고급문제해결 과목 소개 & 운영 방법, 이시형(컴퓨터 학부)

01. 한 학기 동안 무엇을 어떻게 배우는가?

02. 수업 운영 방식

03. 평가 방법

04. 그 외 유의사항

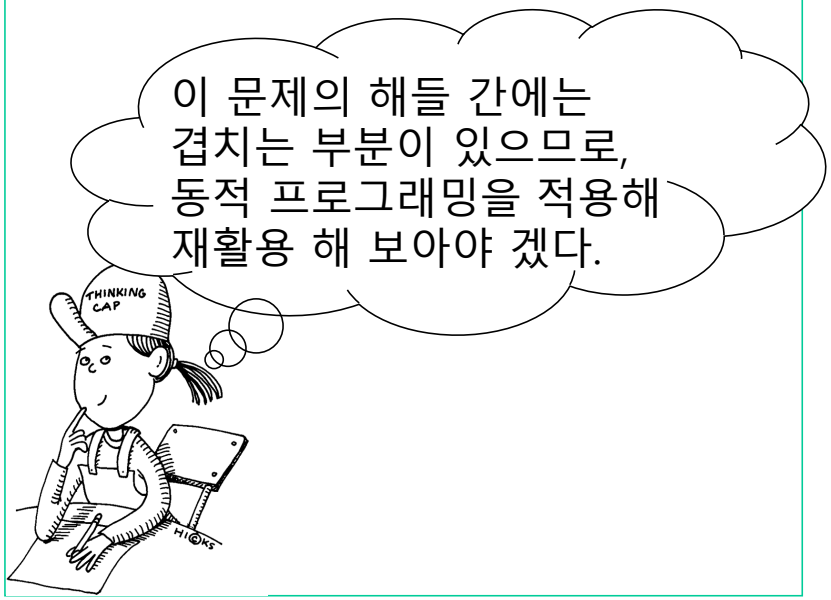
05. 다음 수업 전까지 해야 할 일 정리

무엇을 배우는가?

- 프로그래밍 문제(알고리즘 설계 문제)를 접했을 때 어떤 과정을 거쳐 해결하는지를 배움 - 어떻게 접근하고, 어떻게 해법을 생각해 내는지
- 문제에 대한 해법을 어떻게 최적화하는지 배움 (더 빠르고 더 적은 메모리를 사용하도록)
- 스스로 생각하는 방법을 배움
- 기존에 존재하는 알고리즘 일부 배우거나 활용

어떻게 배우는가?

- 많은 문제를 풀어보며 해법 생각하고 최적화하는 연습을 함
- 문제 풀이 과정에서 유의할 사항들이 습관이 되도록 함



매 주 1~2개 문제를 작은 단계로 나누어 풀이

개념 설명
단계 #1(sub-problem #1) 풀이,
학생 & 강사와 토의, 질문

간단한 예제
통해 문제 이해

개념 설명
단계 #2(sub-problem #2) 풀이,
학생 & 강사와 토의, 질문

1번째
알고리즘 구현

개념 설명
단계 #3(sub-problem #3) 풀이,
학생 & 강사와 토의, 질문

수학적 사고
중요

2번째
알고리즘 구현

지금까지 생각해 온 해법을
프로그램으로 구현하는
실습 문제 풀이

각자 해법 생각해 보고
옆 학생 & 강사와 논의

- 각 단계별 문제를 확실히 이해하며 진행하여, 결국 전체를 잘 이해하도록 함

자료구조/알고리즘

- 현존하는 대표적이고 활용도 높은 알고리즘과 자료구조 이해
- 필요할 때 적절한 방법 잘 선정해 활용할 수 있기 위함
- 여러 코드 예제 보며 코드 작성 능력 향상

고급문제해결

- 문제 풀이 과정 (문제 이해 → 해법 생각 → 최적화) 연습에 초점
- 즉 우리만의 알고리즘을 만들어가는 과정에 초점
- 이러한 과정에서 현존하는 알고리즘/자료구조도 그 일부분으로 복습하고 활용함

코딩 테스트를 대비한 과목?

- 취업 시 코딩 테스트에 도움 되었다는 수강생도 있음

- 하지만 코딩 테스트를 주 목적으로 만들었거나
- 단시간에 빠르게 문제 풀이하는 능력 습득을 주 목표로 하지는 않음

- 연구/개발 중 겪게 되는 문제 풀이에 도움되기 위한 목적으로 준비함
- 더 **일반적인 문제 해결 능력** 습득을 목표로 함

<그 외 수업에 대한 정보>

- 알고리즘 1~2 수업보다 어렵지는 않음

- 기존 수업자료 없이 새로 만들기 시작한 수업으로 3학기 째 개선 & 수정 중
- 이번 학기에는 특히 Java → Python 언어로 변경하면서 수업 자료나 코드 등을 수정하고 있음

선수 지식

- **고등학교 수준 수학**

- **자료구조, 알고리즘 1 수강**
- 기본 자료구조 알고 있으며 (예: 트리, 심볼 테이블)
- 해법의 실행 속도를 대략적으로 생각할 수 있으면 됨 (예: N^2에 비례)

- **프로그래밍 기본 개념 이해: array, recursion, class/object, reference/pointer**
- 생각한 문제 해결 방법을 코드로 작성 가능하다고 가정

- **Python 프로그래밍 가능:** 함수, 리스트, 클래스 활용, Python 설치, 추가 모듈 설치(pip install [모듈이름]), 개발환경 설치/사용법, 디버깅 등을 알고 있다고 가정
- 특히 Python 설치, 개발환경 설치, 환경 설정은 각자 할 수 있다고 가정

실습에 필요한 선수 지식 - Python 코딩 & 개발환경 설치 가능

- **Python** 언어 사용 가능 (수업자료 예제, **실습 과제, 시험 중 실습 문제 풀이에 Python 사용**)
  - 다른 언어보다 간결해 알고리즘 본질에 집중 가능
  - 설치 간편하고 빠름, 설치 오류 적음
  - 모든 패키지, 함수 알 필요 없으나, 실습에서 패키지 A의 함수 B 사용하도록 힌트 주면 입력 파라미터 의미, 반환 값 의미, 사용 예 등은 스스로 검색하고 테스트해 사용할 수 있어야 함
  - 예1: sorted() 함수 사용해 날짜 나타내는 3-tuple 리스트를 년도-월-일 순으로 정렬하시오. [(1999,5,3), (2021,6,21), (2011,4,30), …]
  - 예2: math.atan2() 함수 사용해 점 a, b, c가 이루는 각도 구하시오.
- Python 설치 및 개발환경 설치, 환경 설정은 각자 할 수 있다고 가정

- Python 및 개발환경(IDE)은 **개인 PC**에 설치해 **사용 권장** (컴퓨터 전공자는 스스로 환경 설정 가능해야 함 + 실습실 공용 PC는 오류 잦음)
- 각자 선호하는 개발환경 설치해 사용 가능 (IDLE, 메모장/cmd, Visual Studio Code, PyCharm, …)
- 실습실 공용 PC 사용하는 경우: Python은 설치되어 있으나, IDLE 외 개발 환경은 학교에서 일괄 설치하지 않으므로 필요한 환경은 수업 시작할 때 각자 설치해 사용하세요.

효율적인 문제해결 방법 & 알고리즘이 왜 중요한가?

[Q] 컴퓨터 HW와 네트워크 전송 속도 등이 빨라지면 (예: 양자 컴퓨터/통신) 효율적인 알고리즘의 중요성이 떨어지지 않는가?

- 처리/전송할 데이터도 계속해서 증가
- 점차 더 작은 HW 사용(예: 사물인터넷)할 필요 있으며, 이들은 한정된 처리 속도/메모리 가짐
- 양자 컴퓨터가 기존 컴퓨터가 풀 수 있는 모든 문제를 풀 수 있지는 않음

질문?

고급문제해결 과목 소개 & 운영 방법

01. 한 학기 동안 무엇을 어떻게 배우는가?

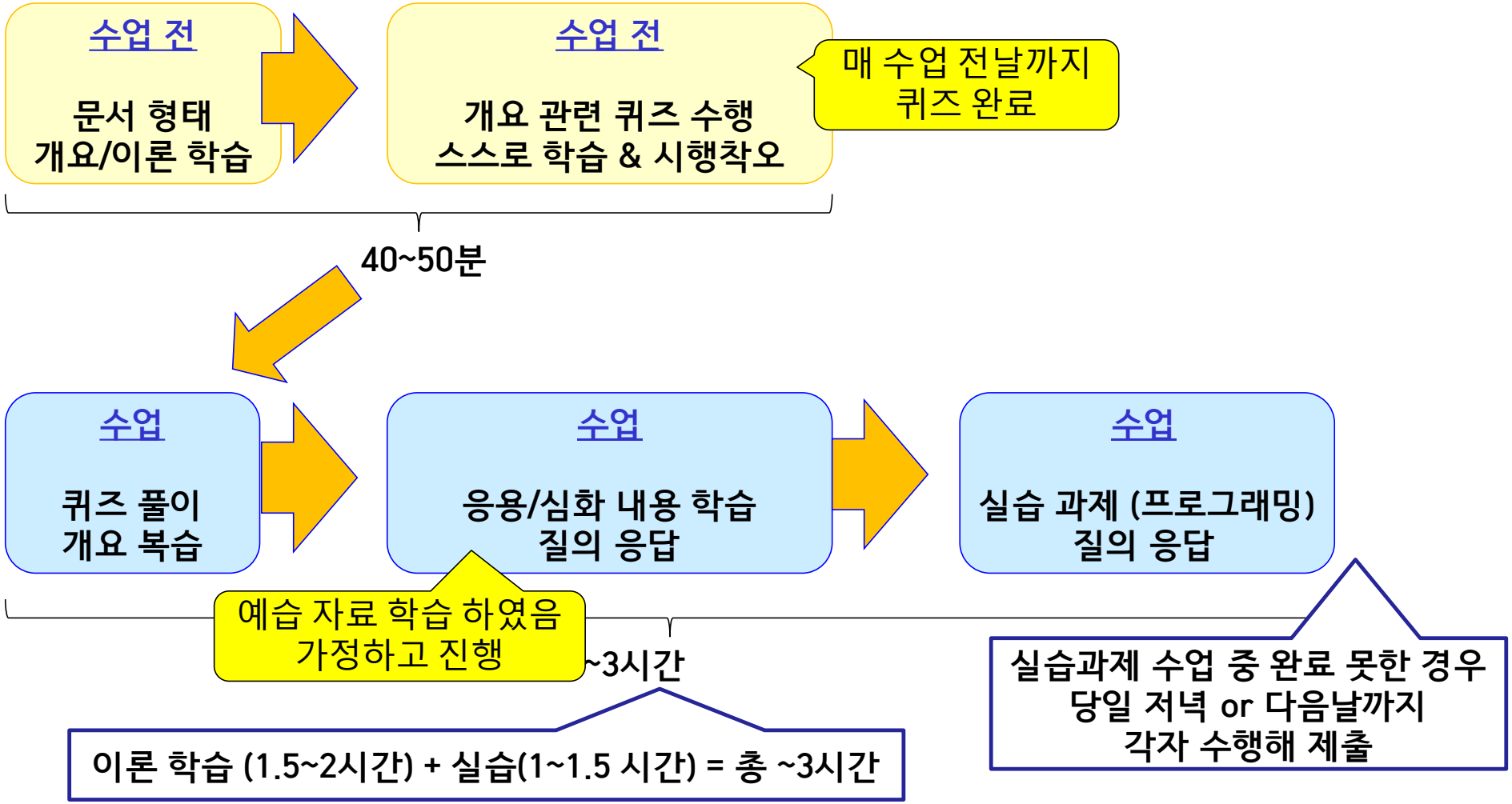
02. 수업 운영 방식

03. 평가 방법

04. 그 외 유의사항

05. 다음 수업 전까지 해야 할 일 정리

수업 전 예습 → 문제풀이/실습/질의응답 (플립 러닝, 거꾸로 학습)



‘거꾸로 학습(플립 러닝)’ 장점 및 중요한 점

- 이틀간 두 번 학습 → 기억에 남음
- 강의실 수업에서 응용/심화학습 가능
- 강의실 수업에서 응용/심화학습에 대한 질문 가능
- 개요 문서 활용해 반복학습 가능

- **반드시 개요를 미리 공부**하고 와야 강의실 수업에서 응용/심화문제 풀이 가능
  (그렇지 않으면 시간 낭비)
- 따라서 퀴즈를 풀고 오도록 함
- 퀴즈는 매주 3번 기회 있으며 최대 점수만 성적에 반영 (부족한 부분 다시 학습 가능)

퀴즈 응시 & 평가 방법

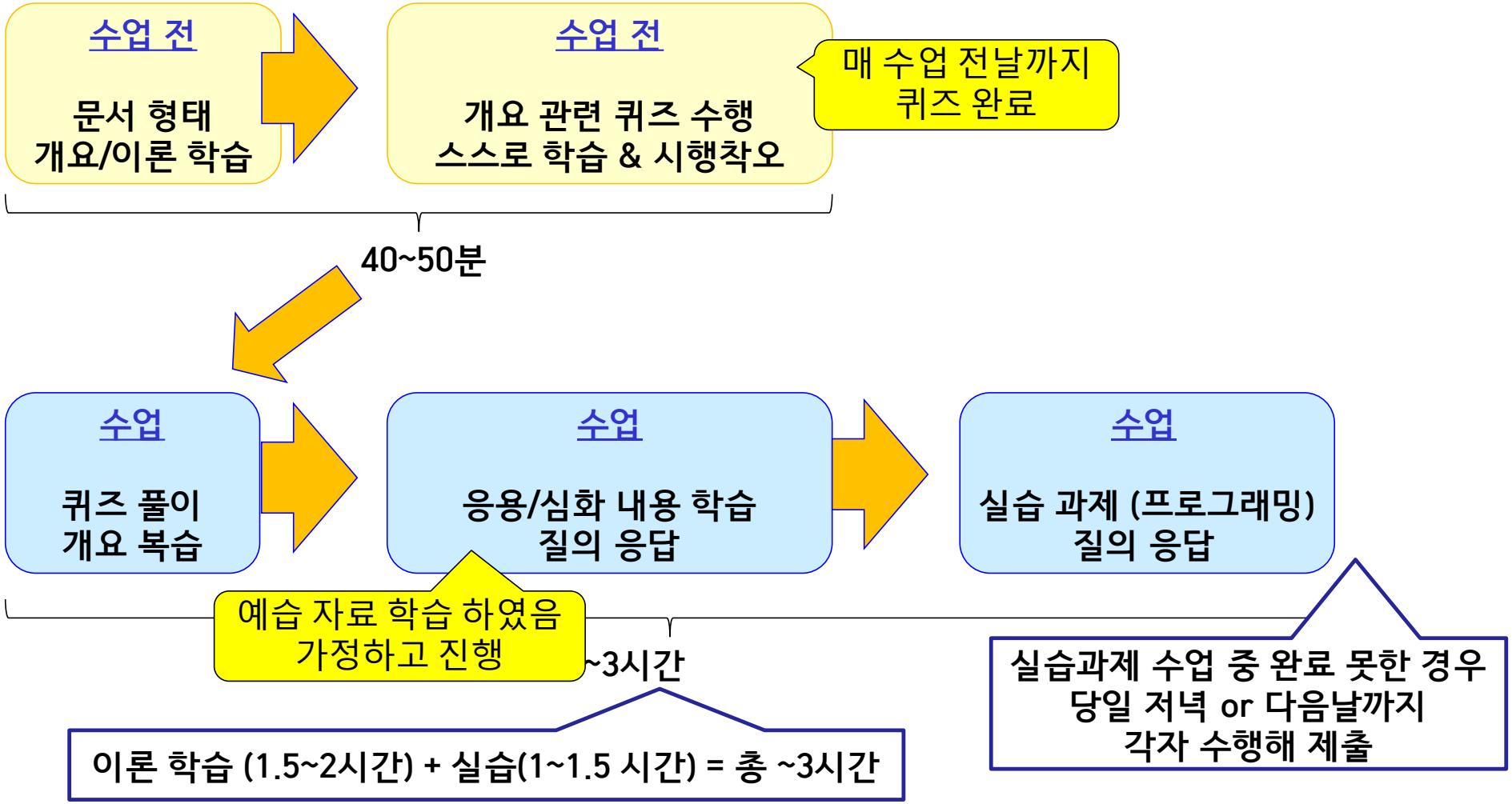
- 'LMS → 강의 콘텐츠 → 해당 주차 수업 → 퀴즈' 에서 응시
- 문서 공부 후 각자 퀴즈 응시
- **수업 전날까지 (11:59pm)** 완료

- 퀴즈 응시 후 바로 점수 확인 가능
- 3회까지 재 응시 가능
- 해당 주차 퀴즈 점수는 **3회 점수 중 최고점**으로

다음 수업 예습자료 & 퀴즈의 예

- 예습자료: PPT or HWP (그림 + 설명 있는 문서)
- 예습자료 학습 후 퀴즈 응시
- 3차까지 재 응시 가능. 만점인 경우 재 응시 필요 없음
- 첨부파일
  - 예습/퀴즈: 수업 전 학습 자료
  - 수업자료: 수업 영상에서 사용하는 자료
  - 코드: 예습자료 및 수업에서 사용하는 코드
  - 그 외 참고자료나 복습을 위한 추가 연습문제 있을 수 있음

예습에 소요되는 시간 + 강의실 수업 시간 = 총 수업 시간



질문?

고급문제해결 과목 소개 & 운영 방법

01. 한 학기 동안 무엇을 어떻게 배우는가?

02. 수업 운영 방식

03. 평가 방법

04. 그 외 유의사항

05. 다음 수업 전까지 해야 할 일 정리

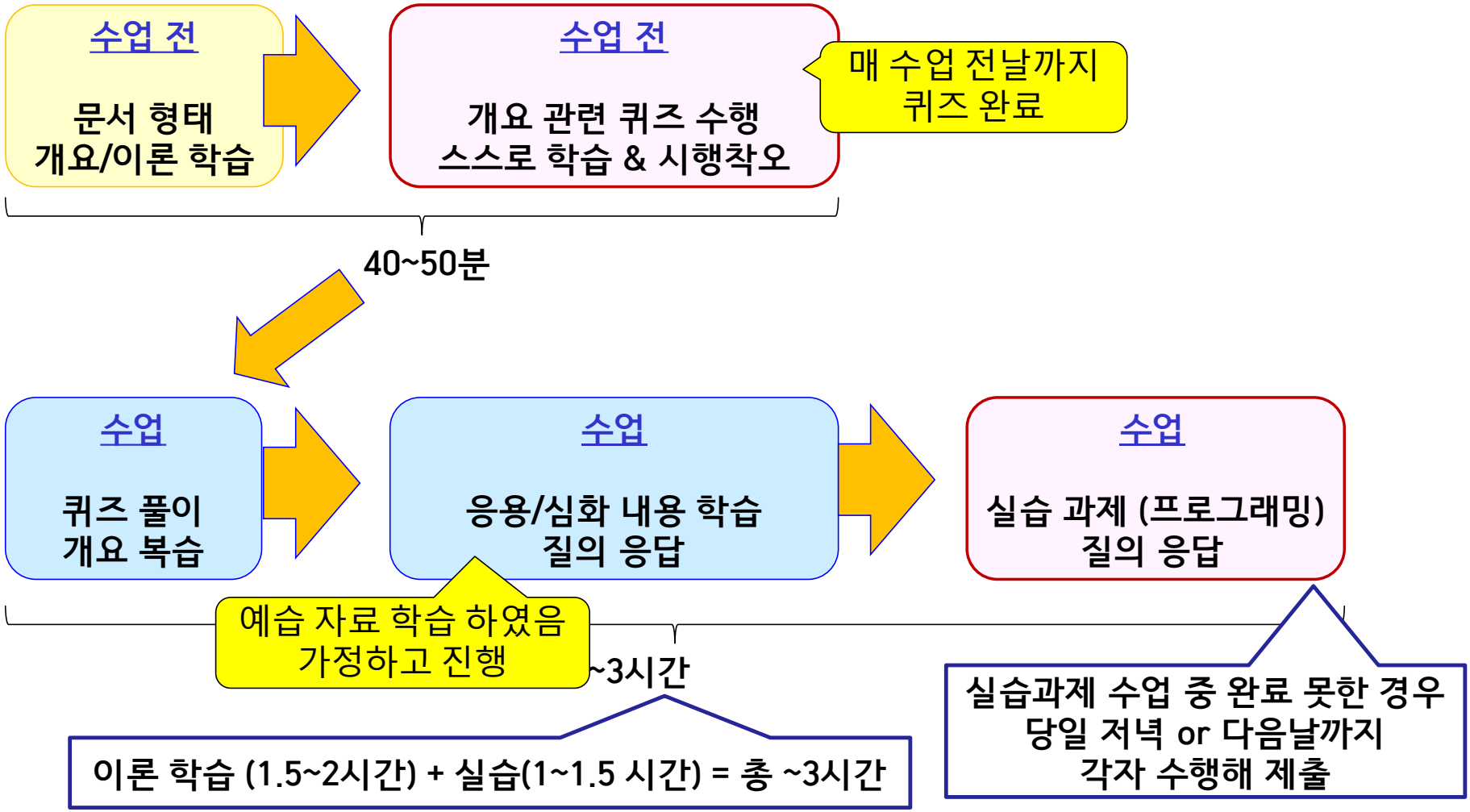
평가: 중간 = 기말, 과제중요

항목	가중치
중간고사 (중간고사 전에 배운 내용 평가, 대면예정)	30%
기말고사 (중간고사 이후에 배운 내용 평가, 대면예정)	30%
과제 (예습자료에 대한 Quiz + 실습 문제)	30%
출석+안전 교육 이수 여부	5+5%

	중간	기말	과제	출석/안전교육	총점
학생 1	81/100	80/100	72/100	15/15 + 수료	79.84
학생 2	69/100	69/100	99/100	15/15 + 수료	81.1

실습실 안전교육은 각자 기한 내 잘 완료하세요.
(기한은 학부사무실에 문의)

'과제 점수(30%)'에 해당하는 부분



수업에서 좋은 결과 내기 위한 도움말

- 매주 예습 & 퀴즈 잘 해오기 (예습 방법은 수업 운영 방식에서 설명)
- 시작 시간에 늦지 않게 오기: 이론 놓치면 실습하기 어려울 수 있음
- 필기할 준비

실습 과제 (프로그래밍) 유의사항

- Python 언어로 작성하고 제출 (오늘 해볼 예정)
- 실습 과제 제출 기한은 보통 다음날까지 (11:59pm)

- **개별 평가**이므로 코드는 꼭 각자 **직접 작성**해 주세요.
- 매주 제출한 코드에 유사도 검사를 합니다.

교재 - LMS 수업자료 활용하세요.

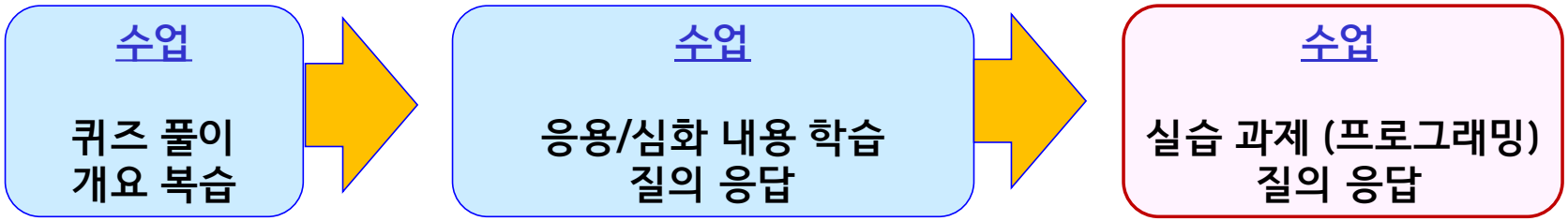
- LMS 강의자료: 수업자료, 예습자료, 퀴즈, 실습 문제

- LMS에 포스팅 되는 자료 보고 모르는 내용 질문하며 따라오는 것 가능
- 수업 내용을 잘 이해했다면 시험 문제 풀이 가능하므로 별도 교재를 읽지 않아도 됨
- 더 자세히 알고 싶은 경우 아래 참조하되 구매보다는 대여 추천

- 'Algorithms' by Robert Sedgewick and Kevin Wayne
- 'Introduction to Algorithms'
  by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein

질문, 면담

- 연구실: IT5 234호
- 하루 전까지 메일로 예약해주세요 -> sihyunglee@knu.ac.kr
- 질문은 가능하면 수업 중 or 실습 문제 풀이 시간을 적극 활용하세요.



질문?

고급문제해결 과목 소개 & 운영 방법

01. 한 학기 동안 무엇을 어떻게 배우는가?

02. 수업 운영 방식 (예습 자료 학습 → 퀴즈 풀이)

03. 평가 방법

04. 그 외 유의사항

05. 다음 수업 전까지 해야 할 일 정리

퀴즈 풀이 연습 (사전 지식 테스트, 기초 코드 읽기)

- 수업 자료의 예제 코드 및 실습 문제 풀이에 **필요한 기초 지식 정도를 알기 위함 + 퀴즈 풀이 방법 경험**
- 'LMS → 강의실 → 1주차 → 퀴즈' 에서 응시
- **평가에 포함되지 않음 (연습 퀴즈)**

실습 과제 풀이 연습

- **평가에 포함되지 않으며, 실습과제 채점 방식을 미리 알아서** 다음 시간 실습 과제 제출 시 실수를 방지하기 위함
- 다음 페이지의 문제 읽고 코드 작성한 후 각자 채점해 보기
- **평가에 포함되지 않음 (연습 과제)**

프로그램 구현 조건

- 정수 입력 n이 주어졌을 때, 1 이상의 정수로 이루어졌으며 합이 n 이하인 모든 수열 찾는 함수 구현

  def findAllSequence (**n**):

- 입력 **n**: ≥1 범위의 정수
  - 위 범위를 벗어나는 값은 입력으로 들어오지 않는다고 가정 (즉 오류 처리 하지 않아도 됨)

- 반환 값: 리스트
  - 리스트의 각 원소는 문제의 조건을 만족하는 각 수열을 담은 리스트임
  - 수열의 원소는 크기가 증가하는 순서여야 함 ($a_i \leq a_{i+1}$)
  - 모든 수열은 서로 달라야 하며 같은 수열을 두 번 이상 반환하면 안 됨
  - 서로 다른 두 수열 간의 순서: 가장 왼쪽 원소부터 차례로 비교할 때, 작은 원소를 가진 수열이 먼저 나와야 하며 (예: [1]이 [2] 이전에 나오고, [1,1]이 [1,2] 이전에 나옴), 원소가 같다면 길이가 짧은 수열이 먼저 나와야 함 (예: [1]이 [1,1] 이전에 나옴)

- 이번 시간에 제공한 코드 FindAllSequence.py에 위 함수 추가해 제출

입출력 예 (합이 n 이하인 모든 수열 찾되, 숫자 작은 수열 먼저 찾기)

```
print(findAllSequence(1))
```

```
<출력 결과>
[[1]]
```

```
print(findAllSequence(2))
```

```
<출력 결과>
[[1], [1, 1], [2]]
```

```
print(findAllSequence(3))
```

```
<출력 결과>
[[1], [1, 1], [1, 1, 1], [1, 2], [2], [3]]
```

```
print(findAllSequence(4))
```

```
<출력 결과>
[[1], [1, 1], [1, 1, 1], [1, 1, 1, 1], [1, 1, 2], [1, 2], [1, 3], [2], [2, 2], [3], [4]]
```

프로그램 구현 조건

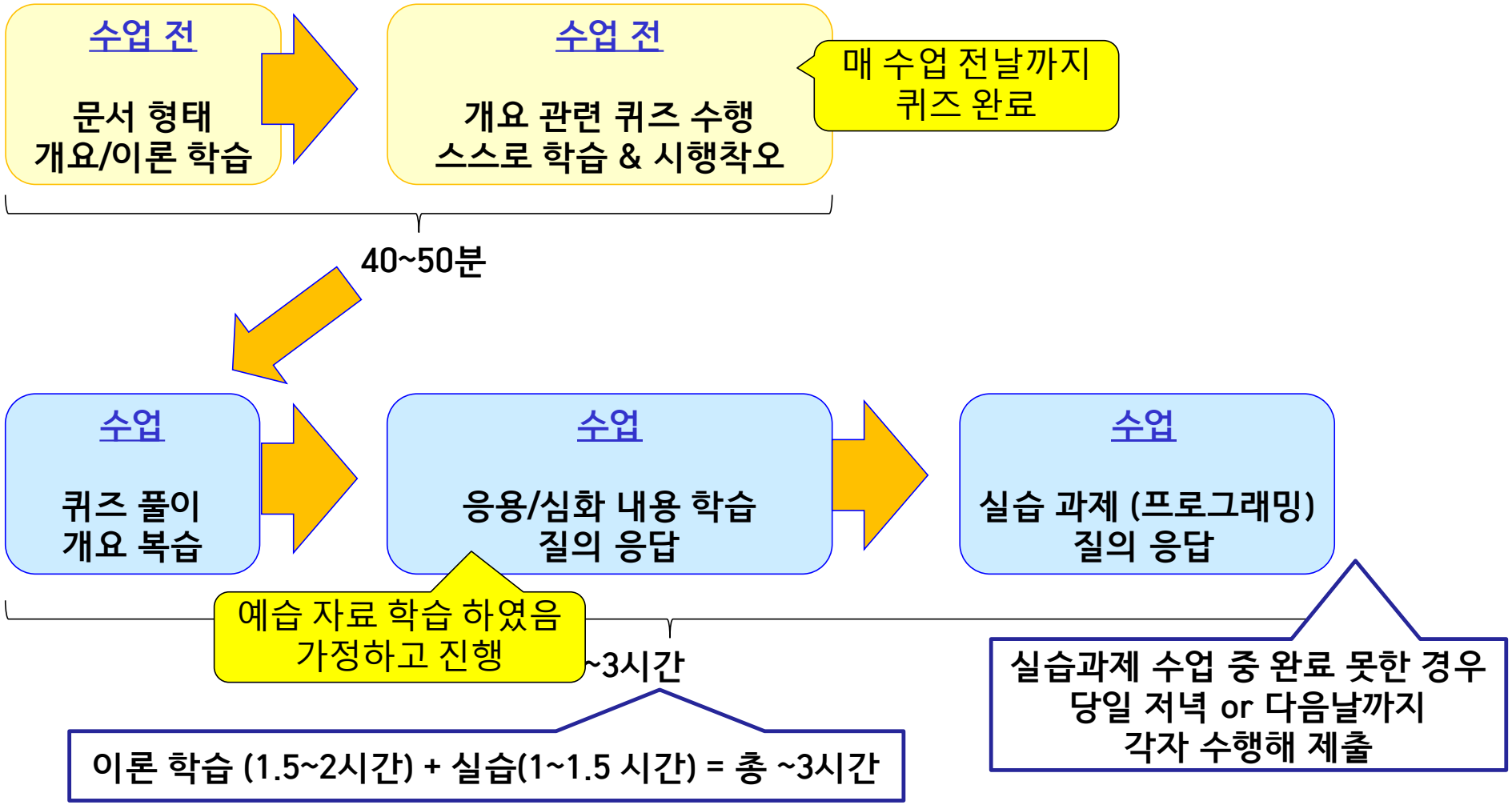
- 최종 결과물로 FindAllSequence.py 파일 하나만 제출하며, 이 파일만으로 코드가 동작해야 함

- import는 사용할 수 없음

- __main__ 아래의 코드는 작성한 함수가 올바름을 확인하는 코드로
- 코드 작성 후 스스로 채점해 보는데 활용하세요.
- 각 테스트 케이스에 대해 P(or Pass) 혹은 F(or Fail)이 출력됩니다.

- 코드를 제출할 때는 __main__ 아래 코드는 제거하거나 수정하지 말고 제출합니다. (채점에 사용되기 때문)

예습 & 퀴즈: 다음 수업 전날까지



스마트 출결 & 공지사항

- 예습 자료 학습 후 수업 전날 11:59pm 까지 Quiz 꼭 보세요.
- 생각하는 시간 중 필기할 필요가 많으므로 수업 전 강의 자료를 출력해 두거나 또는 필기할 수 있는 장비(iPad 등)에 담아 두세요.
- 출석: 첫 주는 수강 정정 기간이므로 모두 출석한 것으로 하겠습니다.