내용	수업 유형	학습 활동
9	이론 : 자료형을 이용한 문제해결 실습 : 파이썬 실습	▪ 컬렉션 자료형 1 (String, list, tuple) ▪ 평소학습1	온라인 녹화강의	온라인 동영상 & 평소학습1
10	이론 : 자료형을 이용한 문제해결 실습 : 파이썬 실습	▪ 컬렉션 자료형 2 (set, dictionary) ▪ 연산자 ▪ 평소학습2		온라인 동영상 & 평소학습2
11	이론 : 제어문을 이용한 문제해결 실습 : 파이썬 실습	▪ 제어문 이해, 선택문 이해 ▪ If 문 실습 ▪ 평소학습3 ▪ (평소학습1 소스코드 제공)		온라인 동영상 & 평소학습3
12	이론 : 제어문을 이용한 문제해결 실습 : 파이썬 실습	▪ 반복문 이해1 ▪ While 문 실습 ▪ 평소학습4 ▪ (평소학습2 소스코드 제공)		온라인 동영상 & 평소학습4
13	이론 : 제어문을 이용한 문제해결 실습 : 파이썬 실습	▪ 반복문 이해2 ▪ for 문 실습 ▪ 평소학습5 ▪ (평소학습3 소스코드 제공) ▪ 기말고사 안내		온라인 동영상 & 평소학습5
14	이론 : 함수를 이용한 문제해결 실습 : 파이썬 실습	▪ 함수의 이해 ▪ 함수 실습 ▪ 평소학습6 ▪ (평소학습4 소스코드 제공)		온라인 동영상 & 평소학습6
15	이론 : 모듈을 이용한 문제해결 실습 : 파이썬 실습	▪ 모듈의 이해 (Tkinter 이론) ▪ Tkinter 실습 ▪ 총정리 ▪ (평소학습5 소스코드 제공)		온라인 동영상
16	기말고사	▪ 기말고사 ▪ (평소학습6 소스코드 제공)		온라인 실시

수업 방법 및 평가 방법

온라인 녹화 강의 -20%

- 동영상은 **월요일부터 일요일까지** 오픈
- 해당 주차의 동영상(들)을 끝까지 들어야만 출석 인정 (끝까지 듣지 않은 경우는 결석(**95%** 까지 인정))
- 해당 주차가 지나면 수강은 가능하나 출석 인정 안됨
- 출석 점수는 ecampus 시스템에서 자동 산출됨(동영상 개수에 따라 산출)

퀴즈 – 10%

- **4**회 실시 / 배점 8점 / 1문제당 1점 / 총 8 문제
- 10분간 실시 / 문제 되돌아가기 불가 / 재시험 불가
- 해당주차의 교재, 동영상, 강의자료 기반 문제

수업 방법 및 평가 방법

- **과제 – 10%**
  - 2회 제출 / 배점10점
  - 지각 제출시 4 점 감점
  - 채점기준 : 제시한 내용이 모두 포함되어 있는가 / 과제 내용이 정확한가 / 제출형식을 잘 지켰는가

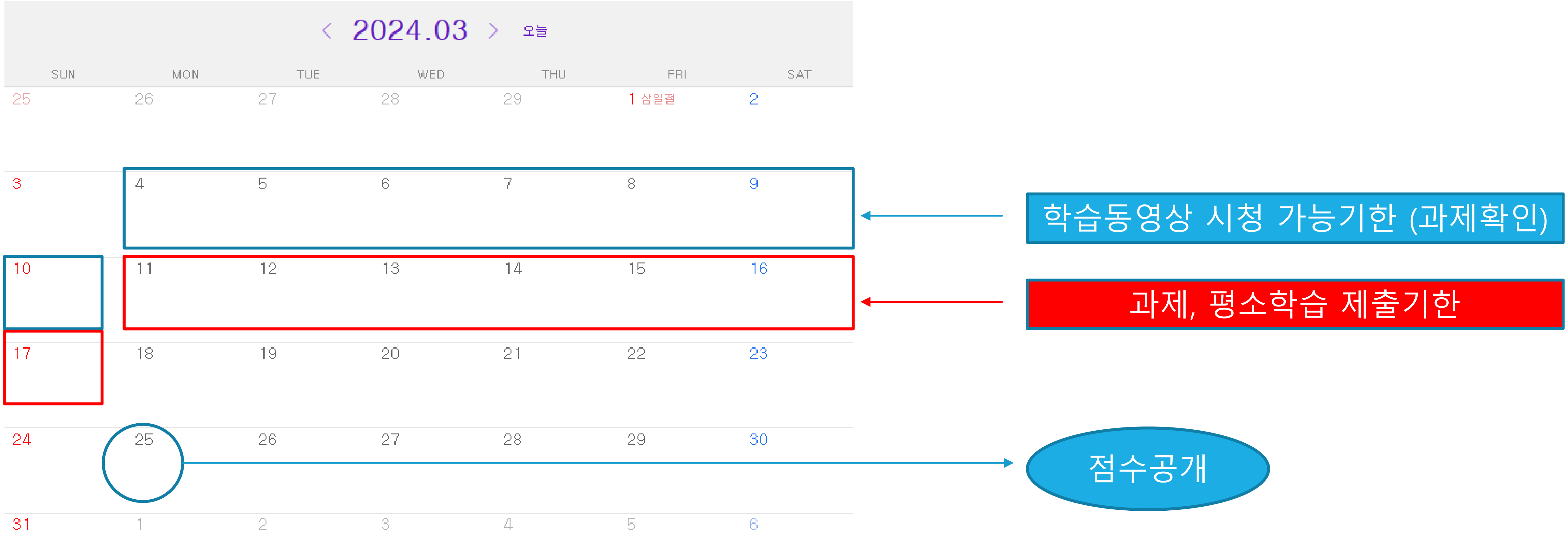
- **평소 학습** (파이썬 실습내용) – **25%**
  - 6회 제출 / 배점 5점
  - 지각 제출 불가
  - 채점기준 : 제시한 조건을 모두 만족하는가 / 강의 내용에 충실하게 작성하였는가 / 프로그램 오류는 없는가 / 제출형식을 잘 지켰는가

수업 방법 및 평가 방법

- **중간고사 (15%) , 기말고사 (20%)**

  - 학교 지정일자에 실시 (토요일 OR 일요일) - 재공지
  - 온라인 / 오픈북
  - 교재, 동영상, 강의자료 기반 주,객관식 혼합 문제
  - 제한시간 있음 : 문제 난이도에 따라 추후 공지
  - 중간고사 – 1주~7주차 내용
  - 기말고사 – 9주차~14주차 내용(파이썬 중심)

과제, 평소학습 제출기한



< 2024.03 > 오늘

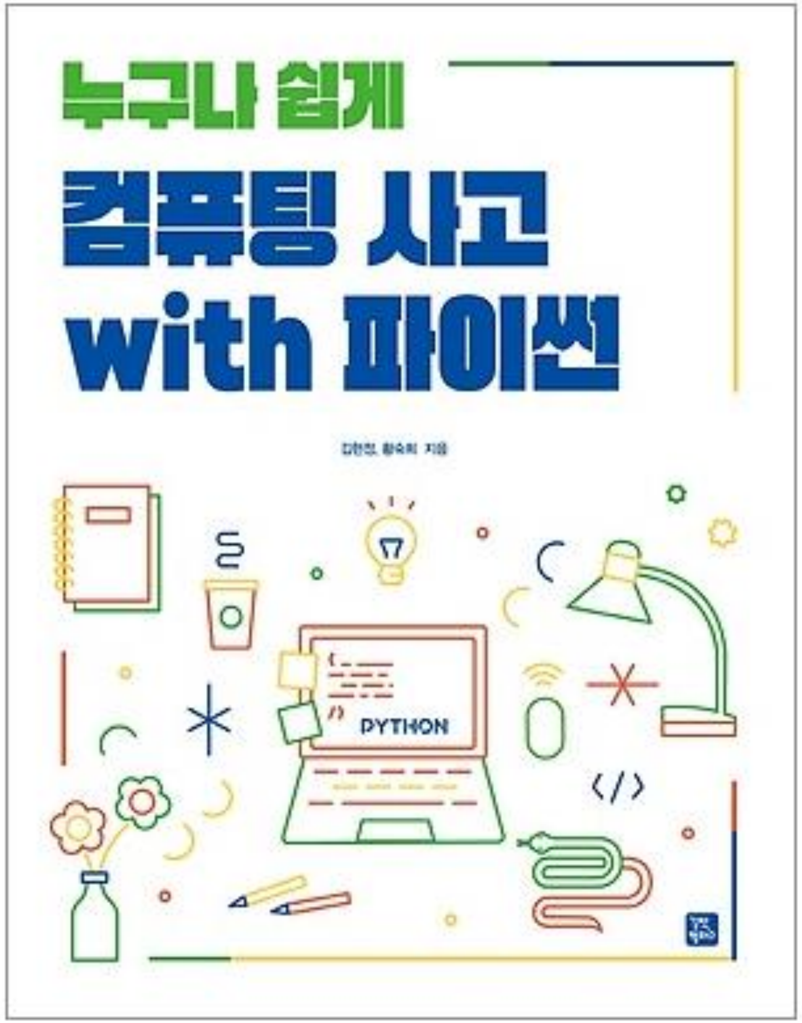
SUN	MON	TUE	WED	THU	FRI	SAT
25	26	27	28	29	1 삼일절	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

학습동영상 시청 가능기한 (과제확인)

과제, 평소학습 제출기한

점수공개

교재



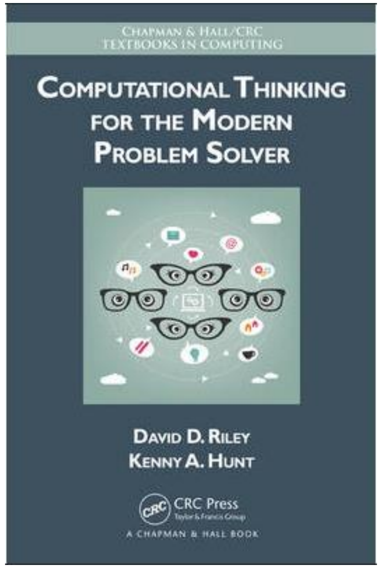
누구나 쉽게

컴퓨팅 사고 with 파이썬

김현정, 황숙희 지음

출판사 : 길벗 캠퍼스

참고 도서 소개



- computational thinking for modern problem solver
- - David D.Riley / Kenny A.Hunt

컴퓨팅
사고
소프트웨어를 통한
문제해결
COMPUTATIONAL THINKING for The Modern Problem Solver
David D. Riley · Kenny A. Hunt 지음
문봉교 · 김웅섭 공역
INFINITY BOOKS

- 컴퓨팅 사고 (소프트웨어를 통한 문제해결)
- - 번역본

참고 도서 소개

Viewpoint Jeannette M. Wing

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions precisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

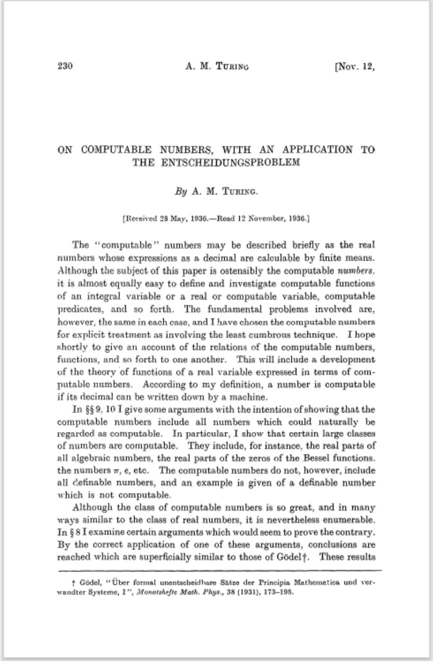
Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

LISA HANEY

COMMUNICATIONS OF THE ACM March 2006/Vol. 49, No. 3 33

- computational thinking
- Jeannette Marie Wing



230 A. M. Turing [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

- on computable numbers, with an applicable to the Entscheidungsproblem
- Alan Mathison Turing

컴퓨팅사고력 향상과 SW코딩 능력 배양을 위한

컴퓨팅사고와 SW코딩

감사합니다.

HWANG, SOOK HI