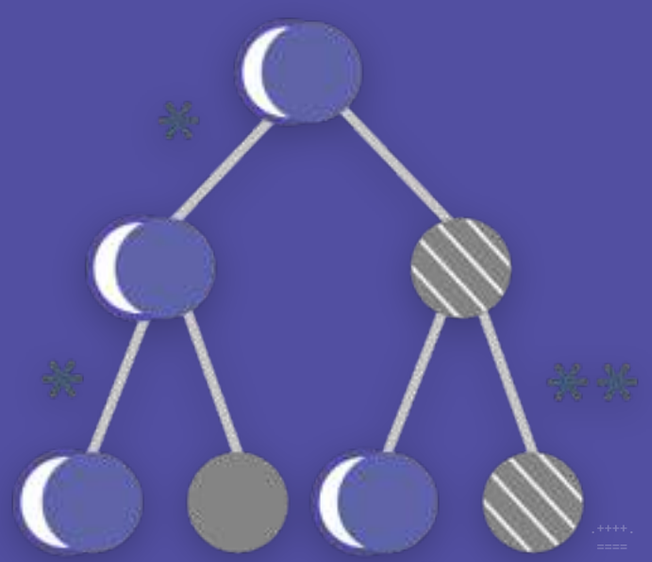
/* elice */

데이터구조 I

신현규 선생님 · 수 20:00

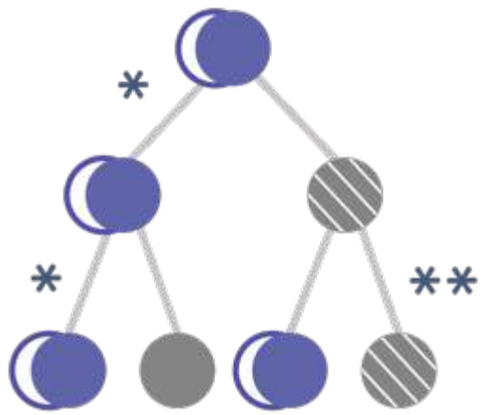


10월 11일 ~ 11월 7일

목차

01 교육과정 소개

데이터구조 I



- **교육기간** | 10월 11일 ~ 11월 07일 (4주)
- **선생님** | 신현규 선생님
- **라이브강의** | 매주 수요일 20:00 ~ 22:40

수강 대상

프로그래밍을 깊이 있게 배우고 싶은 초보자

졸업 및 취업을 앞둔 전산학과 학생

코드의 성능을 끌어올리고 싶은 개발자

수강 대상

프로그래밍을 깊이 있게 배우고 싶은 초보자

➡ 자료구조를 구현할 수 있는 프로그래밍 실력

졸업 및 취업을 앞둔 전산학과 학생

➡ 자료구조를 필요에 따라 디자인 할 수 있는 능력

코드의 성능을 끌어올리고 싶은 개발자

➡ 자료구조의 이용에 따른 성능 분석 능력

선생님 소개

신현규 선생님

- POSTECH 컴퓨터공학과 최우등 졸업
- 한국 정보올림피아드 경시대회 금상

- (전) 캘리포니아 오라클 연구소 소프트웨어 인턴
- (전) 대전창조경제혁신센터 데이터 사이언티스트 과정 총괄강사

조교 소개

주민건 조교

- 한양대학교 ERICA 컴퓨터 공학과
- (전) IM4U 정보영재 교육센터

주차별 커리큘럼

1주차 과정 소개, 배열, 연결리스트, 클래스

- 자료구조는 자료를 담는 주머니입니다. 배열, 연결 리스트의 개념과 장단점을 알아봅니다.

2주차 스택, 큐, 해싱

- 초급 자료구조와 자료를 저장·검색할 때 사용되는 해싱을 배워봅니다.

3주차 트리, 트리순회, 재귀호출

- 나무와 비슷하게 생긴 자료인 트리에 대해 배워보고 트리에서 자료를 탐색하는 알고리즘과 재귀호출을 배워봅니다.

4주차 재귀호출 응용 및 힙

- 재귀호출로 해결할 수 있는 문제를 알아보고 그 의미를 찾아봅니다. 힙에 대해 알아보고,
  이를 이용하여 문제를 해결합니다.

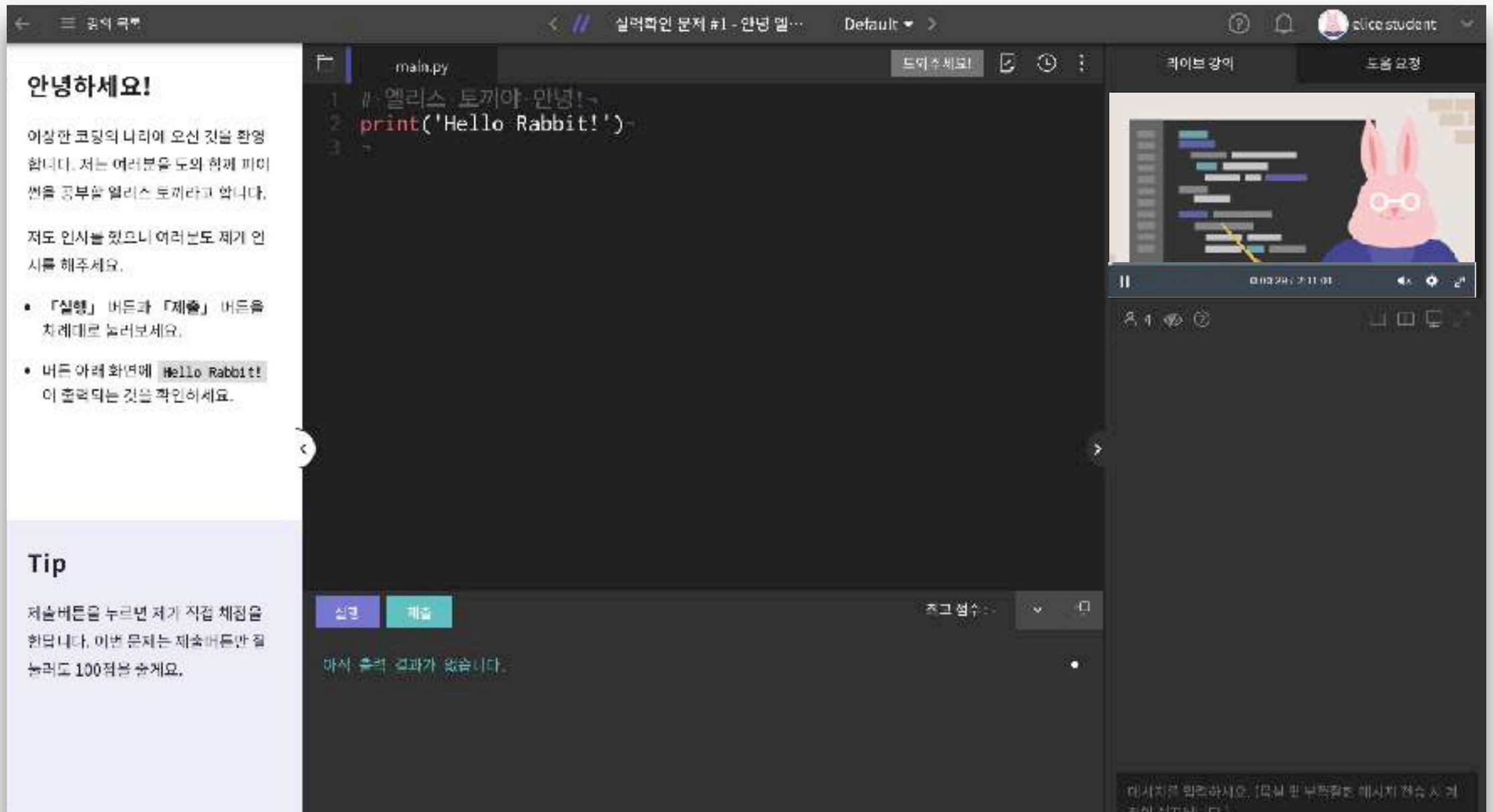
주차별 커리큘럼

5주차	시간복잡도
6주차	그래프 소개, DFS
7주차	그래프 심화, BFS
8주차	강의 요약, 알고리즘 과정 소개

02 엘리스 사용법

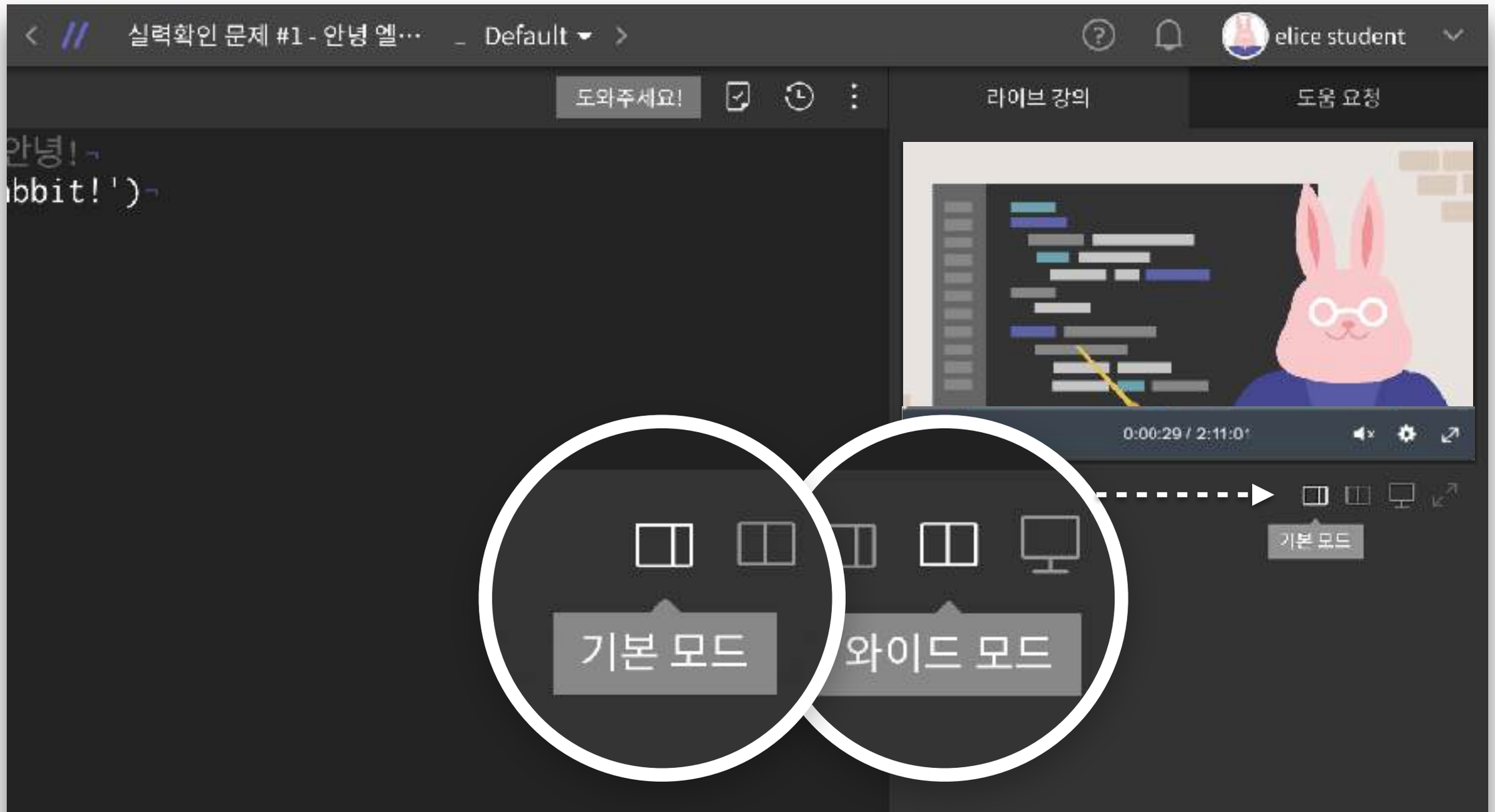
엘리스에서 코딩하기

모든 실습은 **엘리스**에서 진행됩니다.



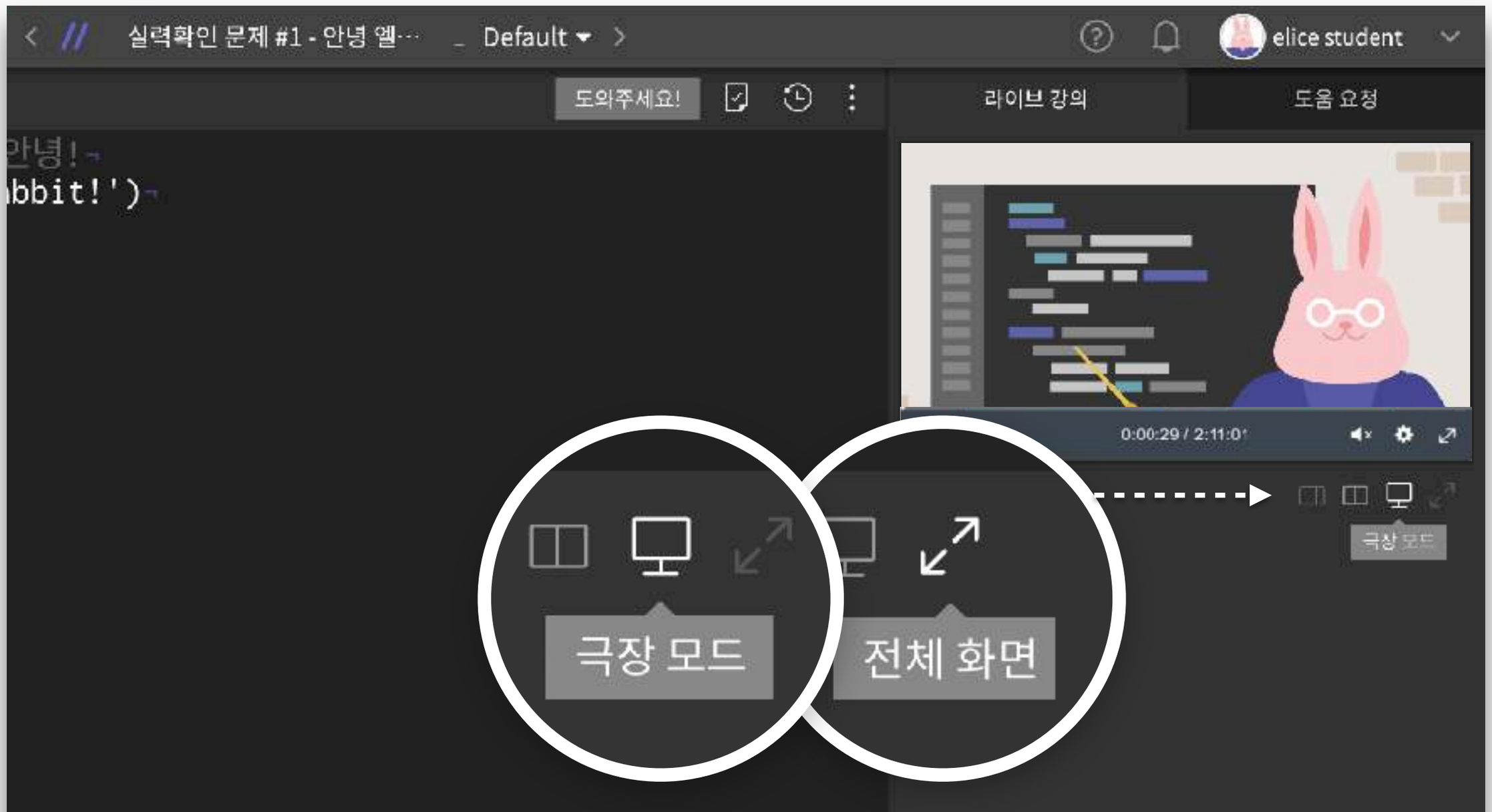
화면 레이아웃

코드를 작성하며 강의를 시청하고 싶을 때는
‘**기본 모드**’와 ‘**와이드 모드**’를 이용하세요.



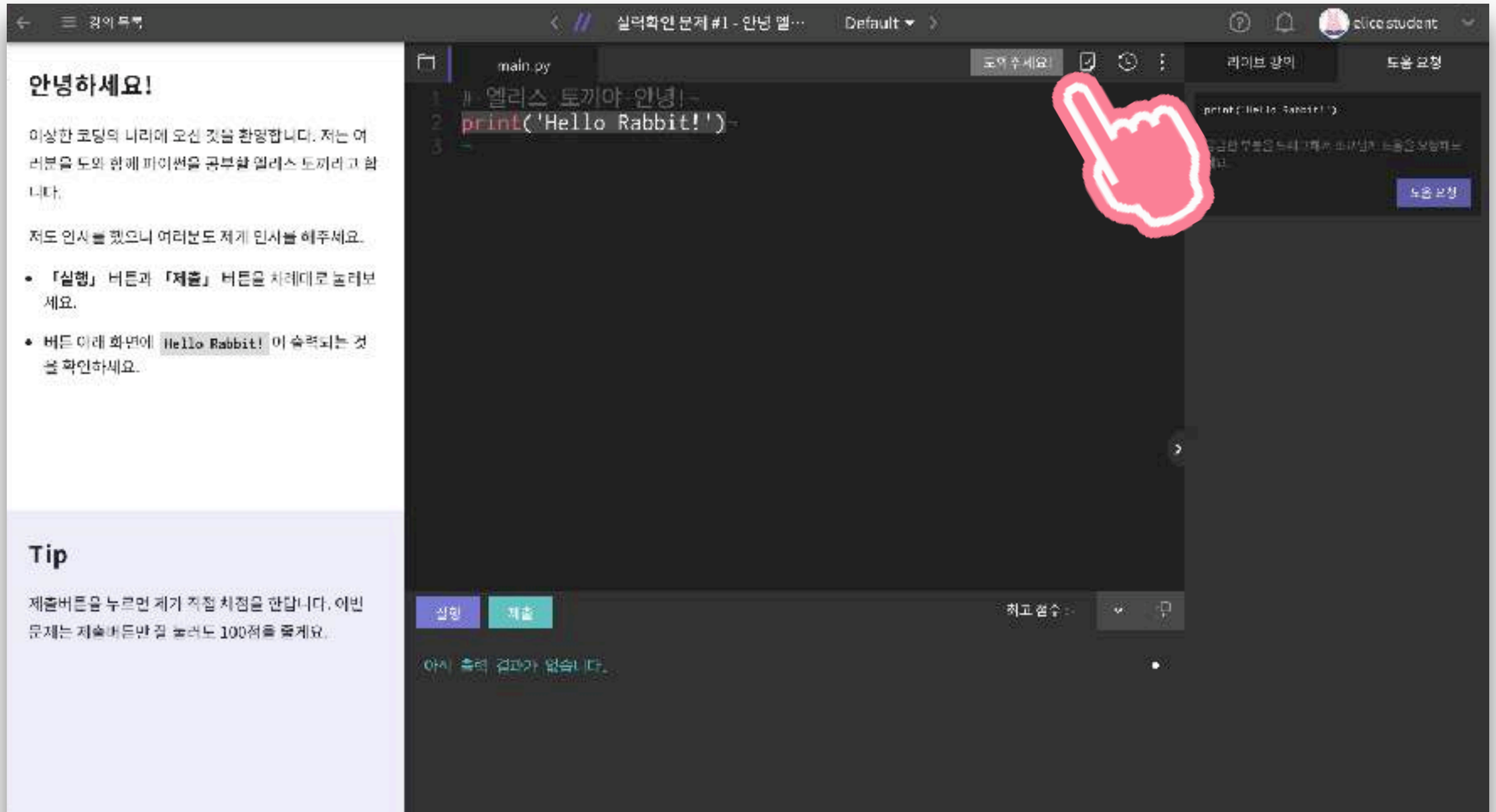
화면 레이아웃

큰 화면으로 강의를 시청하고 싶을 때는
'**극장 모드**'와 '**전체 화면**'을 이용하세요.



도움 요청

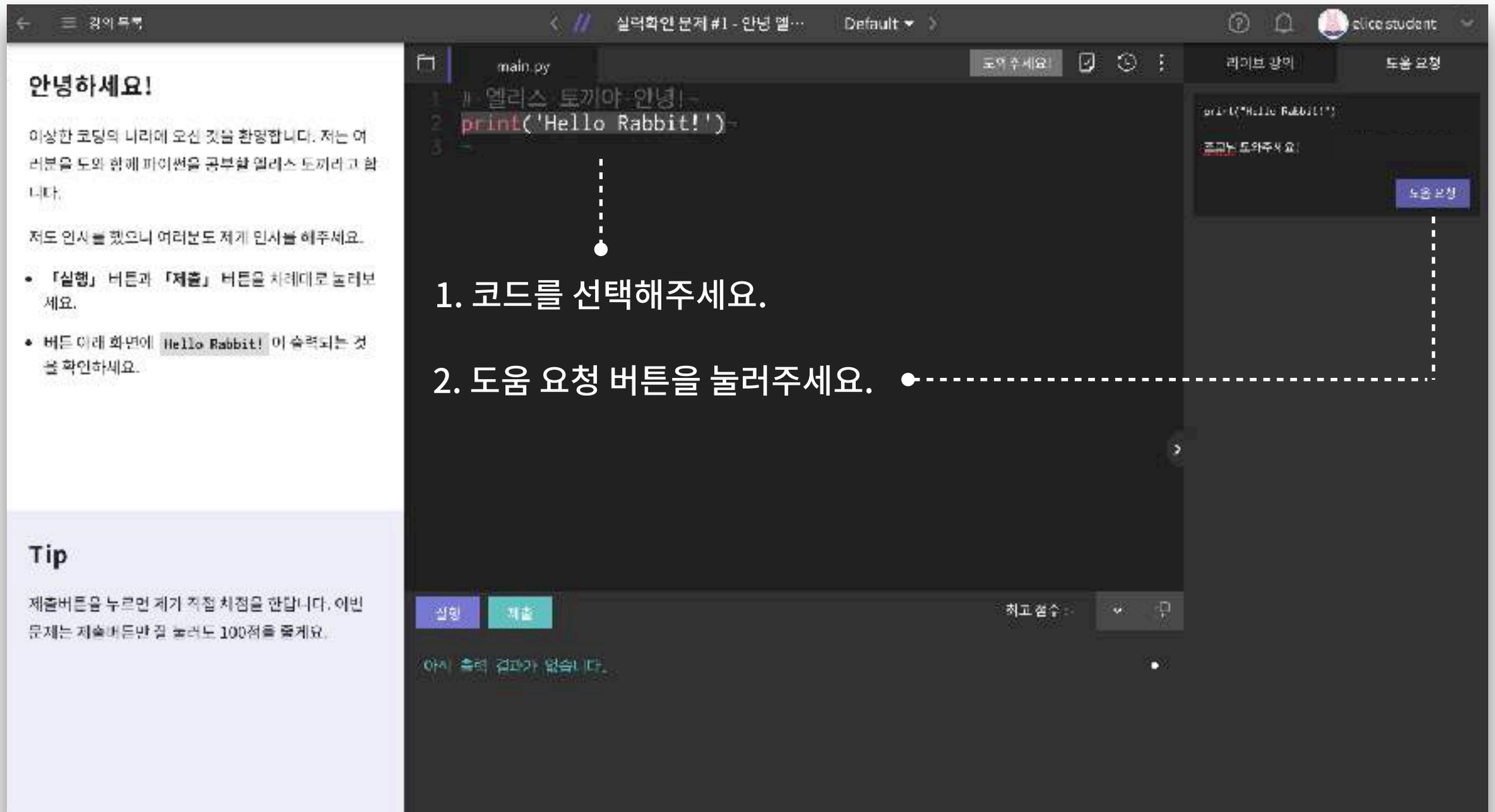
코드를 작성하는 도중 도움이 필요하시면
실습창 상단의 '**도와주세요!**'를 누르세요.



도움 요청

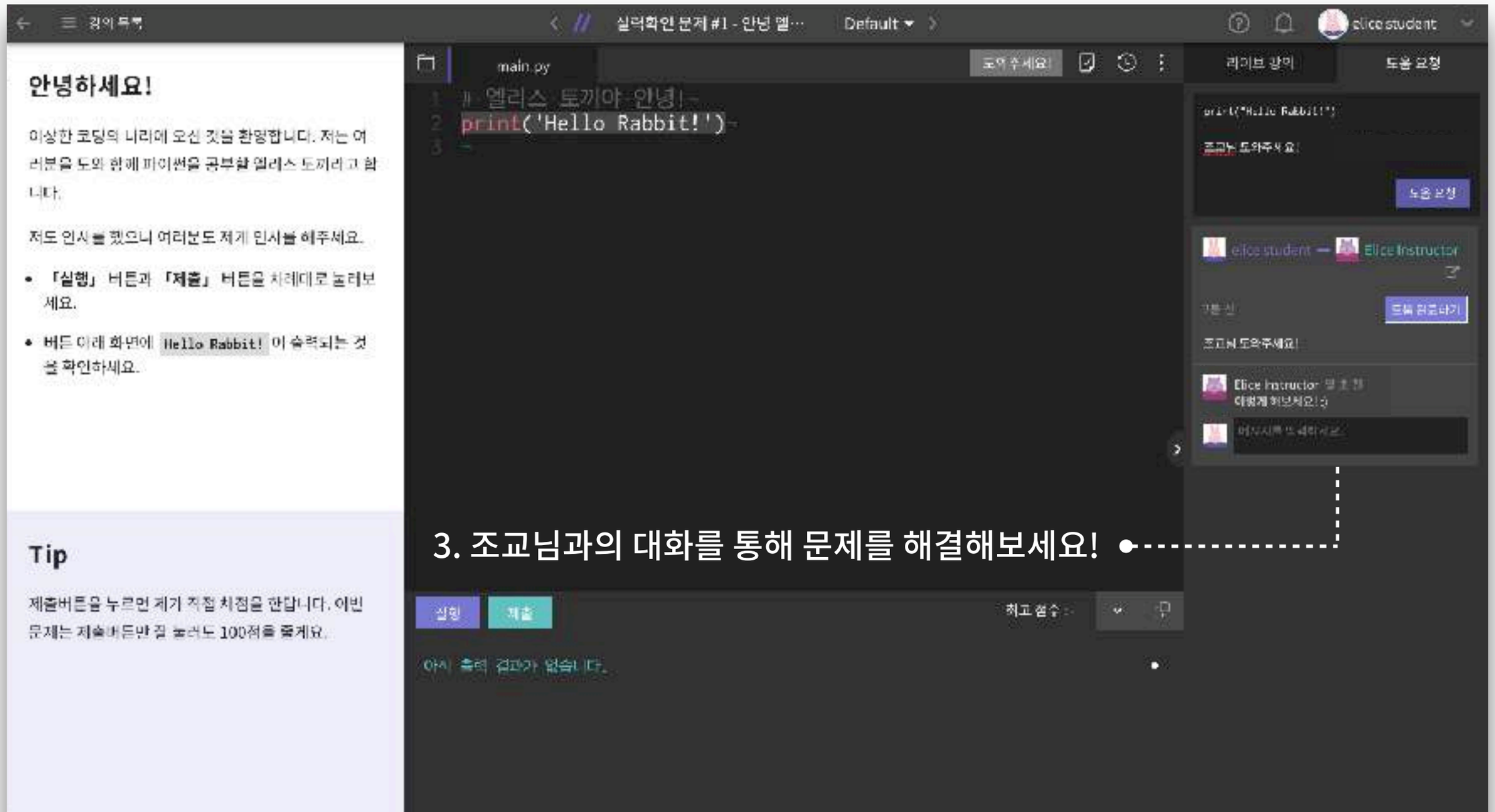
코드 선택과 궁금한 점 작성을 모두 완료 후

‘도움 요청’ 버튼을 누르세요.



도움 요청

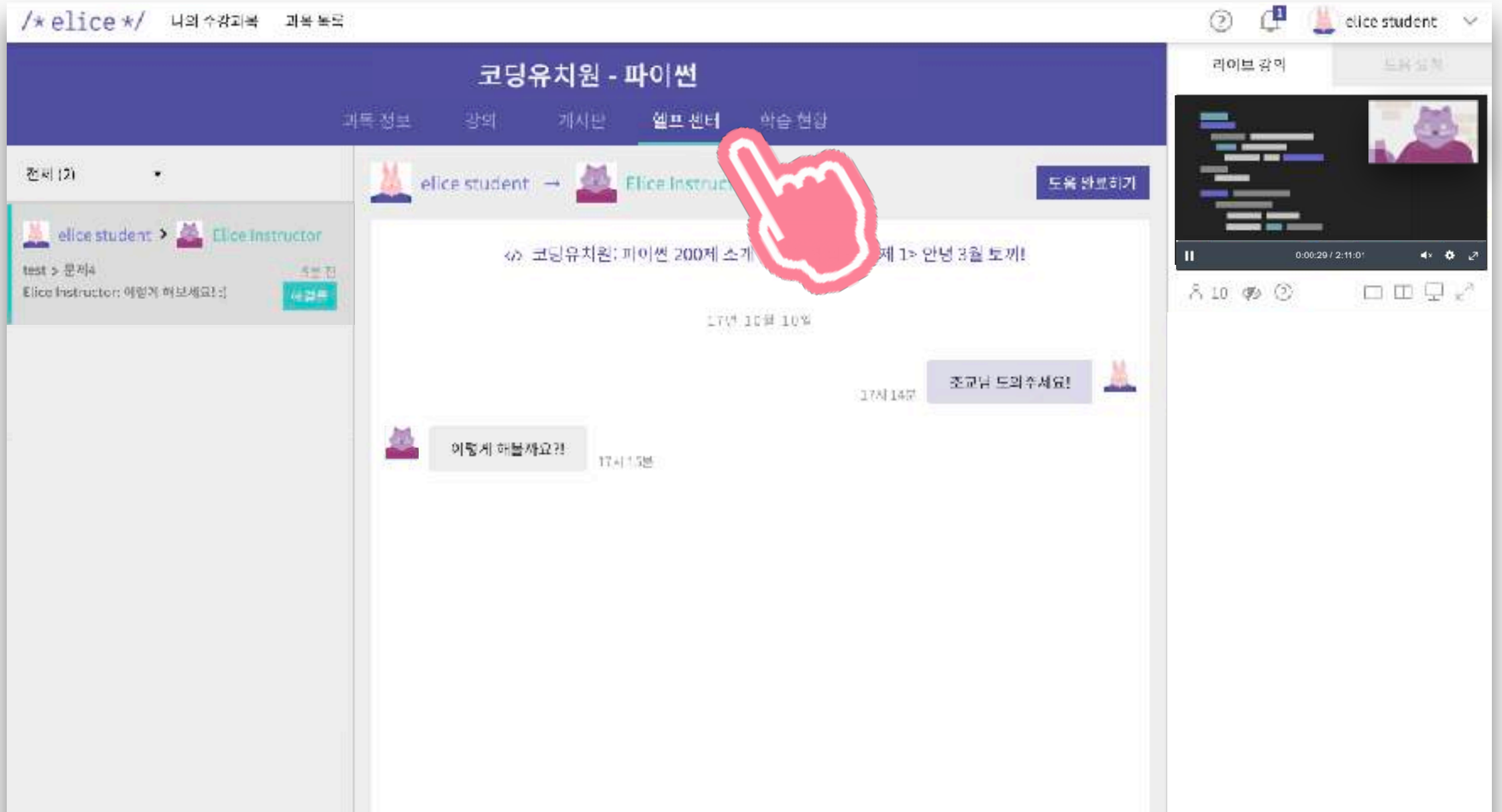
조교님과 대화를 나누고 싶을 때는
도움 요청 게시글의 **댓글을 이용**하세요.



헬프 센터

도움 요청 게시글은

헬프 센터에서 확인할 수 있습니다.



03 배열, 연결리스트, 클래스

컴퓨터 공학의 기본 커리큘럼

1. 프로그래밍 언어

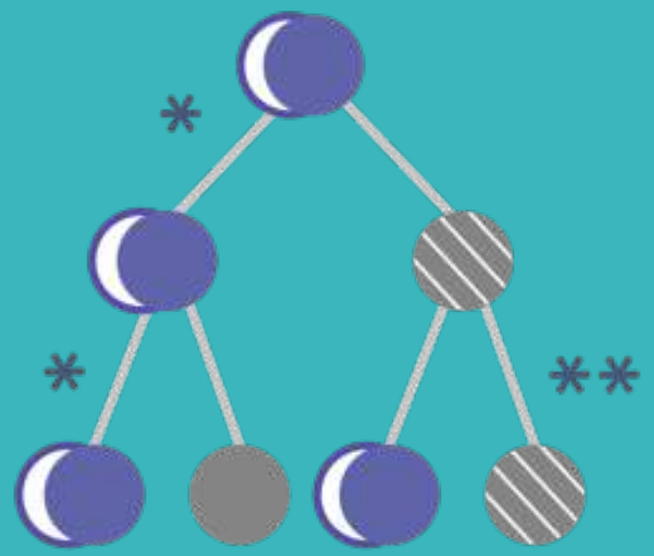
C / C++
Python
Matlab

2. 자료구조

Stack
Queue
Tree

3. 알고리즘

Brute-Force
Divide & Conquer
Dynamic Programming



데이터 구조 (자료구조)

= 데이터를 저장하는 구조

예제: 샴푸통 제작

예제: 샴푸통 제작

예제: 샴푸통 제작


PANTENE
PRO-V
DAILY
moisture
RENEWAL
DREAMcare
SHAMPOO


MOROCCANOIL

예제: 샴푸통 제작


PANTENE
PRO-V
DAILY
moisture
RENEWAL
DREAMcare
SHAMPOO


MOROCCANO L.


TEA
TREE
special
SHAMPOO
PAUL MITCHELL

예제: 샴푸통 제작


PANTENE
PRO-V
DAILY
moisture
RENEWAL
DREAMcare
SHAMPOO


MOROCCANOIL


TEA
TREE
special
SHAMPOO
PAUL MITCHELL


BOTTLE
plane
pus

예제: 샴푸통 제작

예제: 샴푸통 제작


PANTENE
PRO-V
DAILY
moisture
RENEWAL
DREAMcare
SHAMPOO


MOROCCANOIL


TEA
TREE
special
SHAMPOO
PAUL MITCHELL


BOTTLE
plays
pus

예제: 샴푸통 제작

샴푸 보관이 용이

샴푸를 짜는게 비교적 불편

샴푸를 짜는게 비교적 편함

여전히 좀 불편

좀 더 편함

많이 짜는데 시간이 좀 걸림

우주에서 사용하기 좋음

지구에서는 별로..

왜 데이터구조가 **중요한가?**

왜 데이터구조가 **중요한가?**

자료를 담긴 담아야 한다

왜 데이터구조가 **중요한가?**

자료를 담긴 담아야 한다

잘 담으면 좋으니까!

왜 데이터구조가 **중요한가?**

자료를 담긴 담아야 한다

잘 담으면 좋으니까!

왜 데이터구조가 **중요한가?**

자료를 담긴 담아야 한다

잘 담으면 좋으니까!

나의 목적에 맞게 데이터를 담는

그릇을 디자인 하자

변수 (Variable)

= 가장 기본적인 자료구조

```
x = 1
y = 'c'

print x
print y
```

리스트 (List)

= 변수의 나열

	0	1	2	3	4	5	6	7
myList	3	4	2	5	1	2	6	3

리스트 (List)

= 변수의 나열

	0	1	2	3	4	5	6	7
myList	3	4	2	5	1	2	6	3

장점

리스트 (List)

= 변수의 나열

	0	1	2	3	4	5	6	7
myList	3	4	2	5	1	2	6	3

장점 i 번째 원소를 바로 알 수 있다 (myList[i])

리스트 (List)

= 변수의 나열

	0	1	2	3	4	5	6	7
myList	3	4	2	5	1	2	6	3

단점

리스트 (List)

= 변수의 나열

	0	1	2	3	4	5	6	7
myList	3	4	2	5	1	2	6	3

단점 원소의 추가/ 삭제가 까다롭다

리스트 (List)

= 변수의 나열

	0	1	2	3	4	5	6	7
myList	3	4	2	5	1	2	6	3

4번째에 숫자 10을 추가하자

단점 원소의 추가/ 삭제가 까다롭다

리스트 (List)

= 변수의 나열

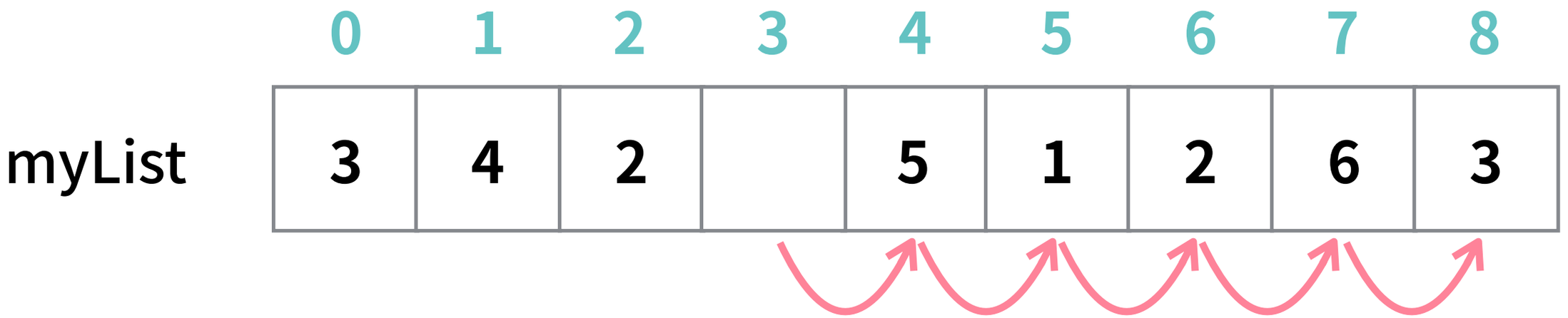
	0	1	2	3	4	5	6	7	8
myList	3	4	2	5	1	2	6	3	

4번째에 숫자 10을 추가하자

단점 원소의 추가/ 삭제가 까다롭다

리스트 (List)

= 변수의 나열



단점 원소의 추가/ 삭제가 까다롭다

리스트 (List)

= 변수의 나열

	0	1	2	3	4	5	6	7	8
myList	3	4	2	10	5	1	2	6	3

4번째에 숫자 10을 추가하자

단점 원소의 추가/ 삭제가 까다롭다

리스트 (List)

= 변수의 나열

	0	1	2	3	4	5	6	7	8
myList	3	4	2	10	5	1	2	6	3

5번째 숫자를 제거하자

단점 원소의 추가/ 삭제가 까다롭다

리스트 (List)

= 변수의 나열

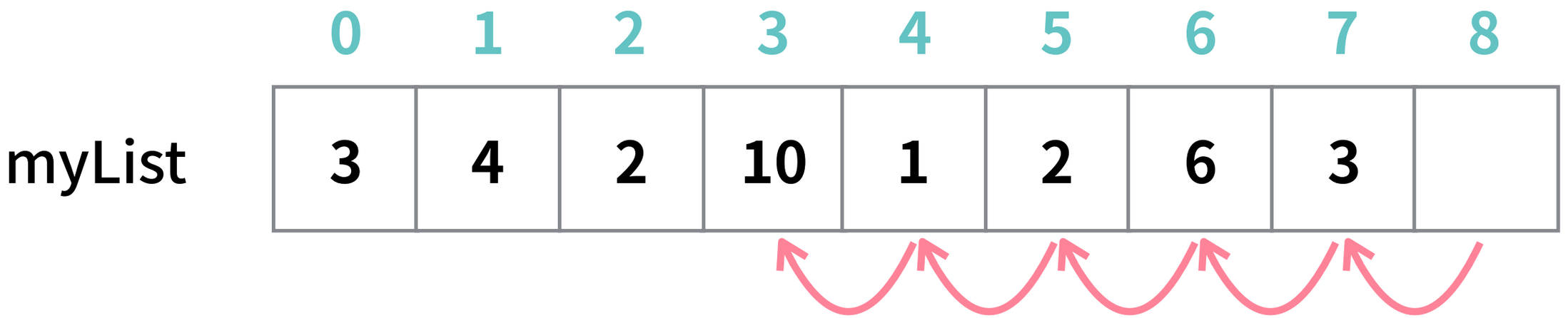
	0	1	2	3	4	5	6	7	8
myList	3	4	2	10		1	2	6	3

5번째 숫자를 제거하자

단점 원소의 추가/ 삭제가 까다롭다

리스트 (List)

= 변수의 나열



단점 원소의 추가/ 삭제가 까다롭다

리스트 (List)

= 변수의 나열

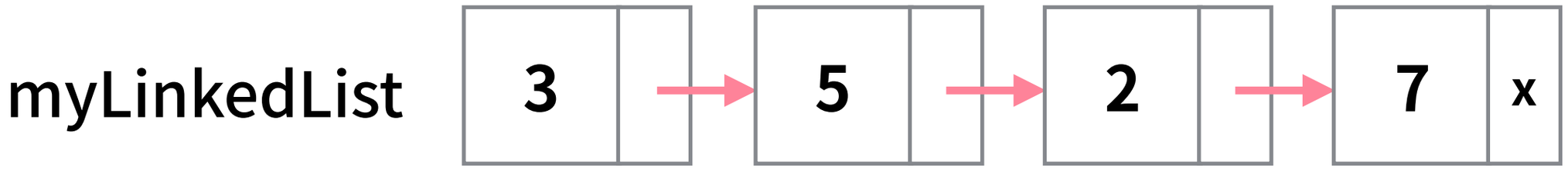
	0	1	2	3	4	5	6	7
myList	3	4	2	10	1	2	6	3

5번째 숫자를 제거하자

단점 원소의 추가/ 삭제가 까다롭다

링크드 리스트 (Linked List)

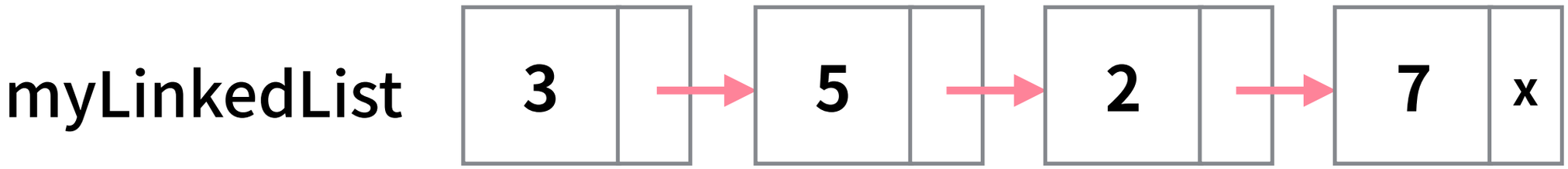
= 여러 개의 변수를 저장하는 다른 방법



링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

변수추가

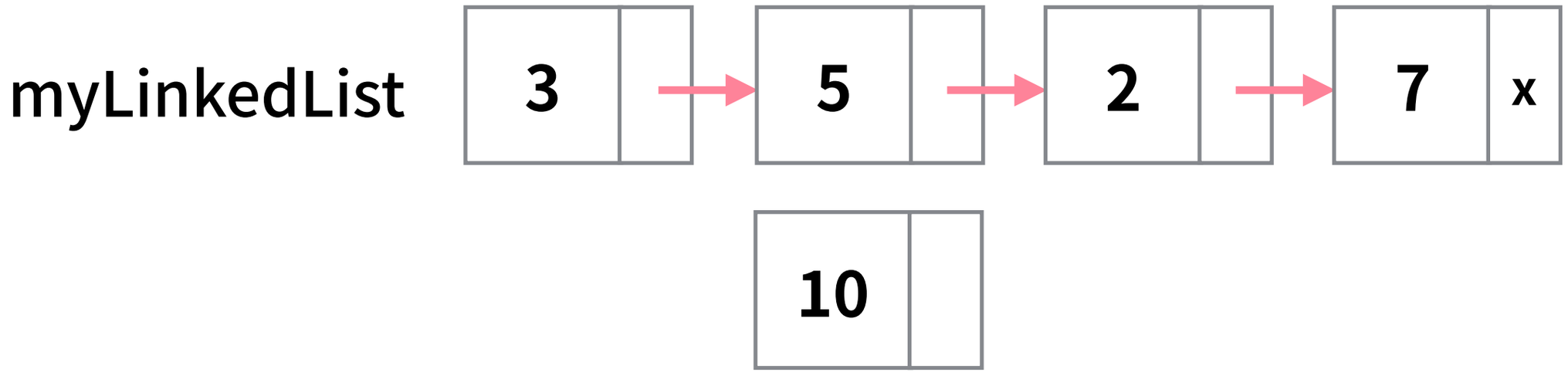


3번째에 숫자 10을 추가하자

링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

변수추가

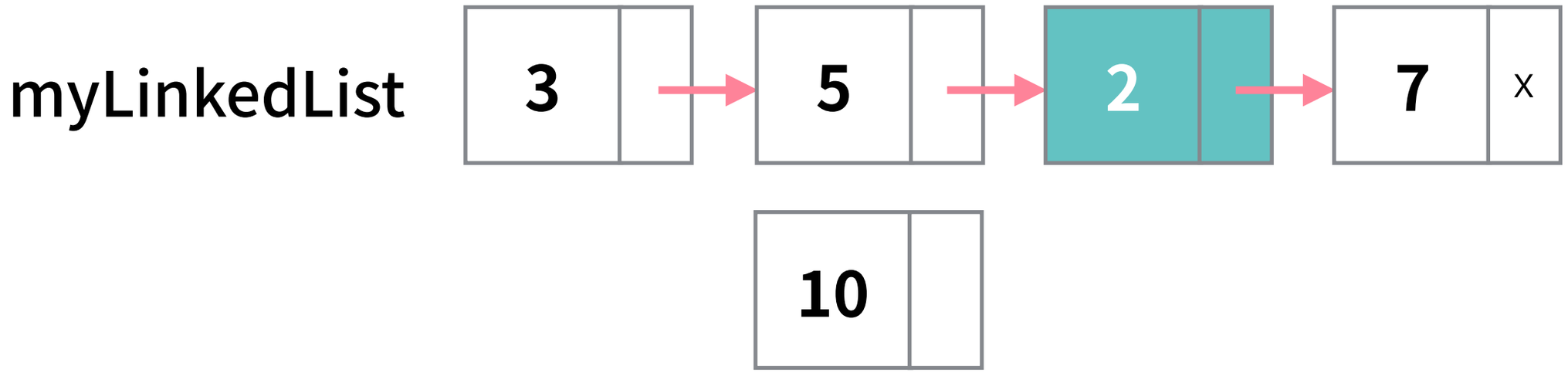


3번째에 숫자 10을 추가하자

링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

변수추가

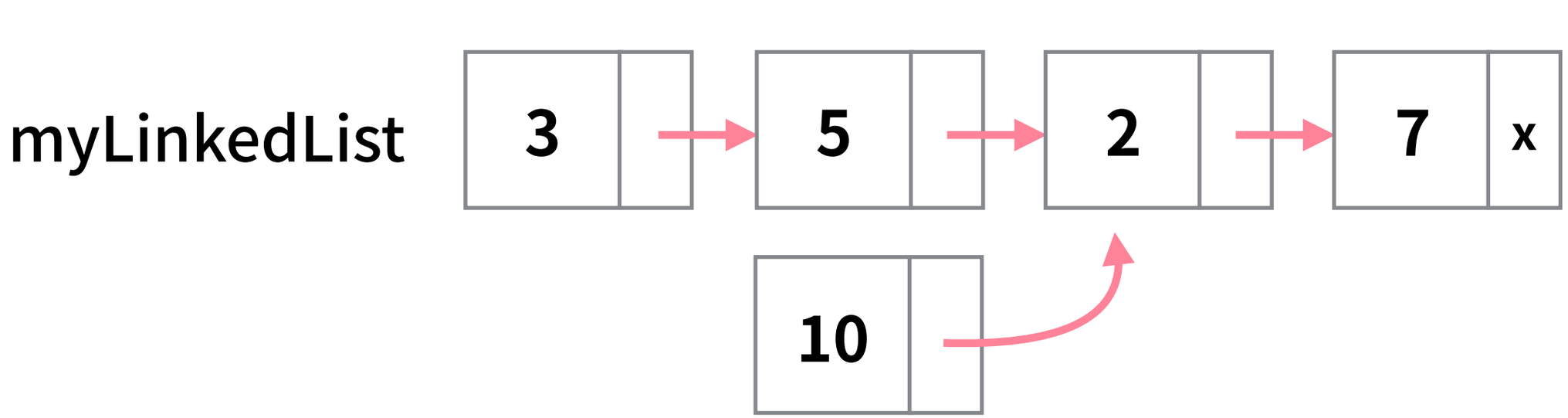


3번째에 숫자 10을 추가하자

링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

변수추가

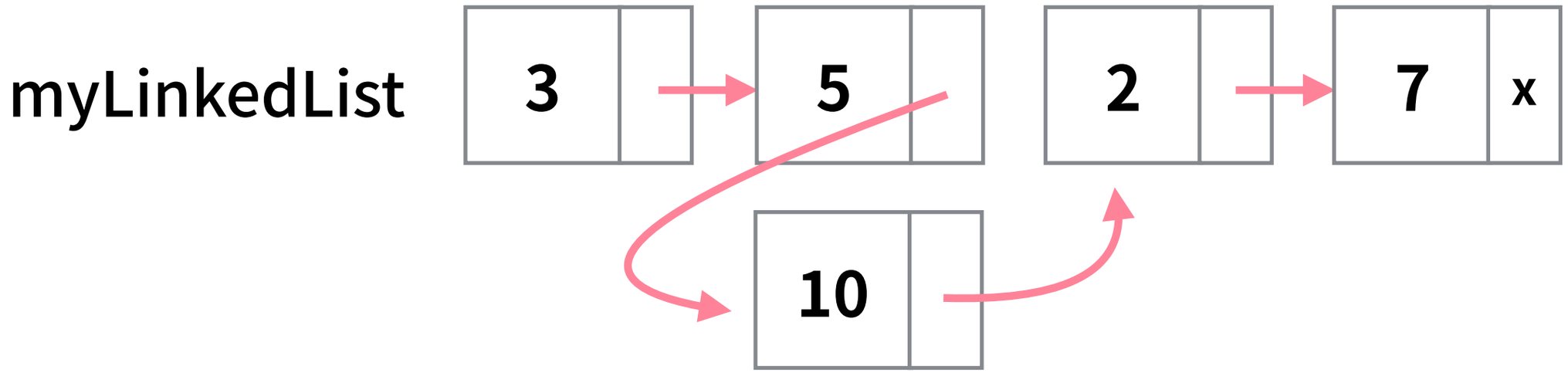


3번째에 숫자 10을 추가하자

링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

변수추가

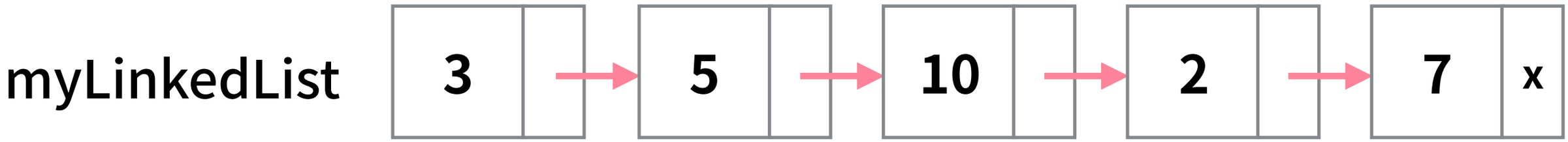


3번째에 숫자 10을 추가하자

링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

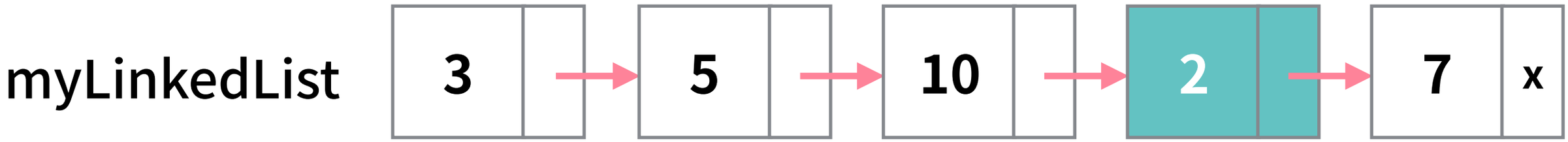
변수추가



링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

변수제거

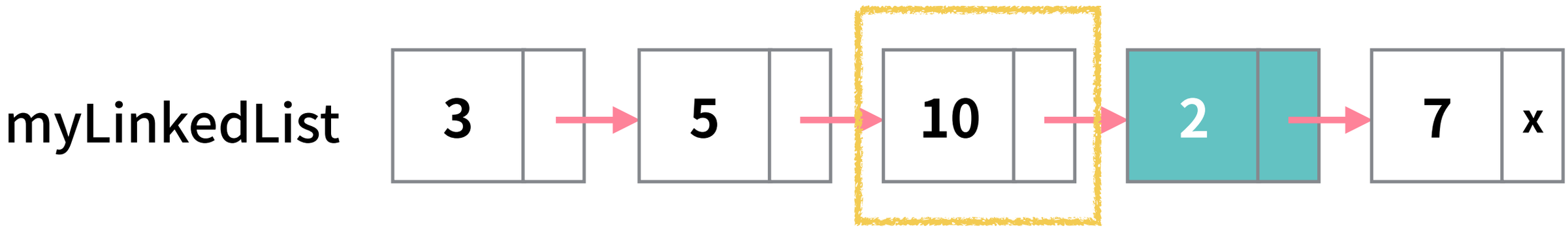


4번째에 숫자 제거하자

링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

변수제거

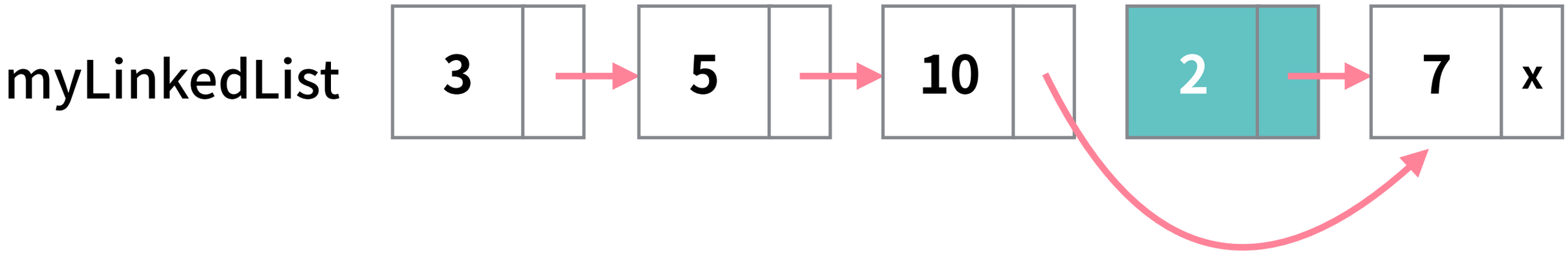


4번째에 숫자 제거하자

링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

변수제거

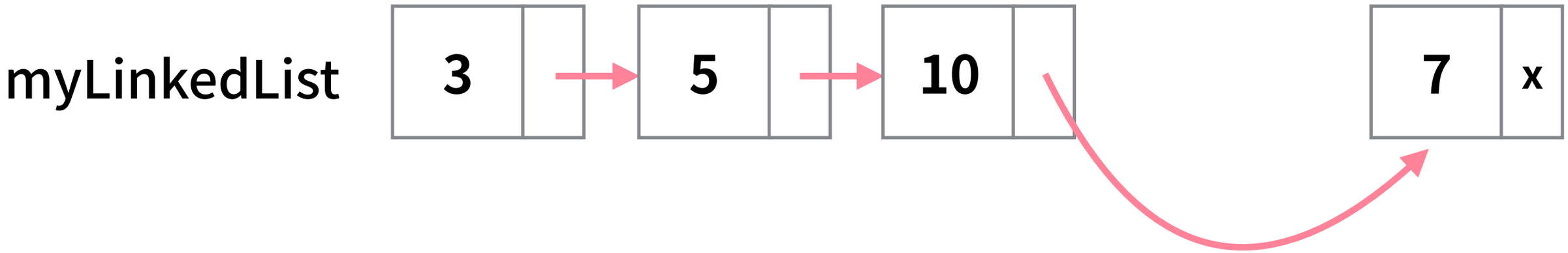


4번째에 숫자 제거하자

링크드 리스트 (Linked List)

= 여러 개의 변수를 저장하는 다른 방법

변수제거

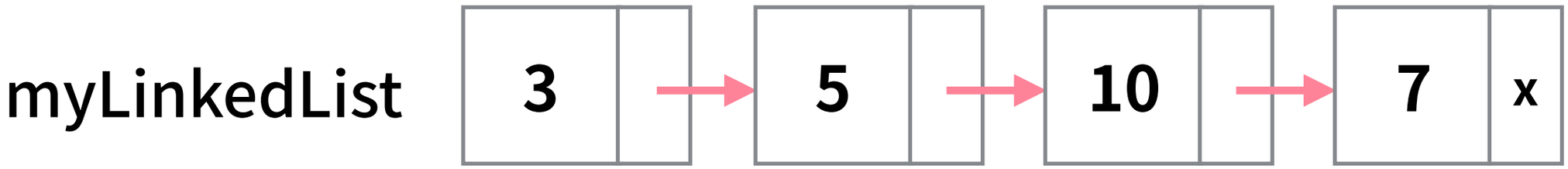


4번째에 숫자 제거하자

링크드 리스트 (Linked List)

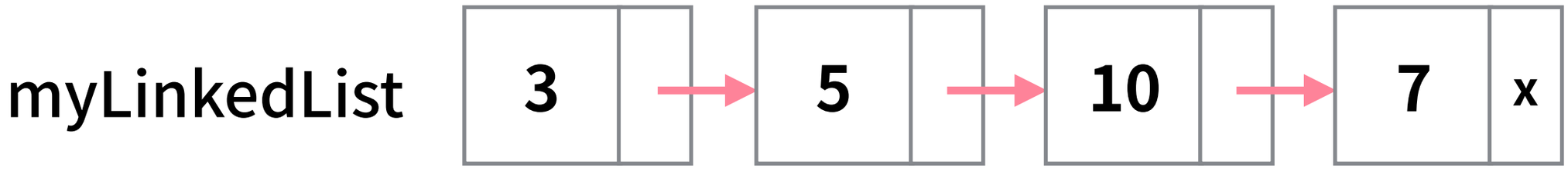
= 여러 개의 변수를 저장하는 다른 방법

변수제거



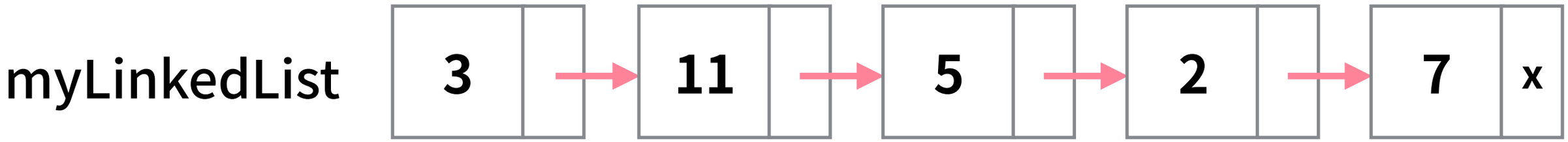
링크드 리스트 (Linked List)

예제) 2번째에 숫자 11을 추가하세요.



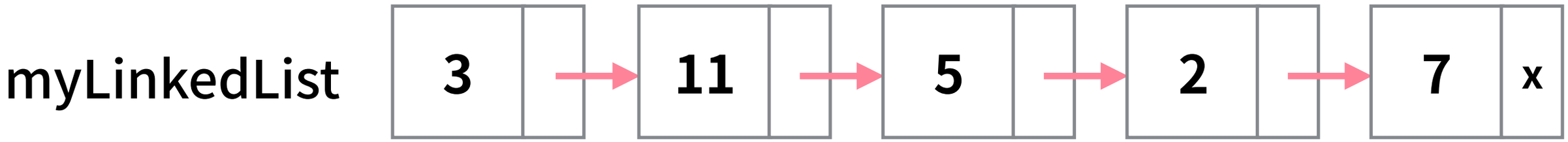
링크드 리스트 (Linked List)

예제) 2번째에 숫자 11을 추가하세요.



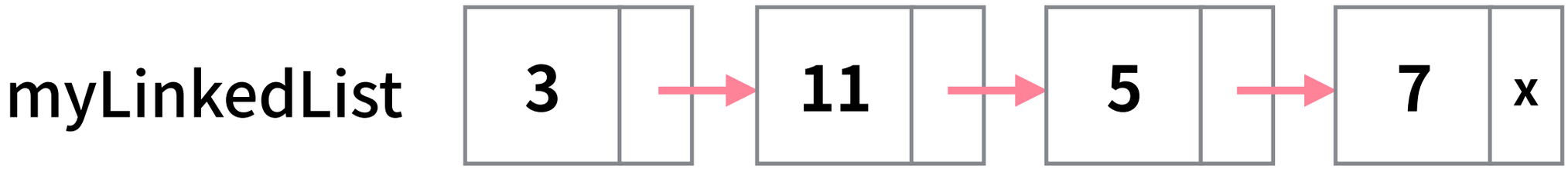
링크드 리스트 (Linked List)

예제) 4번째 숫자를 제거하세요.



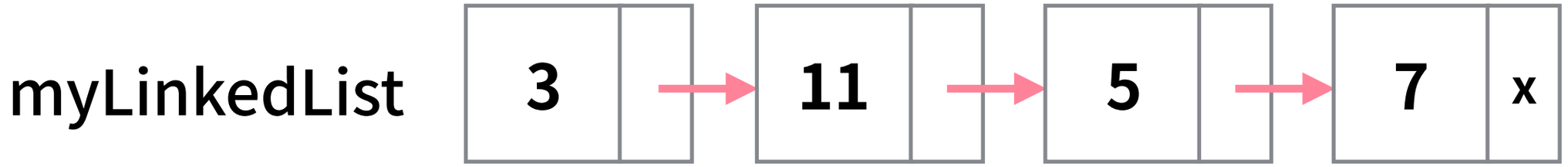
링크드 리스트 (Linked List)

예제) 4번째 숫자를 제거하세요.



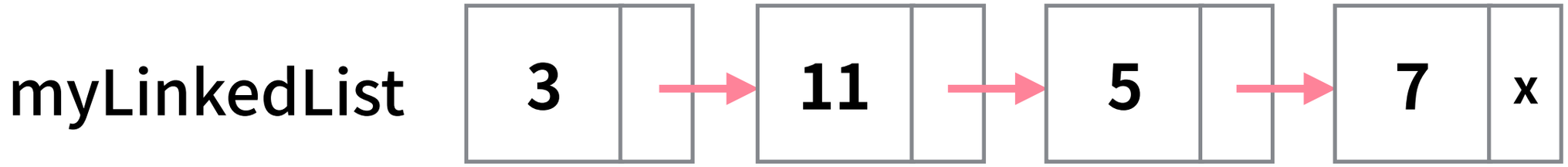
링크드 리스트 (Linked List)

장점 삽입/ 삭제가 빠르다



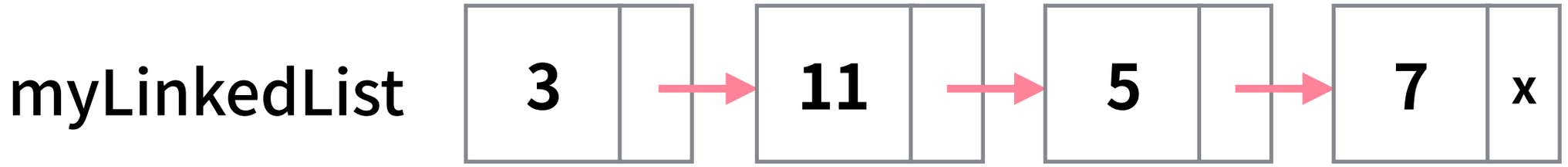
링크드 리스트 (Linked List)

단점 i 번째 원소를 알기가 쉽지 않다



링크드 리스트 (Linked List)

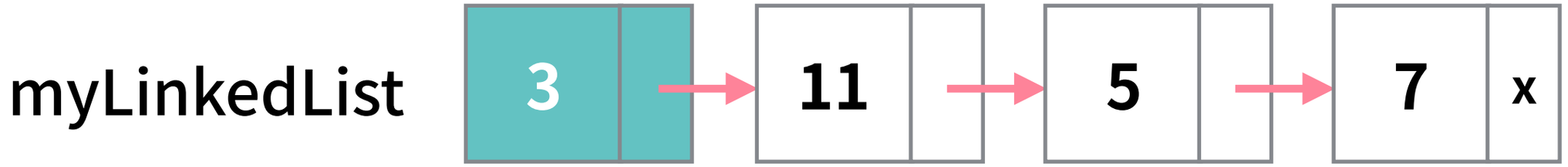
단점 i 번째 원소를 알기가 쉽지 않다



3번째 원소는?

링크드 리스트 (Linked List)

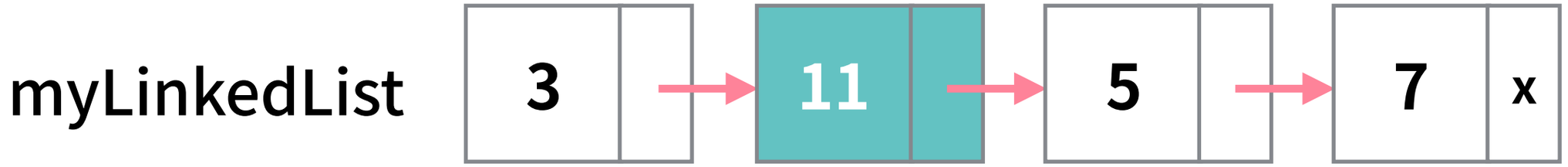
단점 i 번째 원소를 알기가 쉽지 않다



3번째 원소는?

링크드 리스트 (Linked List)

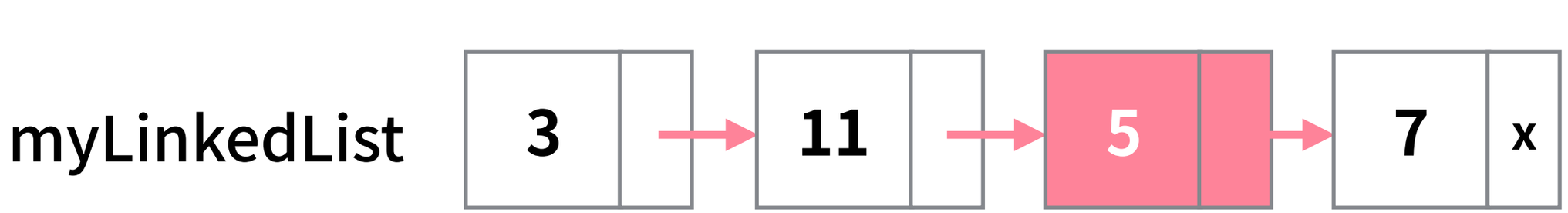
단점 i 번째 원소를 알기가 쉽지 않다



3번째 원소는?

링크드 리스트 (Linked List)

단점 i 번째 원소를 알기가 쉽지 않다



3번째 원소는?

요약

리스트 (List)

장점: i 번째 원소를 바로 알 수 있다

단점: 원소의 추가 / 삭제가 까다롭다

링크드 리스트 (Linked List)

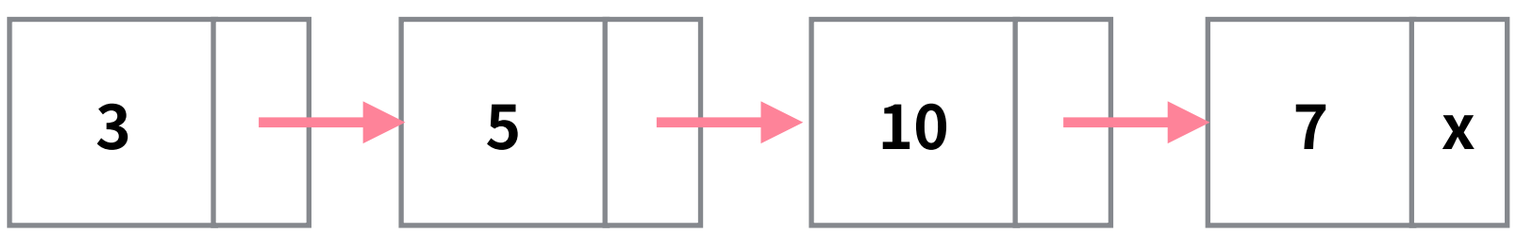
장점: 원소의 추가 / 삭제가 간단하다

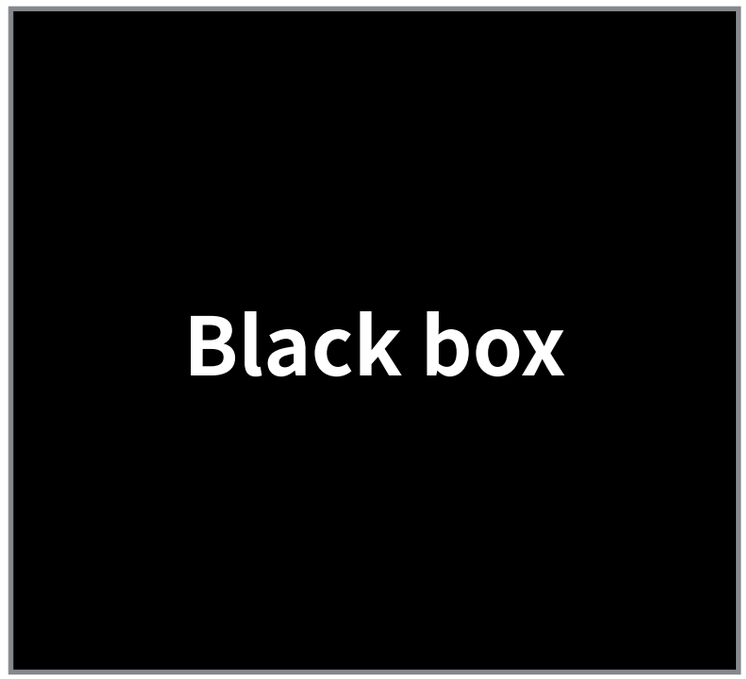
캡슐화: 자료구조 구현의 핵심

캡슐화 (Encapsulation)

자료구조를 사용하는 사람은

자료구조가 어떻게 동작하는지 알 필요가 없다

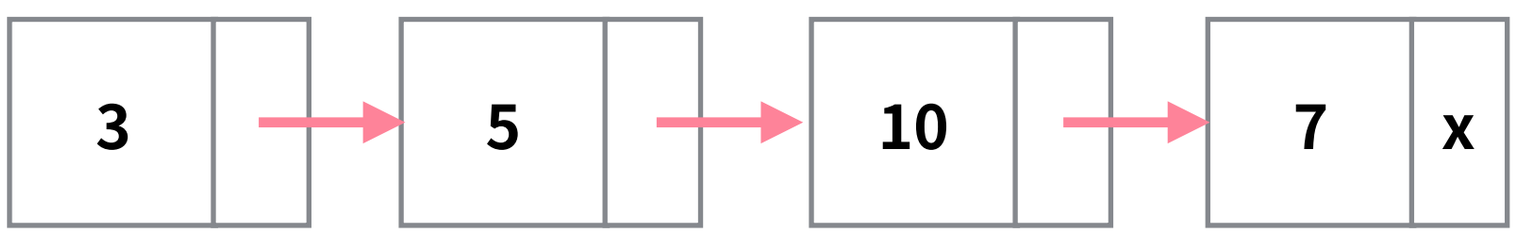




캡슐화 (Encapsulation)

자료구조를 사용하는 사람은

자료구조가 어떻게 동작하는지 알 필요가 없다



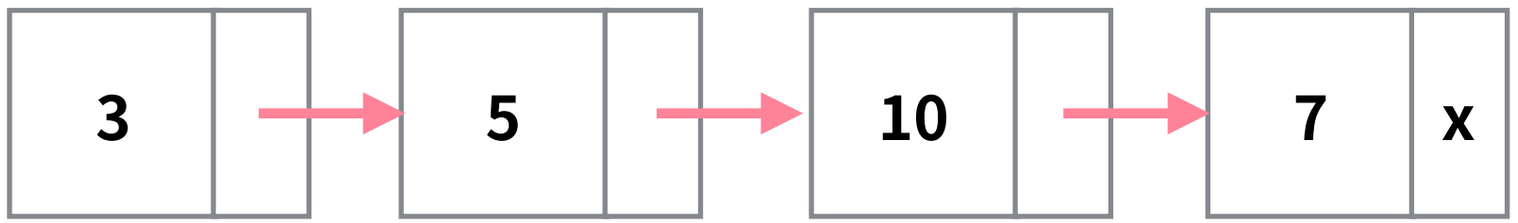
4번째 숫자 4를 넣어야지

Black box

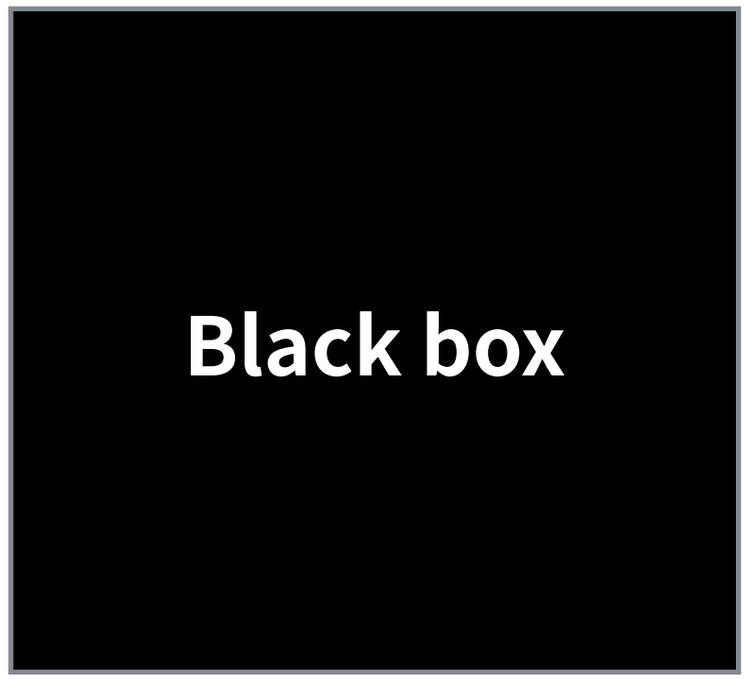
캡슐화 (Encapsulation)

자료구조를 사용하는 사람은

자료구조가 어떻게 동작하는지 알 필요가 없다



4번째 숫자 4를 넣어야지



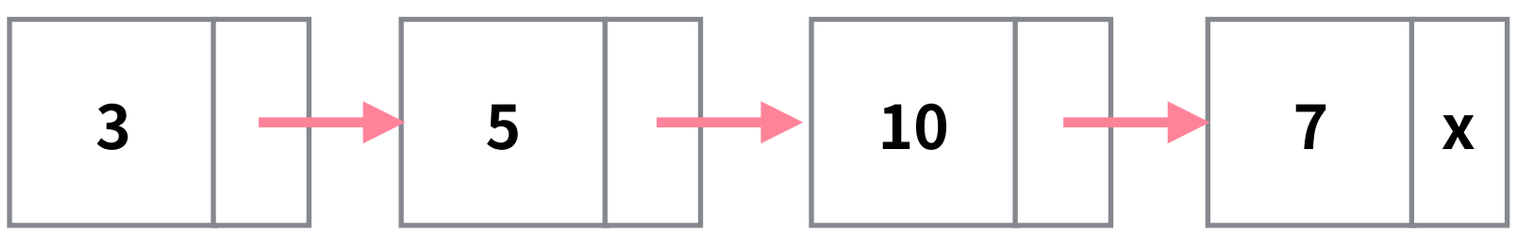
“4번째에 숫자 4좀 넣어줘”

/* elice */

캡슐화 (Encapsulation)

자료구조를 사용하는 사람은

자료구조가 어떻게 동작하는지 알 필요가 없다



4번째 숫자 4를 넣어야지

Black box

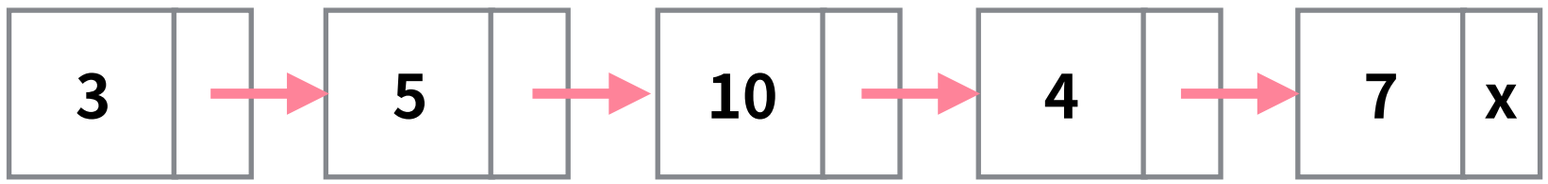
넣었습니다 :)

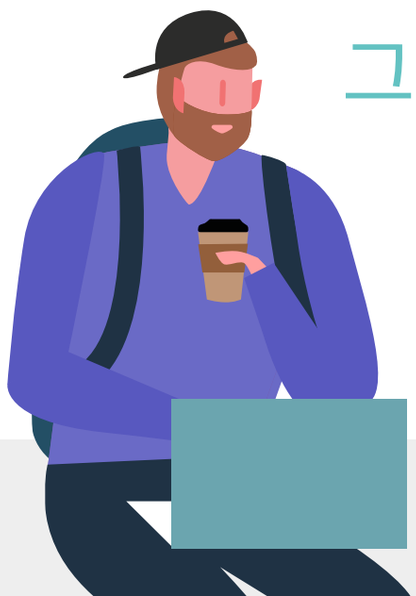
/* elice */

캡슐화 (Encapsulation)

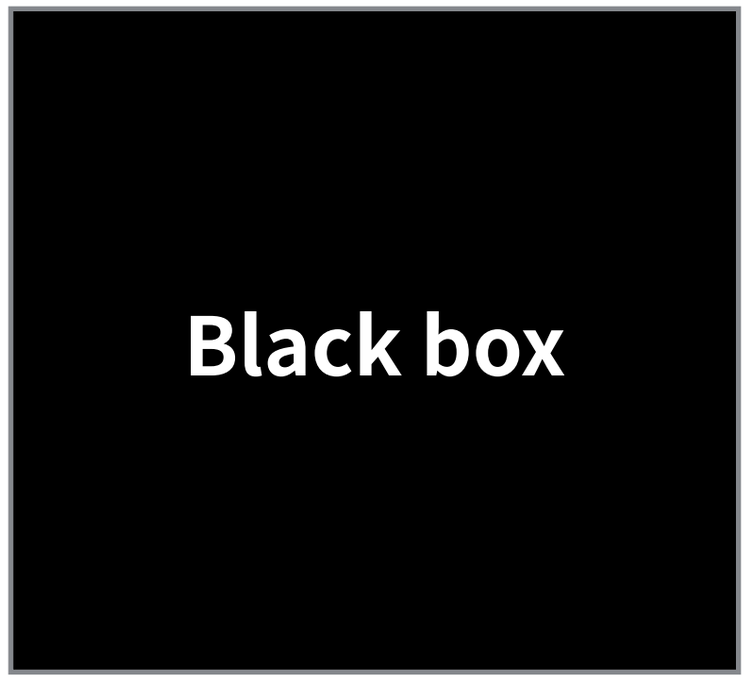
자료구조를 사용하는 사람은

자료구조가 어떻게 동작하는지 알 필요가 없다





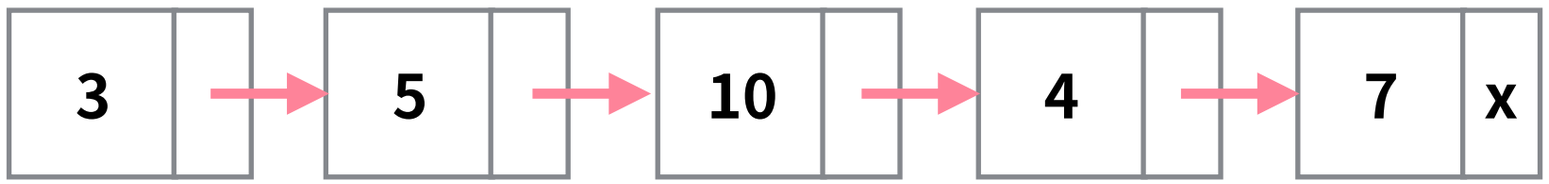
그럼 이제 잘 들어갔겠군

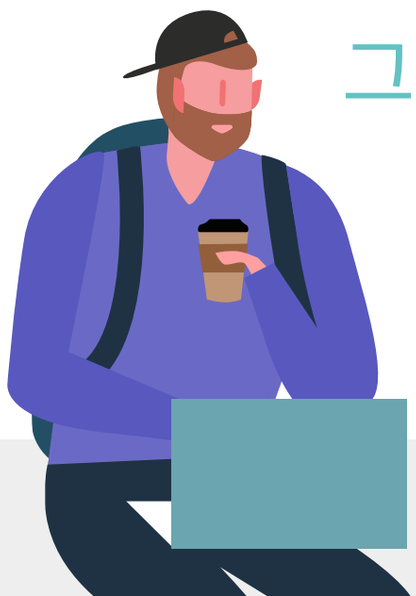


캡슐화 (Encapsulation)

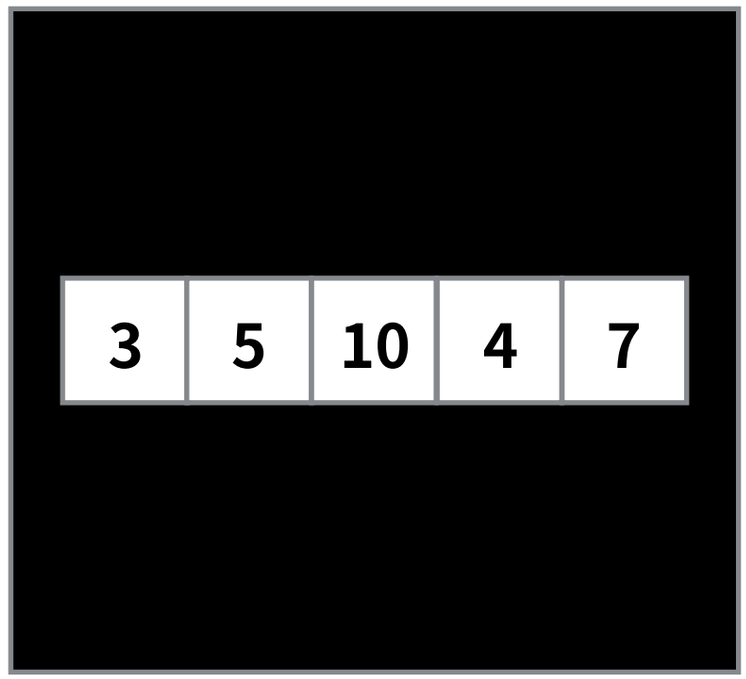
자료구조를 사용하는 사람은

자료구조가 어떻게 동작하는지 알 필요가 없다





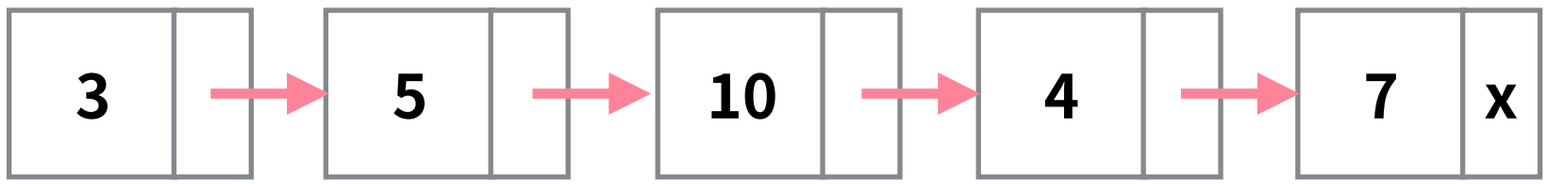
그럼 이제 잘 들어갔겠군

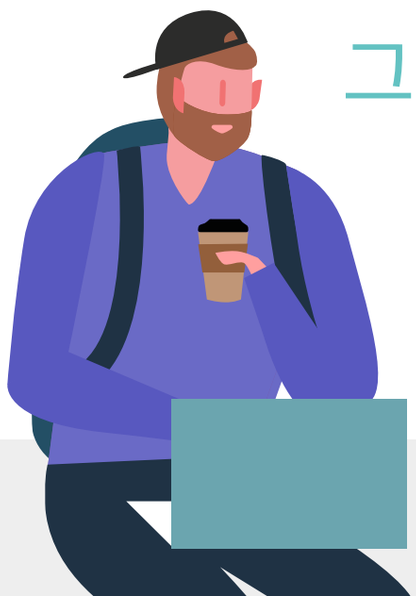


캡슐화 (Encapsulation)

자료구조를 사용하는 사람은

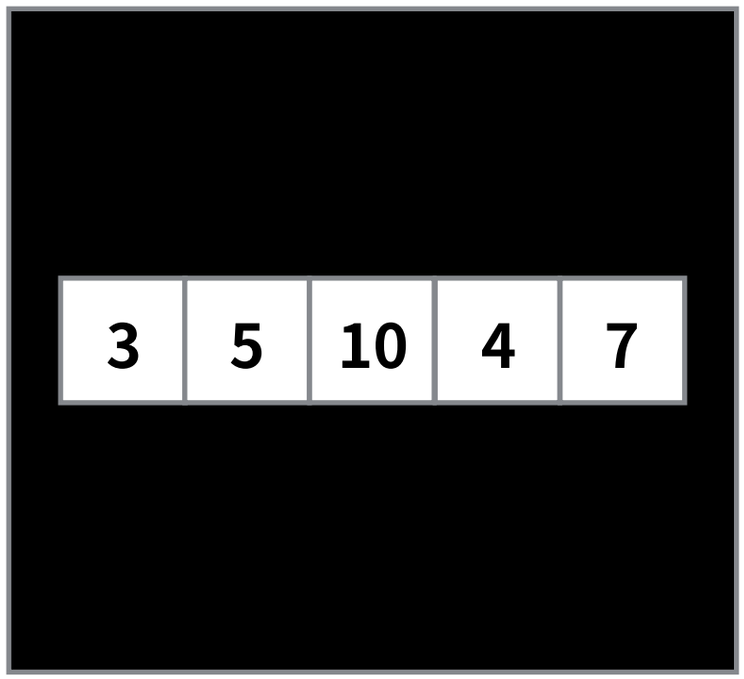
자료구조가 어떻게 동작하는지 알 필요가 없다





그럼 이제 잘 들어갔겠군

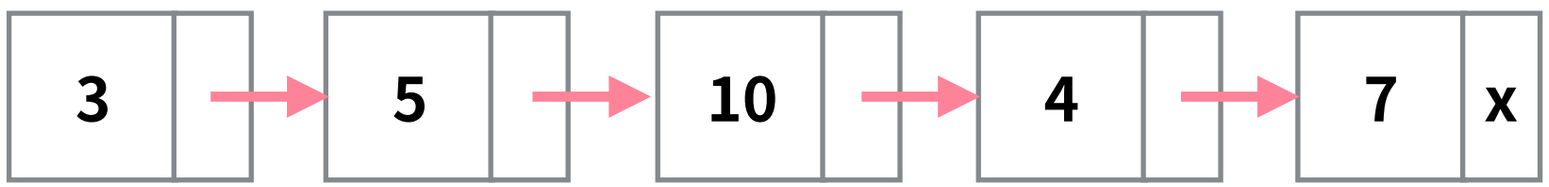
2번째 숫자를 빼자!

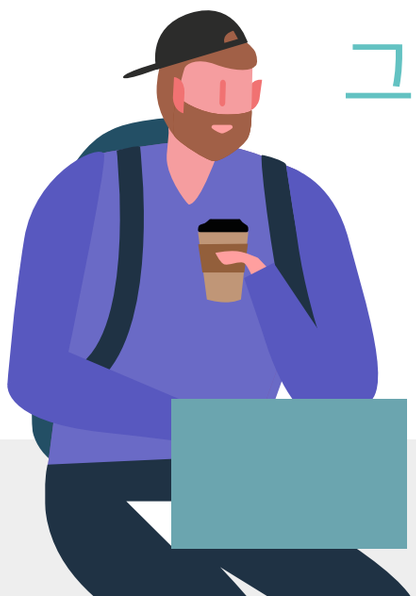


캡슐화 (Encapsulation)

자료구조를 사용하는 사람은

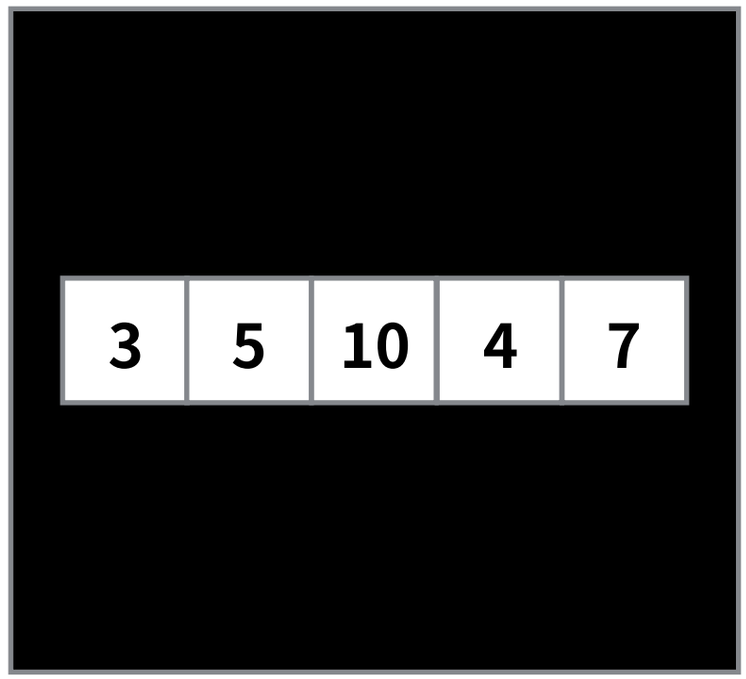
자료구조가 어떻게 동작하는지 알 필요가 없다





그럼 이제 잘 들어갔겠군

2번째 숫자를 빼자!



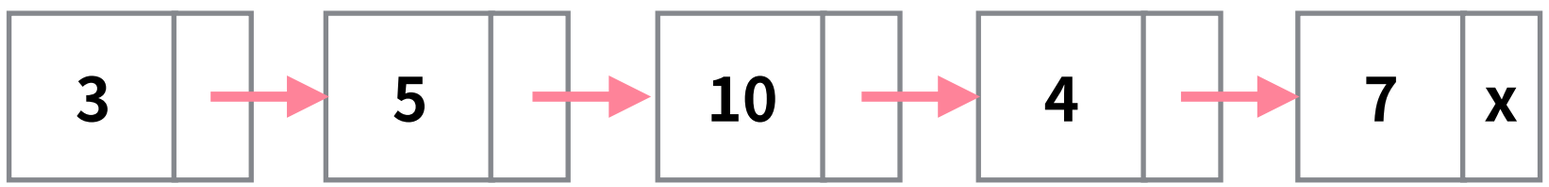
“2번째 숫자 좀 빼줘”

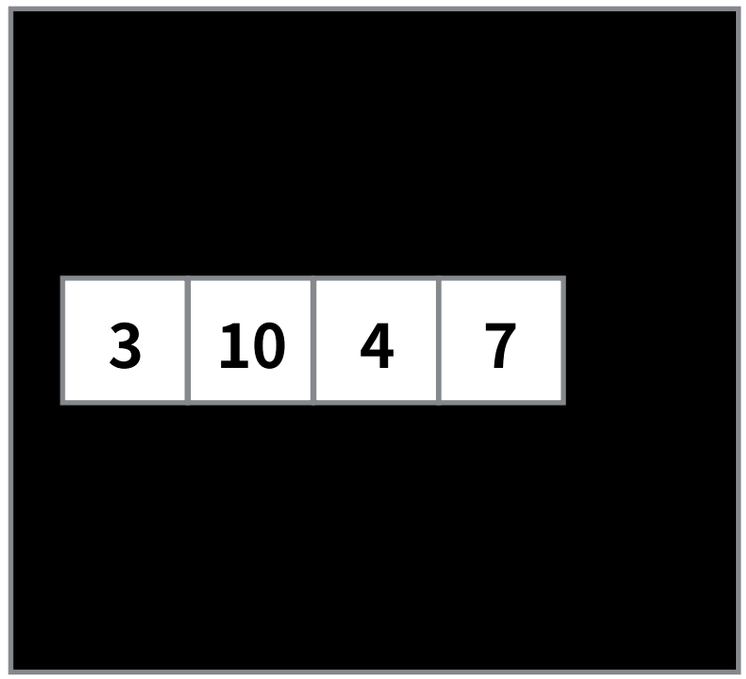
/* elice */

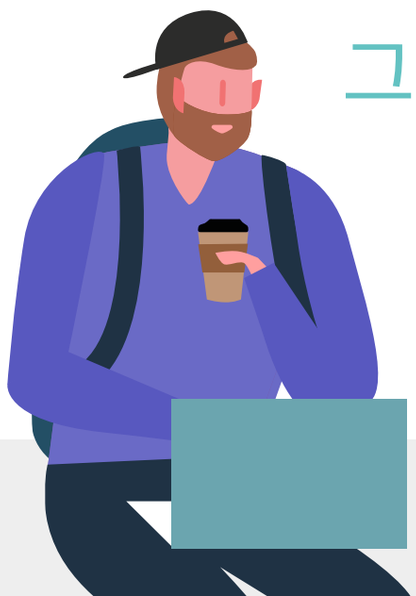
캡슐화 (Encapsulation)

자료구조를 사용하는 사람은

자료구조가 어떻게 동작하는지 알 필요가 없다







그럼 이제 잘 들어갔겠군

2번째 숫자를 빼자!

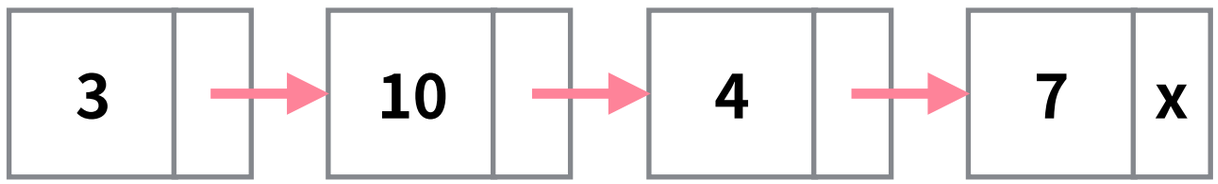
뺐습니다 :)

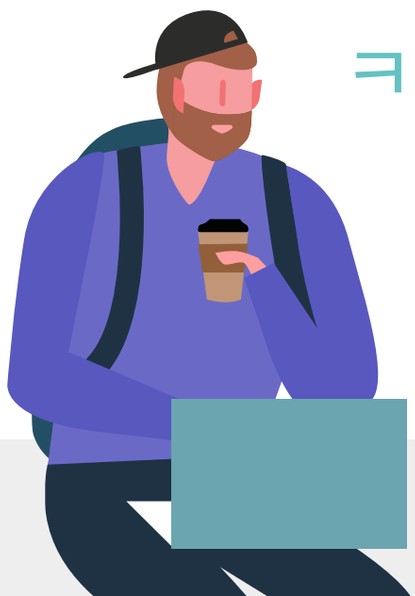
/* elice */

캡슐화 (Encapsulation)

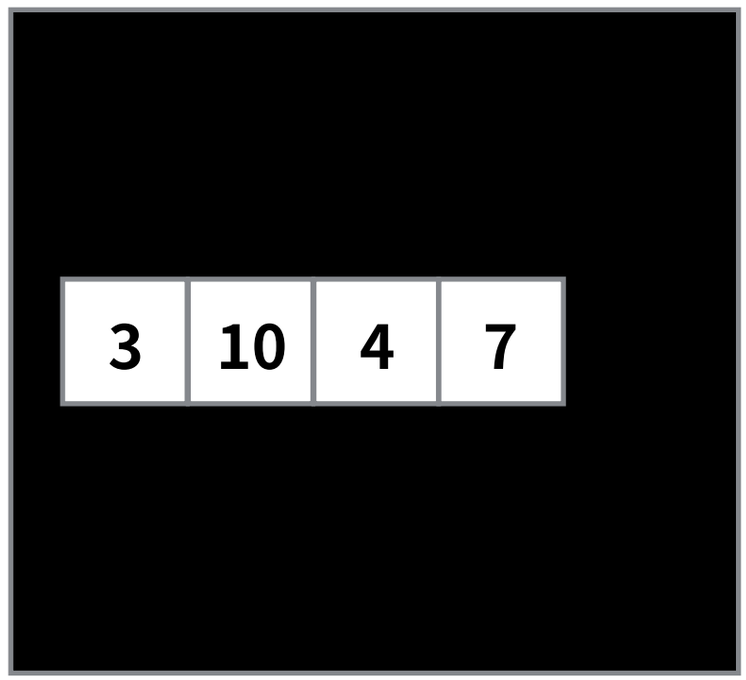
자료구조를 사용하는 사람은

자료구조가 어떻게 동작하는지 알 필요가 없다





ㅋㅋㅋ말잘듣넹



클래스 (Class) : 캡슐화의 구현

클래스는 하나의 Black box를 구현할 때 사용한다

사용자가 알아야 하는 것

이 클래스의 사용법

클래스를 제작하는 사람이 알아야 하는 것

클래스와 인스턴스

클래스와 인스턴스

클래스: 자료구조가 어떻게 구성되어 있는지에 대한 설명서

인스턴스: 하나의 자료구조 그 자체

클래스와 인스턴스

Tea Tree Special Shampoo의 설명서

클래스와 인스턴스

Tea Tree Special Shampoo의 설명서

클래스

인스턴스

클래스 예제

최댓값을 반환하는 자료구조를 만들어 보자

사용자가 알아야 하는 것 (또는 사용자가 원하는 것)

- 자료구조에 숫자를 추가하기 위한 **명령**
- 자료구조에 있는 숫자를 제거하기 위한 **명령**
- 자료구조 내에 있는 숫자들 중 최댓값을 구하기 위한 **명령**

제작하는 사람이 알아야 하는 것 (또는 구현해야 하는 것)

- 자료구조에 숫자를 추가하기 위한 **구현**
- 자료구조에 있는 숫자를 제거하기 위한 **구현**
- 자료구조 내에 있는 숫자들 중 최댓값을 구하기 위한 **구현**

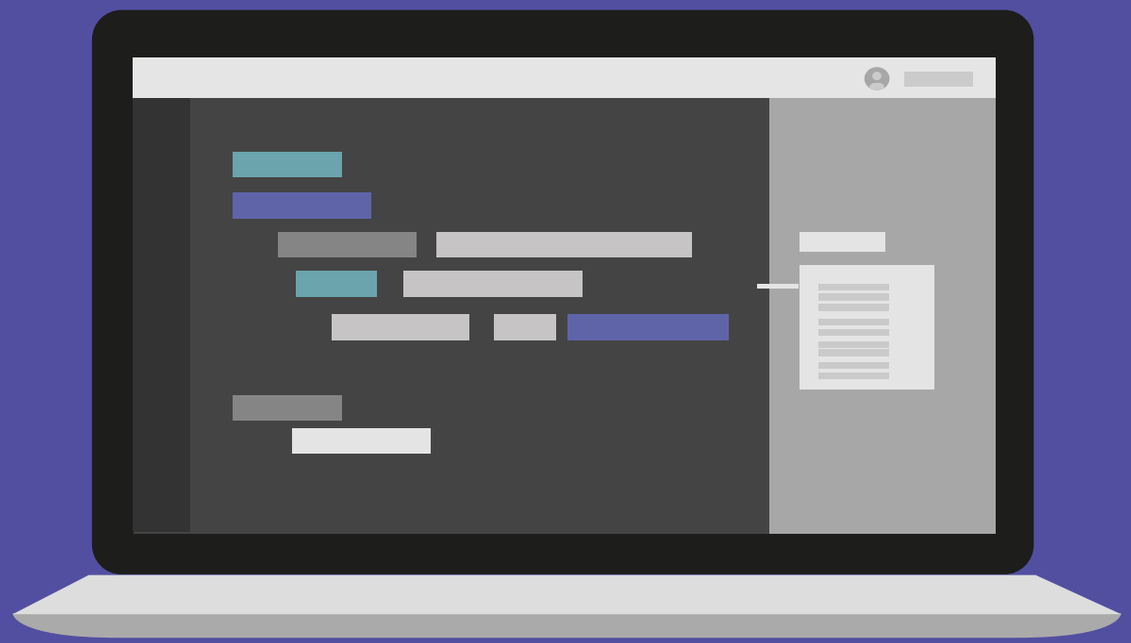
클래스 예제 : 선언

```
class maxMachine :
    def __init__(self) :
        self.data = []

    def addNumber(self, n) :
        self.data.append (n)

    def removeNumber(self, n) :
        self.data.remove (n)

    def getMax(self) :
        ret_value = -1

        for i in range(len(self.data)) :
            if self.data[i] > ret_value :
                ret_value = self.data[i]

        return ret_value
```

클래스 예제 : 클래스 사용

```
def main() :
    instance = maxMachine ()

    instance.addNumber (1)
    instance.addNumber (4)
    instance.addNumber (2)
    instance.addNumber (3)

    print(instance.getMax())

    instance.removeNumber (4)

    print(instance.getMax())
```

[예제 1] 최댓값 기계



그래서 무엇을 써야하나

리스트를 써야하나

링크드 리스트를 써야하나 ?

그래서 무엇을 써야하나

리스트를 써야하나

링크드 리스트를 써야하나 ?

그때그때 달라요

반드시 숙지할 것

자료구조를 배우는 이유

나의 목적에 맞게 데이터를 담는

그릇을 디자인하기 위함

반드시 숙지할 것

자료구조를 배우는 이유

나의 목적에 맞게 데이터를 담는

그릇을 디자인하기 위함

→ **목적이 먼저, 자료구조는 그 다음!**

반드시 숙지할 것

목적이 먼저, 자료구조는 그 다음!

1. 무슨 자료를 담을지 파악한다.
2. 이 자료를 이용해서 무엇을 할 것인지 파악한다.
3. 목적을 빠르게 달성할 수 있는 자료구조를 디자인한다.

예제: 샴푸통 제작

샴푸 보관이 용이

샴푸를 짜는게 비교적 불편

샴푸를 짜는게 비교적 편함

여전히 좀 불편

좀 더 편함

많이 짜는데 시간이 좀 걸림

우주에서 사용하기 좋음

지구에서는 별로..

예제: 샴푸통 제작

이용 목적에 따라 그 **효율성**이 다름

[예제 2] 구슬 넣기

양쪽이 열려있는 파이프에 구슬을 넣을 때,

최종 구슬 배치는 ?

입력의 예

```
3
1 0
2 1
3 0
```

출력의 예

```
3 1 2
```

해법: 리스트 사용

리스트를 파이프라 생각하고 숫자를 삽입하자

myList

해법: 리스트 사용

리스트를 파이프라 생각하고 숫자를 삽입하자

왼쪽으로 숫자 1 삽입 !

myList

해법: 리스트 사용

리스트를 파이프라 생각하고 숫자를 삽입하자

왼쪽으로 숫자 1 삽입 !

해법: 리스트 사용

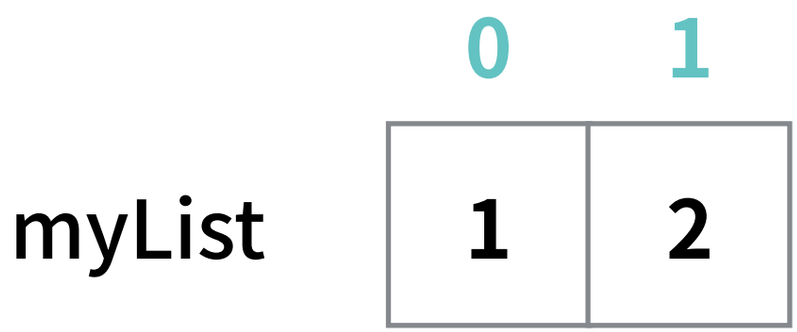
리스트를 파이프라 생각하고 숫자를 삽입하자

오른쪽으로 숫자 2 삽입 !

해법: 리스트 사용

리스트를 파이프라 생각하고 숫자를 삽입하자

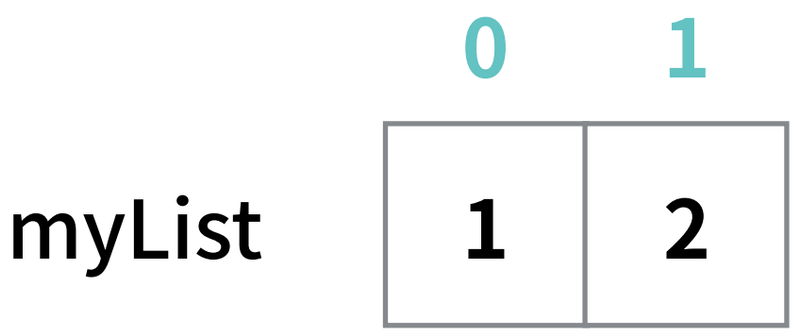
오른쪽으로 숫자 2 삽입 !



해법: 리스트 사용

리스트를 파이프라 생각하고 숫자를 삽입하자

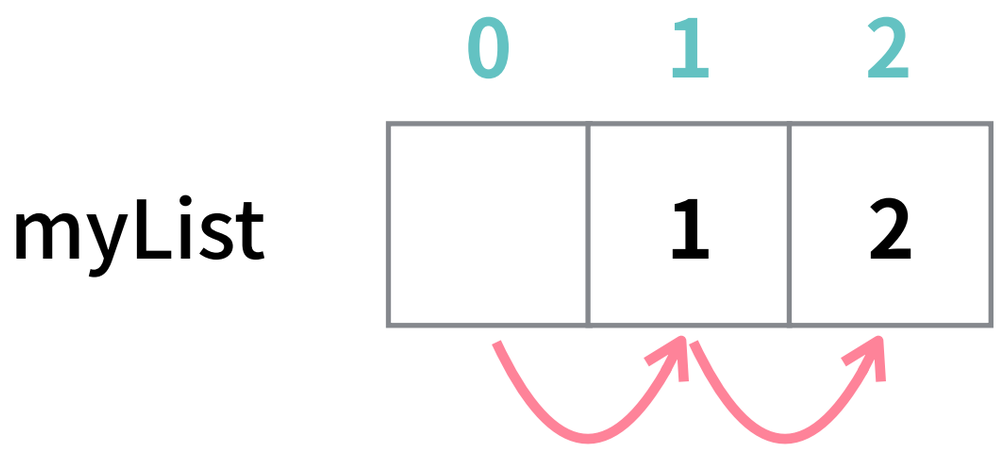
왼쪽으로 숫자 3 삽입 !



해법: 리스트 사용

리스트를 파이프라 생각하고 숫자를 삽입하자

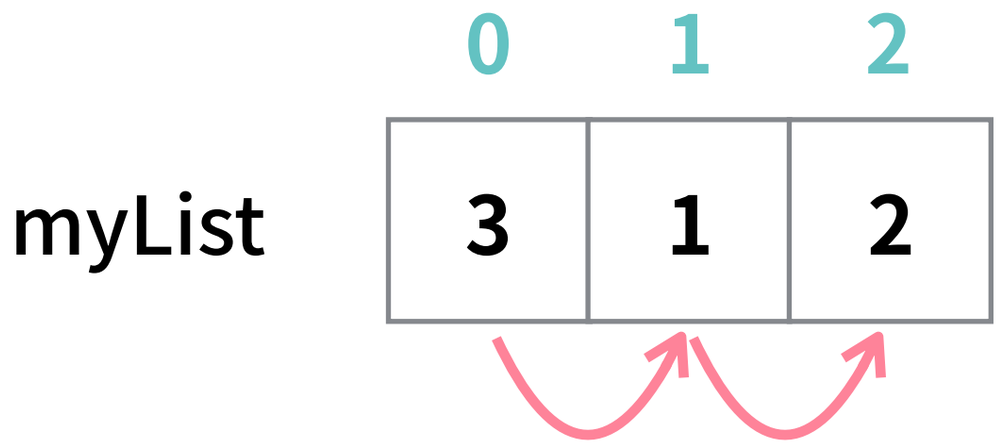
왼쪽으로 숫자 3 삽입 !



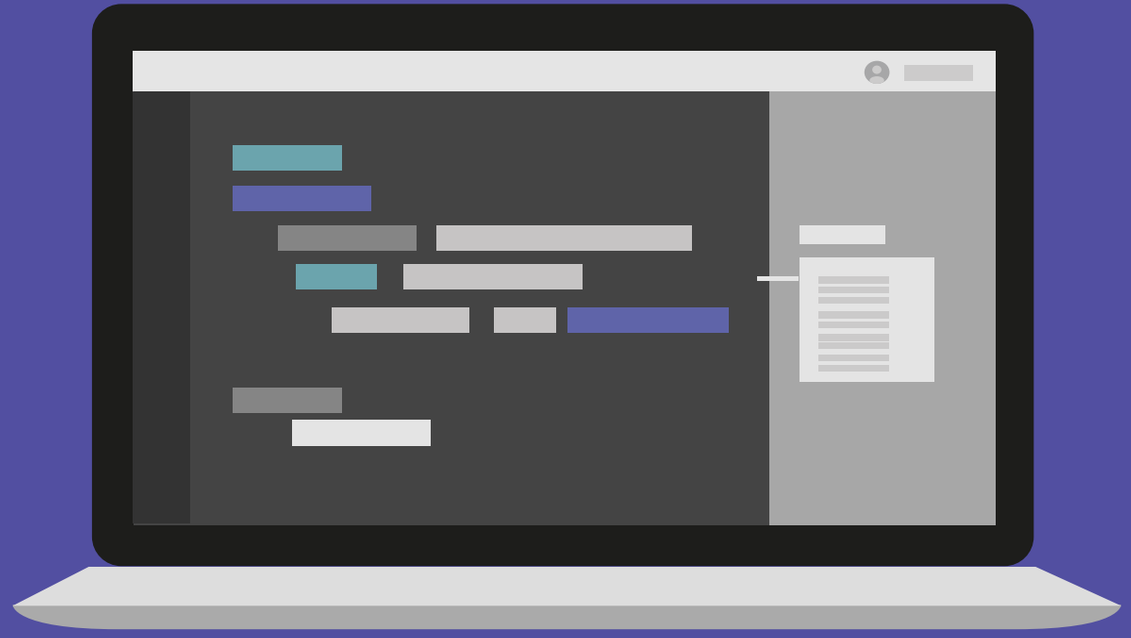
해법: 리스트 사용

리스트를 파이프라 생각하고 숫자를 삽입하자

왼쪽으로 숫자 3 삽입 !



[예제 2] 구슬 넣기 (리스트)



좋은 해법인가?

기준

좋은 해법인가?

기준

풀이가 깔끔한가 ?

얼마나 빠른가 ?

리소스는 얼마나 잡아먹는가 ?

코딩을 하는데는 얼마나 오래 걸리는가 ?

좋은 해법인가?

기준

~~풀이가 깔끔한가 ?~~

얼마나 빠른가 ?

~~리소스는 얼마나 잡아먹는가 ?~~

~~코딩을 하는데는 얼마나 오래 걸리는가 ?~~

시간이 덜 걸리는 코드

수행하는 명령의 수가 적으면 시간이 덜 걸린다

```
sum = 0
for i in range(30)
    sum = sum + 1
```

```
sum = 0
for i in range(300)
    sum = sum + 1
```

시간이 덜 걸리는 코드

수행하는 명령의 수가 적으면 시간이 덜 걸린다

```
sum = 0
for i in range(30)
    sum = sum + 1
```

```
sum = 0
for i in range(300)
    sum = sum + 1
```

거의 10배라고 봐도됨

이 풀이가 수행하는 명령 수

리스트를 파이프라 생각하고 숫자를 삽입하자

왼쪽으로 숫자 1 삽입 !

myList

이 풀이가 수행하는 명령 수

리스트를 파이프라 생각하고 숫자를 삽입하자

왼쪽으로 숫자 1 삽입 !

이 풀이가 수행하는 명령 수

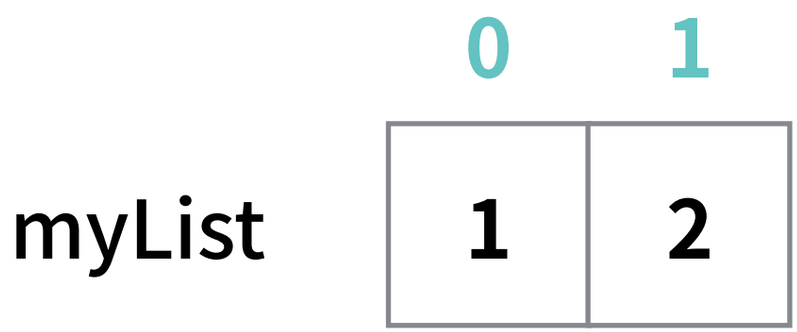
리스트를 파이프라 생각하고 숫자를 삽입하자

오른쪽으로 숫자 2 삽입 !

이 풀이가 수행하는 명령 수

리스트를 파이프라 생각하고 숫자를 삽입하자

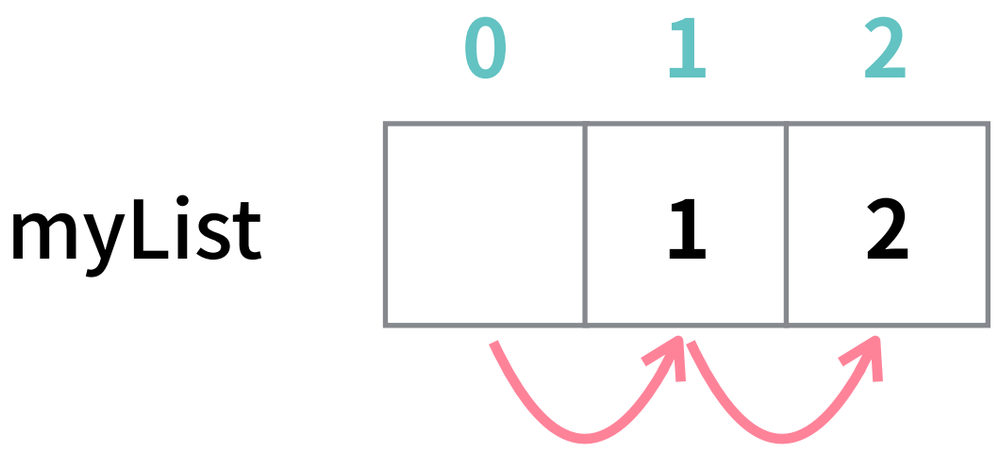
오른쪽으로 숫자 2 삽입 !



이 풀이가 수행하는 명령 수

리스트를 파이프라 생각하고 숫자를 삽입하자

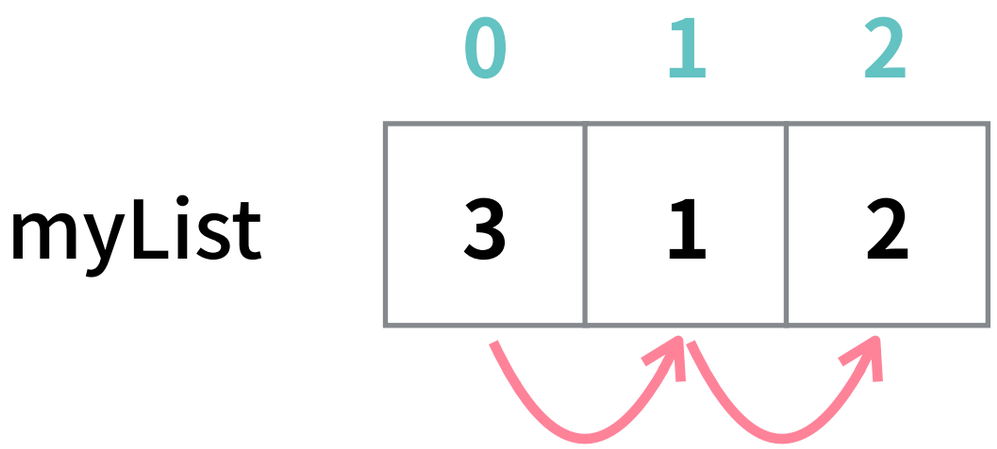
왼쪽으로 숫자 3 삽입 !



이 풀이가 수행하는 명령 수

리스트를 파이프라 생각하고 숫자를 삽입하자

왼쪽으로 숫자 3 삽입 !



대충 몇개의 명령을 수행할까 ?

대충 몇개의 명령을 수행할까 ?

우리 알고리즘이 무슨 일을 하는가 ?

대충 몇개의 명령을 수행할까 ?

우리 알고리즘이 무슨 일을 하는가 ?

숫자 하나를 왼쪽으로 삽입

숫자 하나를 오른쪽으로 삽입 → 1번의 명령

대충 몇개의 명령을 수행할까 ?

우리 알고리즘이 무슨 일을 하는가 ?

숫자 하나를 왼쪽으로 삽입 → 숫자 개수만큼

숫자 하나를 오른쪽으로 삽입 → 1번의 명령

대충 몇개의 명령을 수행할까 ?

우리 알고리즘이 무슨 일을 하는가 ?

숫자 하나를 왼쪽으로 삽입 → 숫자 개수만큼

숫자 하나를 오른쪽으로 삽입 → 1번의 명령

운 좋을땐 빠르고 운 나쁠땐 느리다

항상 운이 나쁘진 않은데...

근데 **만약에** 운이 나쁘면?

최악의 경우에는 어떤가 ?

최악의 경우에는 어떤가 ?

숫자 n개를 모두 왼쪽으로만 삽입하는 경우

최악의 경우에는 어떤가 ?

숫자 n개를 모두 왼쪽으로만 삽입하는 경우

만약 숫자가 3개라면

```
1 + 2 + 3 = 6
```

최악의 경우에는 어떤가 ?

숫자 n개를 모두 왼쪽으로만 삽입하는 경우

만약 숫자가 3개라면

```
1 + 2 + 3 = 6
```

만약 숫자가 5개라면

```
1 + 2 + 3 + 4 + 5 = 15
```

최악의 경우에는 어떤가 ?

숫자 n개를 모두 왼쪽으로만 삽입하는 경우

만약 숫자가 3개라면

$$1 + 2 + 3 = 6$$

만약 숫자가 5개라면

$$1 + 2 + 3 + 4 + 5 = 15$$

만약 숫자가 n개라면

$$\frac{n(n+1)}{2}$$

결론

현재 알고리즘은 최악의 경우에 제곱으로 오래걸린다

결론

현재 알고리즘은 최악의 경우에 제곱으로 오래걸린다

더 잘할수는 없나 ?

결론

현재 알고리즘은 최악의 경우에 제곱으로 오래걸린다

더 잘할수는 없나 ?

더 빠른 풀이를 만들 수는 없나 ?

해법: 링크드 리스트 사용

링크드 리스트를 파이프라 생각하고 숫자를 삽입하자

해법: 링크드 리스트 사용

링크드 리스트를 파이프라 생각하고 숫자를 삽입하자

왼쪽으로 숫자 1 삽입 !

myLinkedList

해법: 링크드 리스트 사용

링크드 리스트를 파이프라 생각하고 숫자를 삽입하자

왼쪽으로 숫자 1 삽입 !

해법: 링크드 리스트 사용

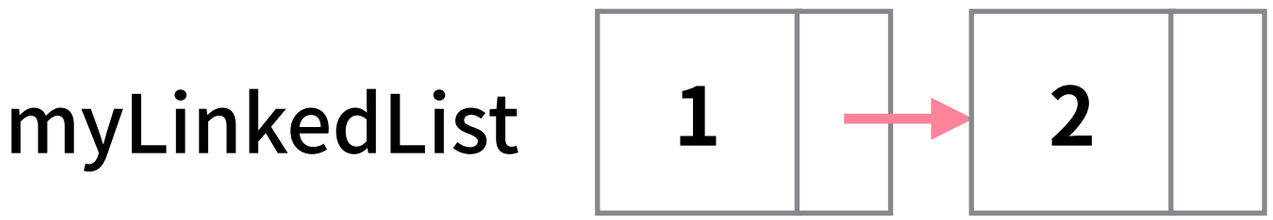
링크드 리스트를 파이프라 생각하고 숫자를 삽입하자

오른쪽으로 숫자 2 삽입 !

해법: 링크드 리스트 사용

링크드 리스트를 파이프라 생각하고 숫자를 삽입하자

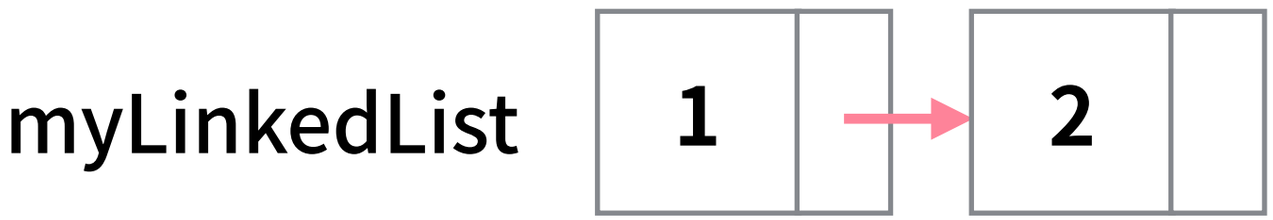
오른쪽으로 숫자 2 삽입 !



해법: 링크드 리스트 사용

링크드 리스트를 파이프라 생각하고 숫자를 삽입하자

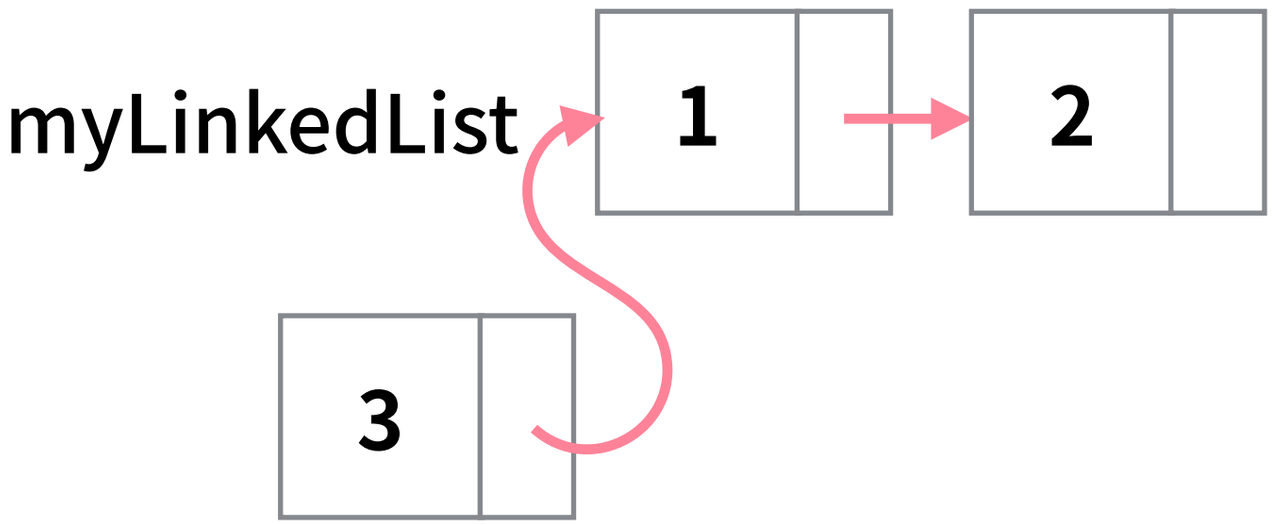
왼쪽으로 숫자 3 삽입 !



해법: 링크드 리스트 사용

링크드 리스트를 파이프라 생각하고 숫자를 삽입하자

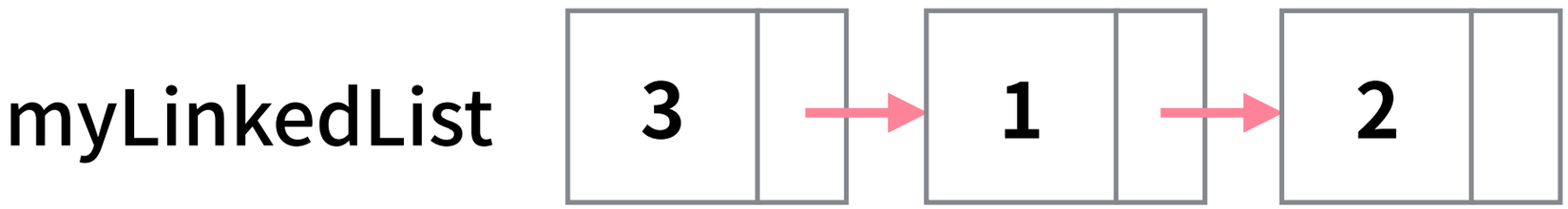
왼쪽으로 숫자 3 삽입 !



해법: 링크드 리스트 사용

링크드 리스트를 파이프라 생각하고 숫자를 삽입하자

왼쪽으로 숫자 3 삽입 !



대충 몇개의 명령을 수행할까 ?

우리 알고리즘이 무슨 일을 하는가 ?

대충 몇개의 명령을 수행할까 ?

우리 알고리즘이 무슨 일을 하는가 ?

숫자 하나를 왼쪽으로 삽입 → 1번의 명령

숫자 하나를 오른쪽으로 삽입 → 1번의 명령

대충 몇개의 명령을 수행할까 ?

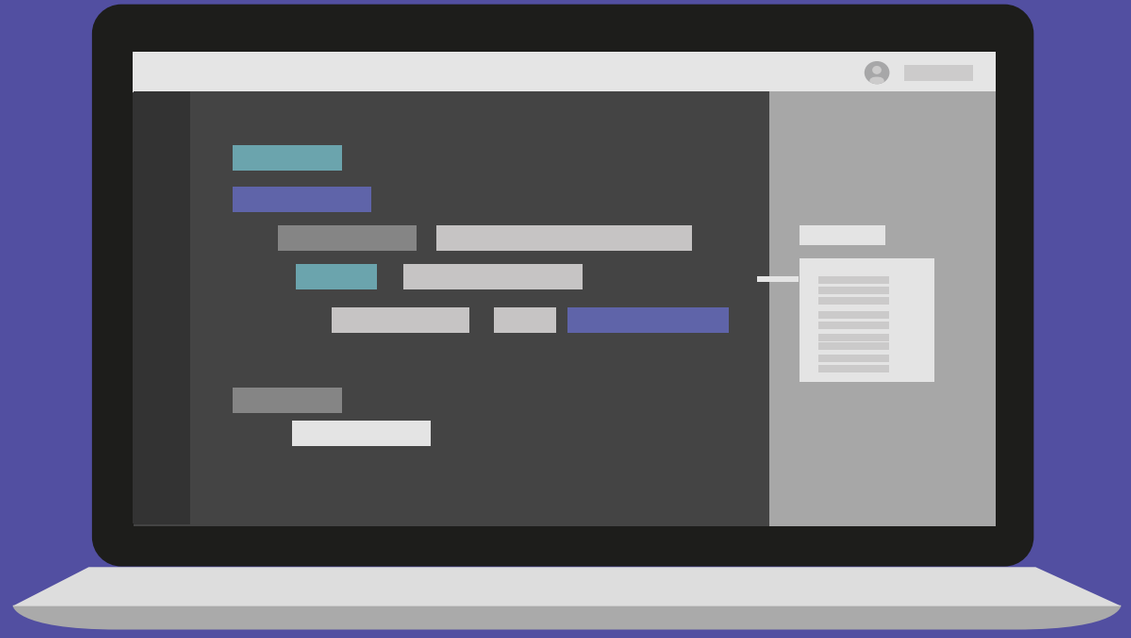
우리 알고리즘이 무슨 일을 하는가 ?

숫자 하나를 왼쪽으로 삽입 → 1번의 명령

숫자 하나를 오른쪽으로 삽입 → 1번의 명령

운 좋을땐 빠르고 **운 나쁠때 빠르다**

[예제 3] 구슬 넣기 (링크드리스트)



수행 시간 비교

링크드 리스트로 구현한 것이

리스트로 구현한 것보다 **빠르다**

연산을 덜 하기 때문

수행 시간 비교

링크드 리스트로 구현한 것이

리스트로 구현한 것보다 **빠르다**

연산을 덜 하기 때문

따라서 이 문제를 풀 때에는

링크드 리스트를 쓰는 것이 더 좋다

어느 자료구조를 써야하나 ?

이 문제에서는 링크드 리스트가 더 빨랐음

어느 자료구조를 써야하나 ?

이 문제에서는 링크드 리스트가 더 빨랐음

연산 횟수를 계산

어느 자료구조를 써야하나 ?

이 문제에서는 링크드 리스트가 더 빨랐음

연산 횟수를 계산

코딩하기 전에 파악할 수 있음

어느 자료구조를 써야하나 ?

이 문제에서는 링크드 리스트가 더 빨랐음

연산 횟수를 계산

코딩하기 전에 파악할 수 있음

목적 달성을 위한 연산 횟수를 줄이는

자료구조를 택하자

어느 자료구조를 써야하나 ?

이 문제에서는 링크드 리스트가 더 빨랐음

연산 횟수를 계산

코딩하기 전에 파악할 수 있음

목적 달성을 위한 연산 횟수를 줄이는

자료구조를 택하자

자료구조가 실제로 어떻게 구현되어 있는지를 알아야 함

어느 자료구조를 써야하나 ?

이 문제에서는 링크드 리스트가 더 빨랐음

연산 횟수를 계산

코딩하기 전에 파악할 수 있음

목적 달성을 위한 연산 횟수를 줄이는

자료구조를 택하자

자료구조가 실제로 어떻게 구현되어 있는지를 알아야 함

라이브러리가 어떻게 구현되어 있는지를 모르면 분석할 수 없음

어느 자료구조를 써야하나 ?

연산 횟수를 줄이면 장땡인가 ?

꼭 그렇지는 않지만,

적어도 이번 과정에서는 Yes

감사합니다!

신현규

E-mail : hyungyu.sh@kaist.ac.kr

Kakao : yougatup

/* elice */

문의 및 연락처

academy.elice.io

contact@elice.io

facebook.com/elice.io

blog.naver.com/elicer