HAPPY HACKING

스택, 큐 (Stack, Queue)

HAPPY HACKING

1. 스택 (Stack)

2. 큐 (Queue)

프로그램 = 데이터 구조 + 알고리즘

Niklaus Wirth

Data Structure 데이터 구조

데이터를 다양한 방식으로 **저장**하고

조회, 삽입, 변경, 삭제와 같은 **조작** 기능 제공한다.

왜 데이터 구조가 중요한가?

그냥 아무데나, 아무렇게나 담으면 안될까?

HAPPY HACKING

아무데나 담기 ~= 변수

0. 데이터 구조 & 알고리즘

문제 상황에 따라 **더 적합한 통**이 필요하다!

0. 데이터 구조 & 알고리즘

문제 상황에 따라 **더 적합한 도구**가 필요하다!

물통 == 물 + 통

물을 필요에 따라 저장하고 활용할 수 있으므로

문제를 더 효율적으로 풀기 위한 도구가 된다.

데이터 구조 == 데이터 + 구조

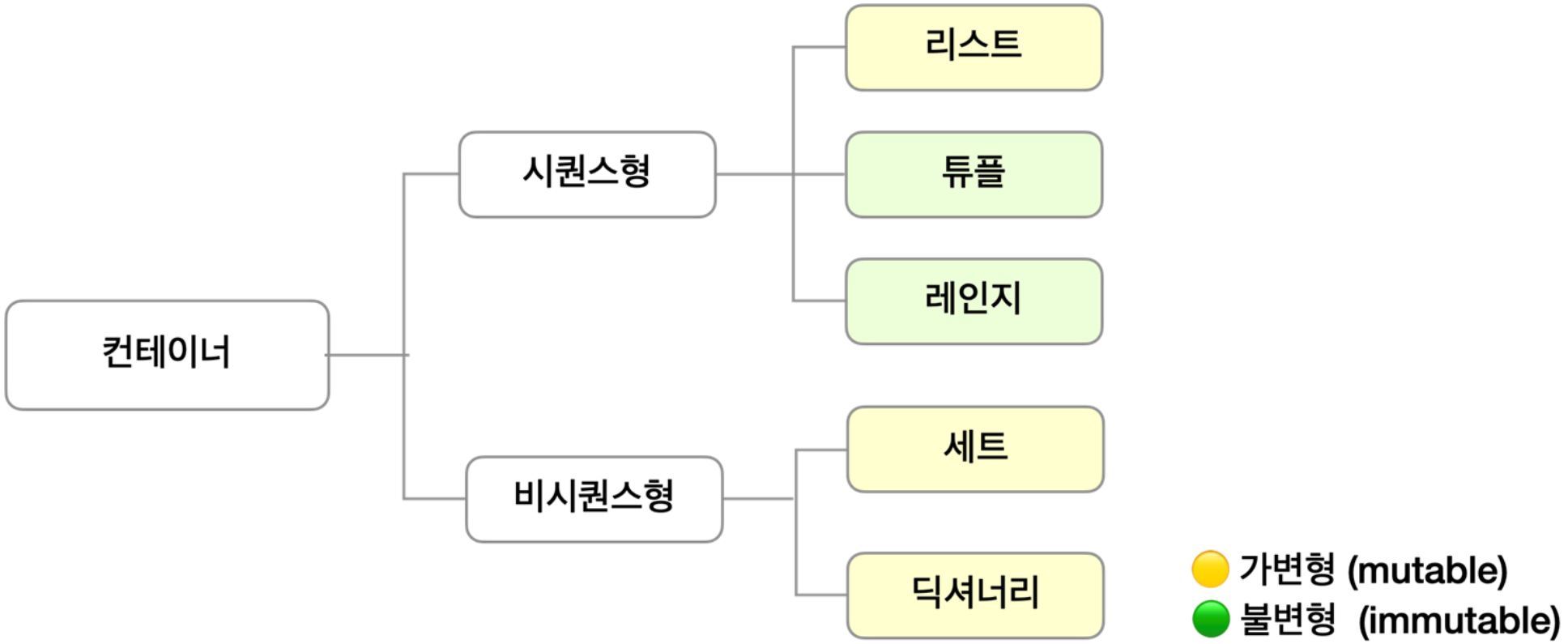
데이터를 필요에 따라 저장하고 활용할 수 있으므로

문제를 더 효율적으로 풀기 위한 도구가 된다.

어떻게 저장하고 &
어떻게 활용(조작)할 수 있는지

구조를 안다는 것

파이썬의 기본 데이터 구조



4. 데이터 구조 & 알고리즘

코딩 테스트 정복을 위한 데이터 구조와 알고리즘

- ~~Array (배열)~~
- ~~Linked List (연결리스트)~~
- ~~Hash (해시)~~
- Stack (스택)
- Queue (큐)
- Priority Queue (우선순위 큐)
- Heap (힙)
- Tree (트리)
- Graph (그래프)

[기본]
완전탐색, 재귀,
시뮬레이션, 그리디

[심화]
DFS, BFS, 백트래킹,
이진탐색, DP, 다익스트라,
크루스칼, 프림

왜 써야하는지(why)

데이터 구조를 배우는 이유: 왜 만들어졌고, 언제 써야하는지 알기 위해

HAPPY HACKING

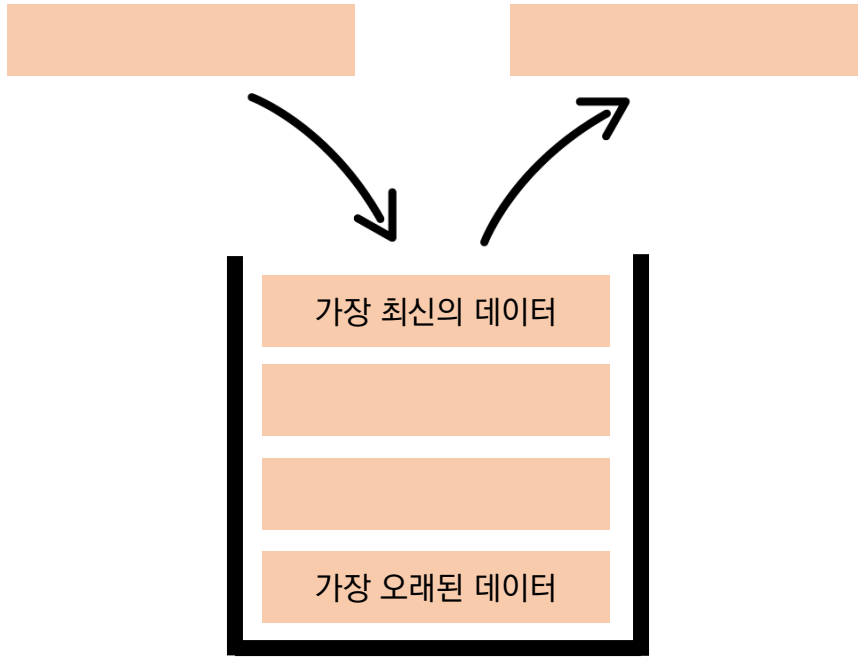
1. 스택 (Stack)

1. 스택 (Stack)

HAPPY HACKING

Stack은 쌓는다는 의미로써, 마치 접시를 쌓고 빼듯이 데이터를 한쪽에서만 넣고 빼는 자료구조

가장 마지막에 들어온 데이터가 가장 먼저 나가므로 LIFO(Last-in First-out, 후입선출) 방식

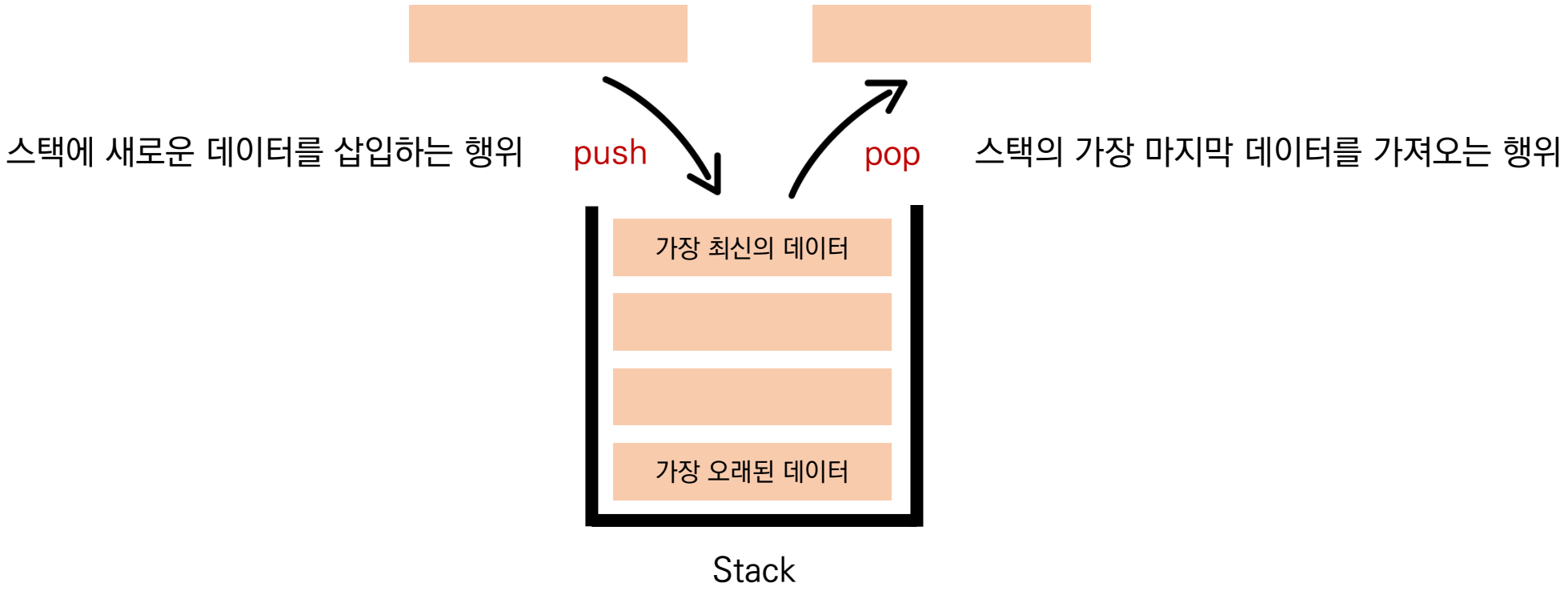


스택의 동작 과정

1. 스택 (Stack)

HAPPY HACKING

스택 자료구조의 대표 동작



후입선출: 들어온 순서와 반대로 나감

Stack의 특징

왜 Stack을 써야할까(why) ?

데이터 구조를 배우는 이유: 왜 만들어졌고, 언제 써야하는지 알기 위해

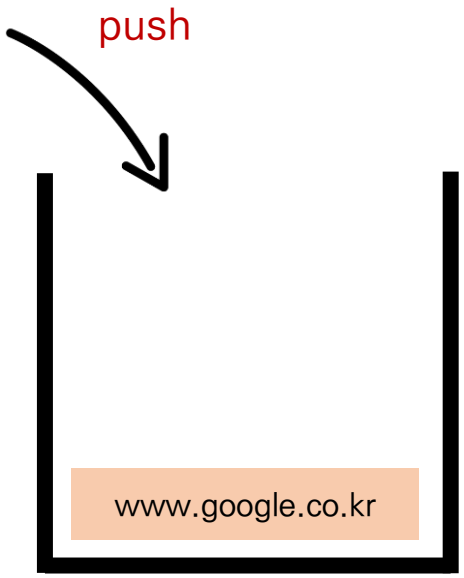
1. 뒤집기, 되돌리기, 되돌아가기

Stack이 필요한 이유 == Stack의 Use Case

1. 스택 (Stack)

HAPPY HACKING

스택 자료구조 쉽게 이해하기 – 1



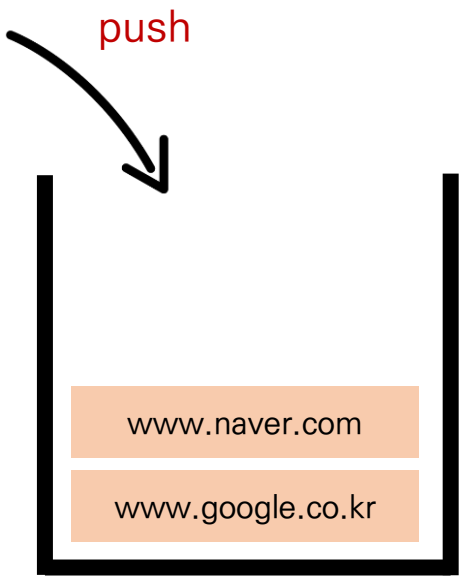
Stack

내 브라우저 화면

1. 스택 (Stack)

HAPPY HACKING

스택 자료구조 쉽게 이해하기 – 2



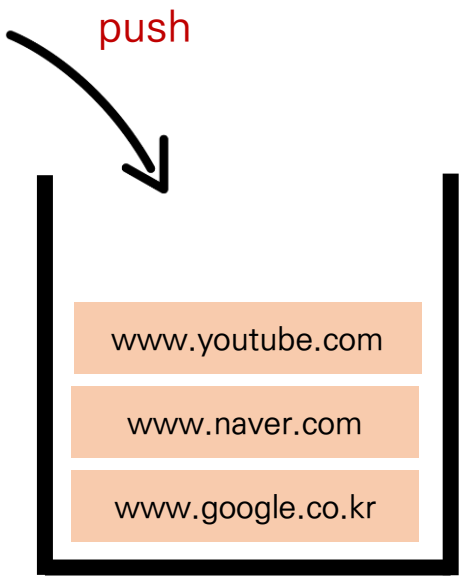
Stack

내 브라우저 화면

1. 스택 (Stack)

HAPPY HACKING

스택 자료구조 쉽게 이해하기 – 3



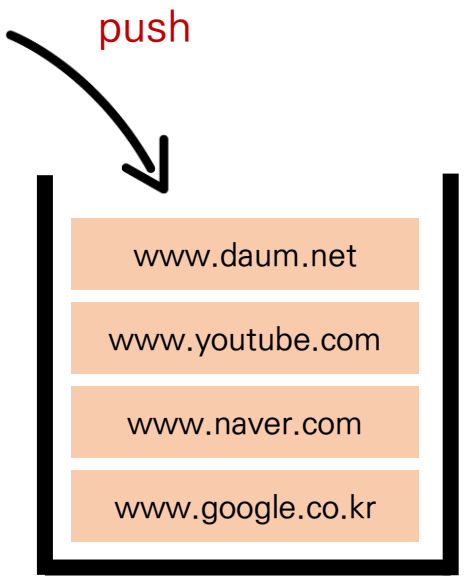
Stack

내 브라우저 화면

1. 스택 (Stack)

HAPPY HACKING

스택 자료구조 쉽게 이해하기 – 4



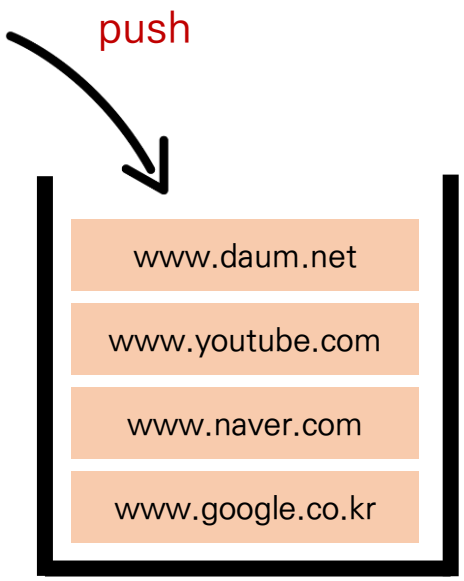
Stack

내 브라우저 화면

1. 스택 (Stack)

HAPPY HACKING

스택 자료구조 쉽게 이해하기 – 5

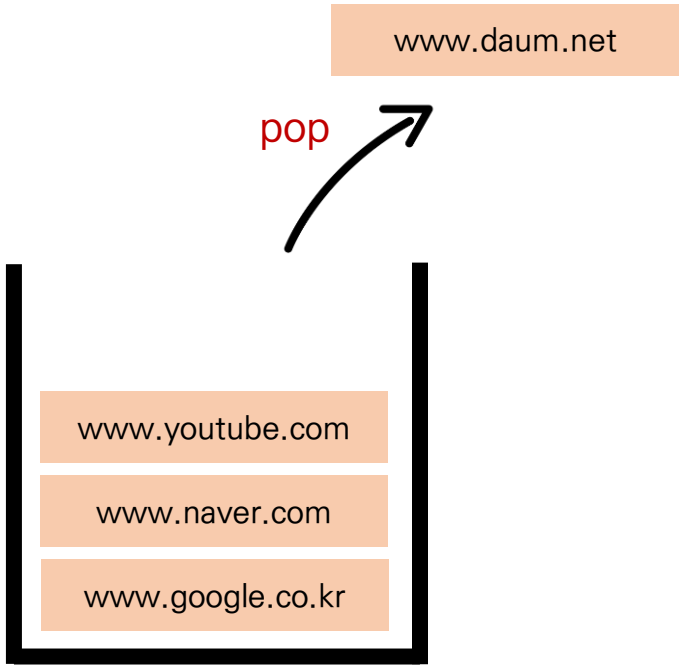


Stack

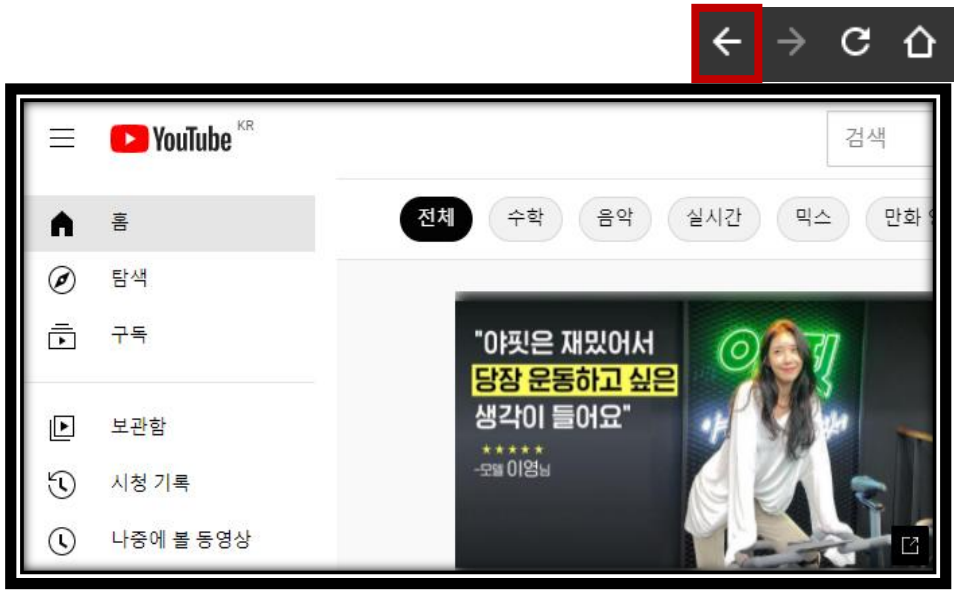
내 브라우저 화면

1. 스택 (Stack)

스택 자료구조 쉽게 이해하기 – 6



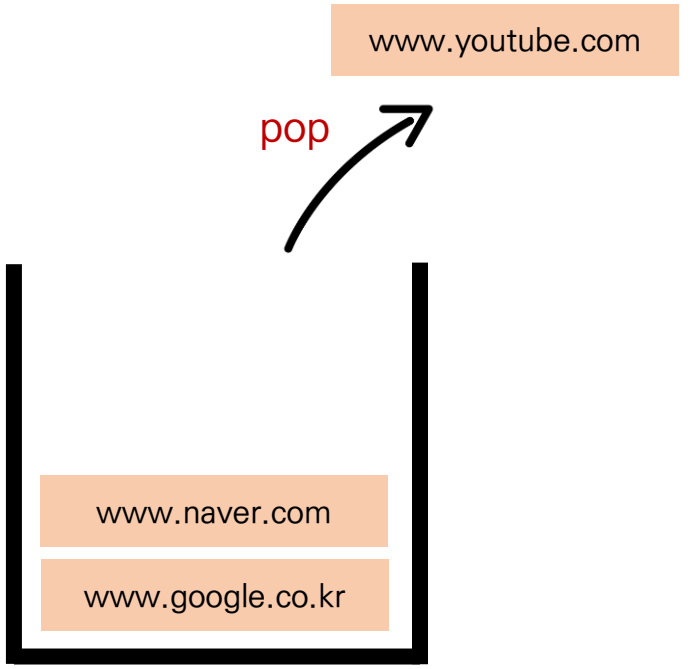
Stack



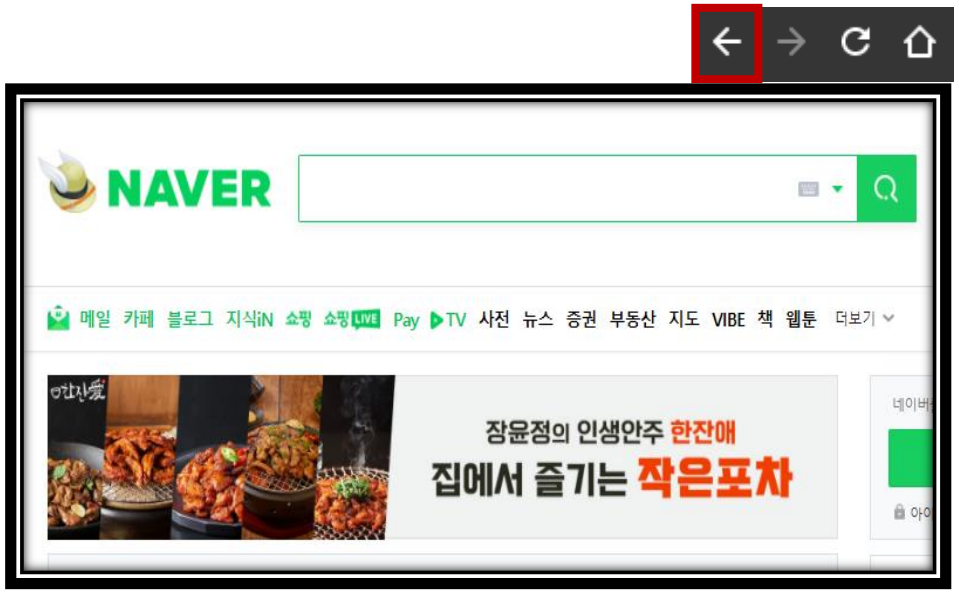
내 브라우저 화면

1. 스택 (Stack)

스택 자료구조 쉽게 이해하기 – 7



Stack

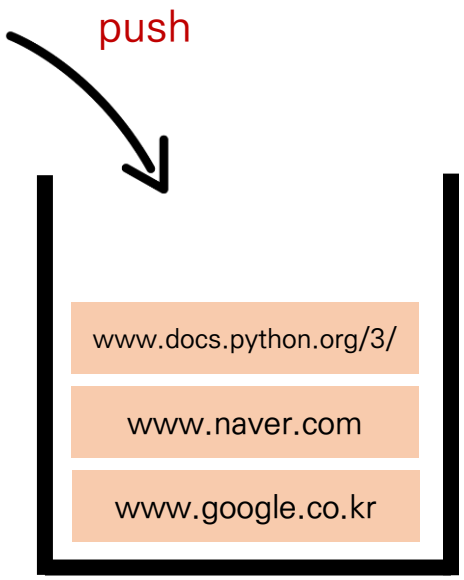


내 브라우저 화면

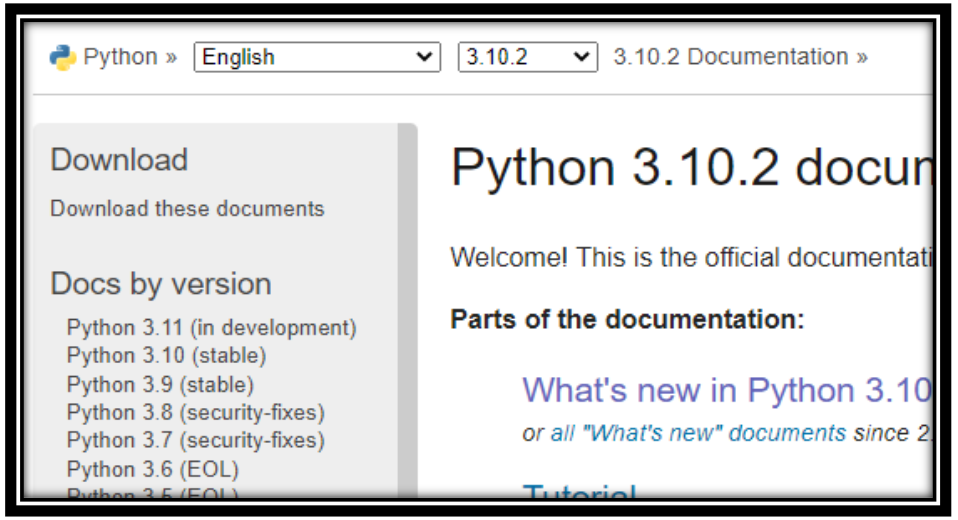
1. 스택 (Stack)

HAPPY HACKING

스택 자료구조 쉽게 이해하기 – 8



Stack



내 브라우저 화면

HAPPY HACKING

브라우저 히스토리
ctrl + z
단어 뒤집기

2. 마무리 되지 않은 일을 임시 저장

Stack이 필요한 이유 == Stack의 Use Case

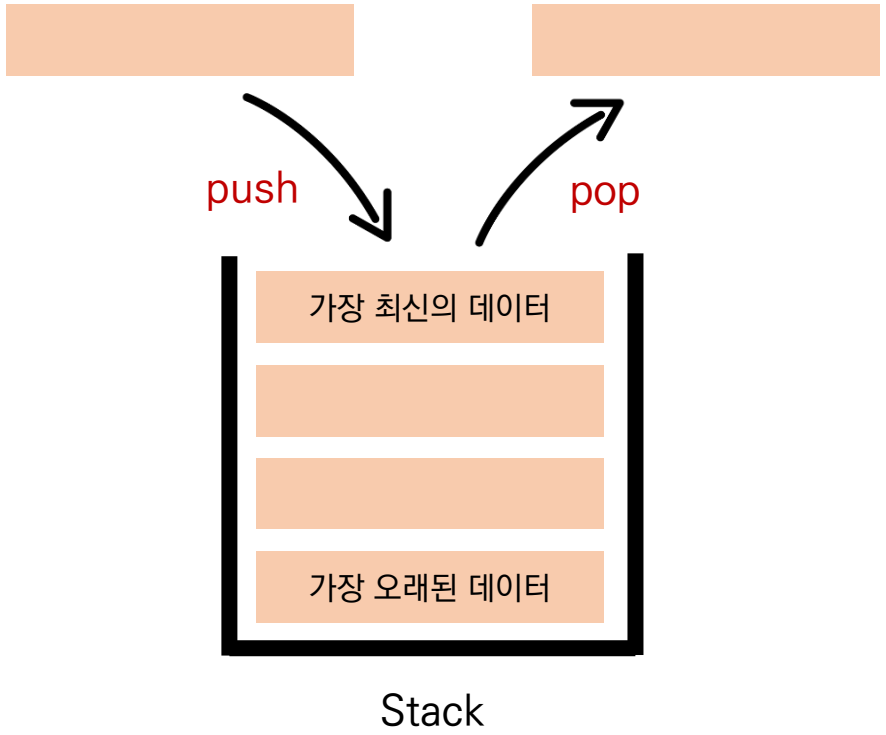
HAPPY HACKING

괄호 매칭
함수 호출
백트래킹
DFS(깊이 우선 탐색)

1. 스택 (Stack)

HAPPY HACKING

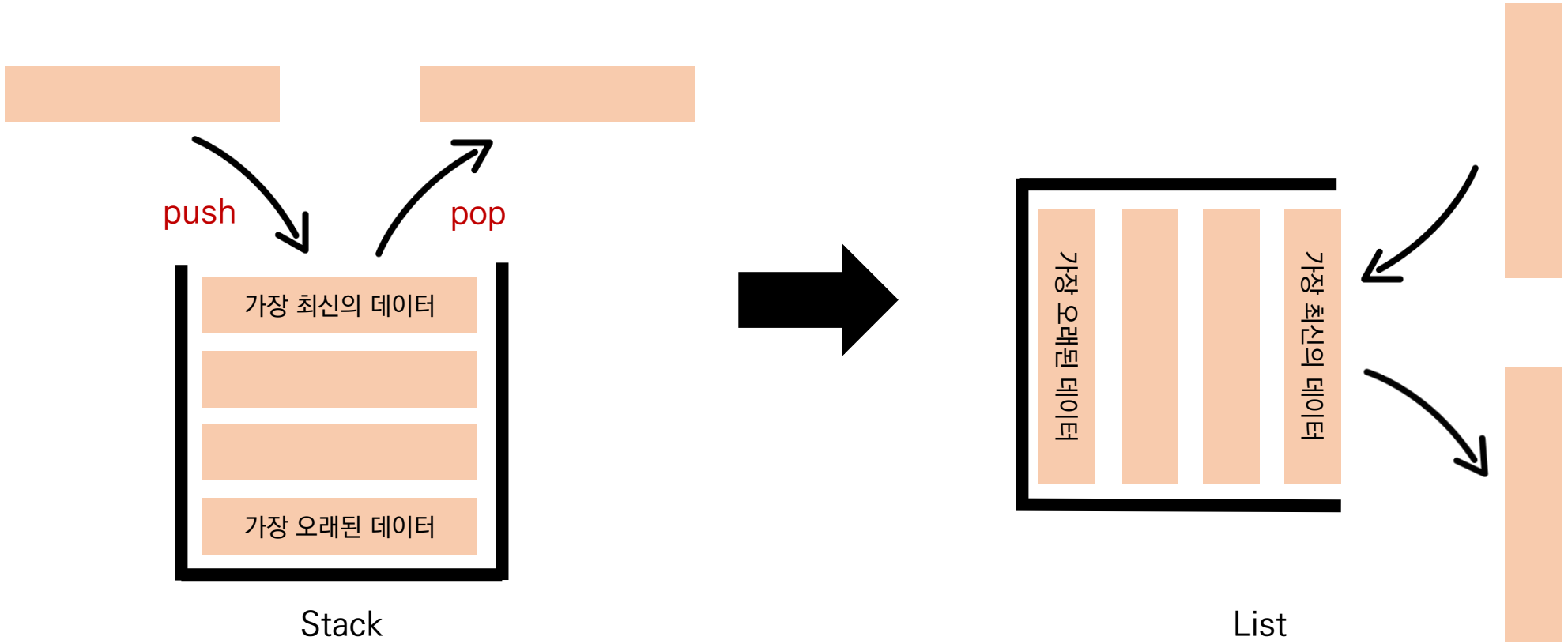
파이썬은 리스트(List)로 스택을 간편하게 사용할 수 있다 !



1. 스택 (Stack)

HAPPY HACKING

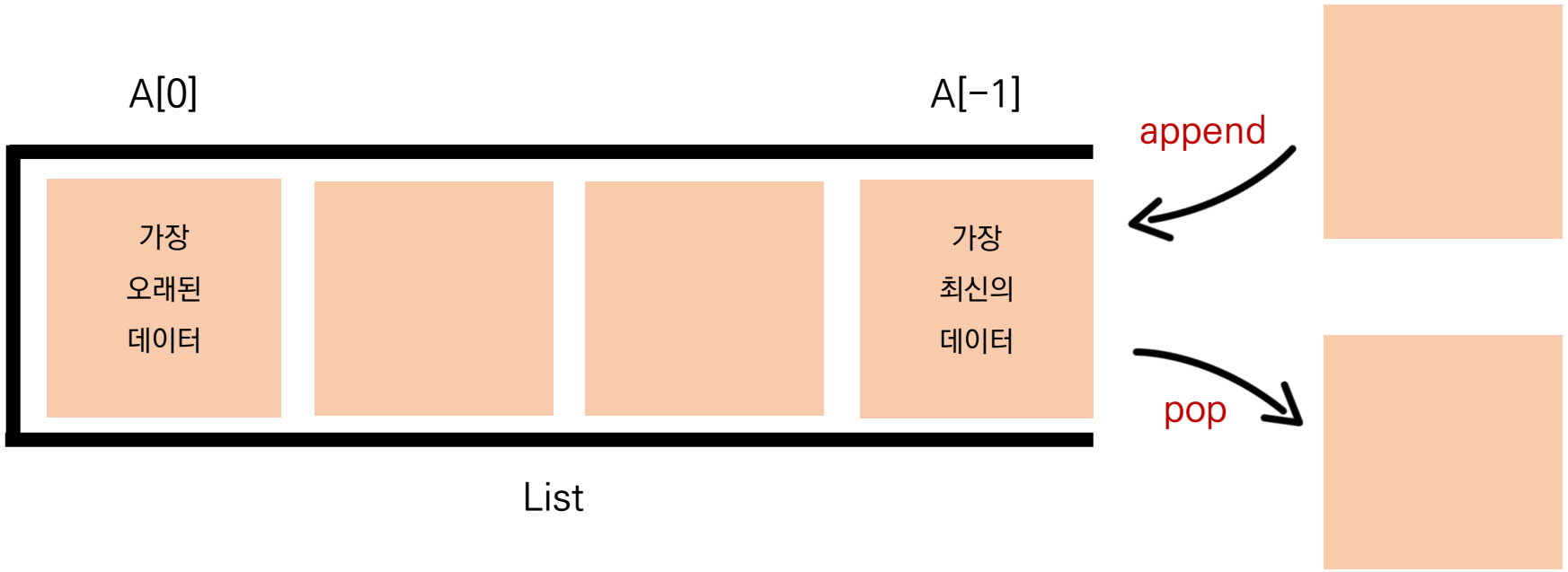
파이썬은 리스트(List)로 스택을 간편하게 사용할 수 있다 !



1. 스택 (Stack)

HAPPY HACKING

파이썬은 리스트(List)로 스택을 간편하게 사용할 수 있다 !



HAPPY HACKING

스택 연습

문제 번호	문제	링크
BOJ 10773	제로	https://www.acmicpc.net/problem/10773

HAPPY HACKING

‘BOJ 10773 제로’ 풀이

```python
stack = []

for _ in range(int(input())):
    number = int(input())

    if number == 0:
        stack.pop()
    else:
        stack.append(number)

print(sum(stack))
```

HAPPY HACKING

‘BOJ 10773 제로’ 풀이 – 1

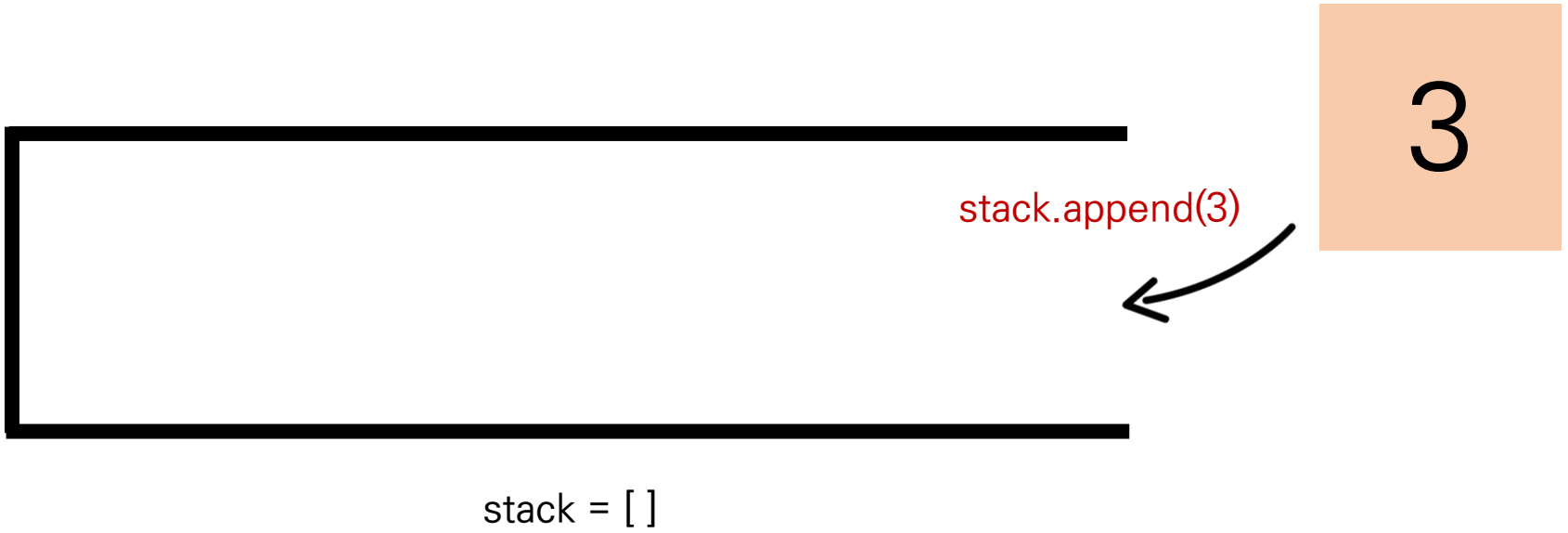
예제 입력 1 복사

```
4
3
0
4
0
```

stack = [ ]

HAPPY HACKING

‘BOJ 10773 제로’ 풀이 – 2



HAPPY HACKING

‘BOJ 10773 제로’ 풀이 – 3

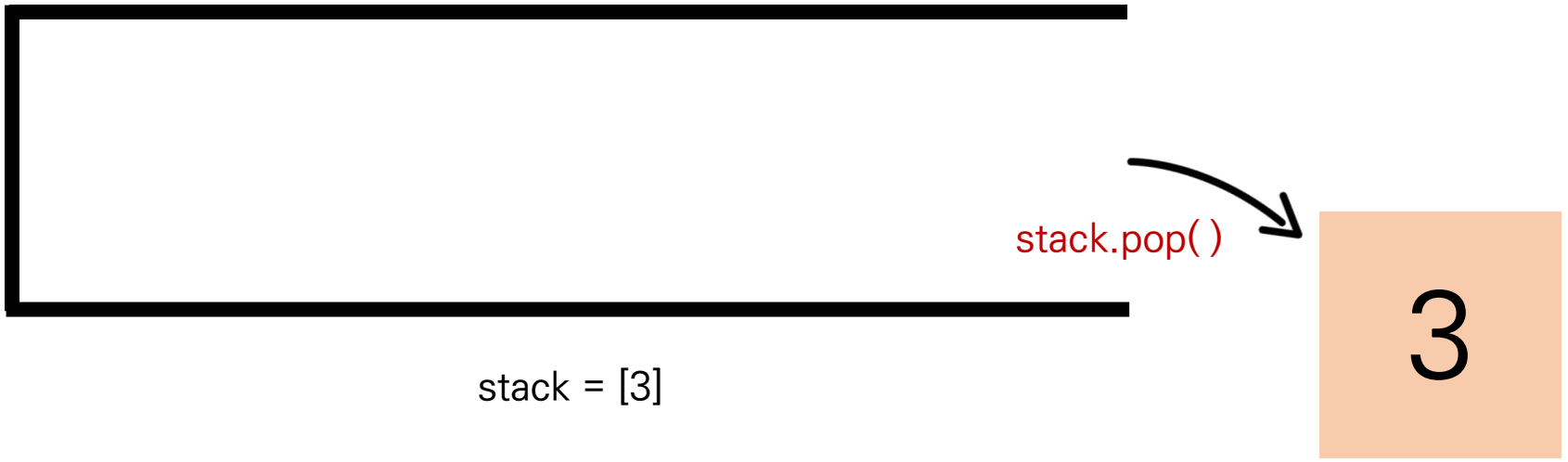
예제 입력 1 복사

```
4
3
0
4
0
```

stack = [3]

HAPPY HACKING

‘BOJ 10773 제로’ 풀이 – 4



HAPPY HACKING

‘BOJ 10773 제로’ 풀이 – 5

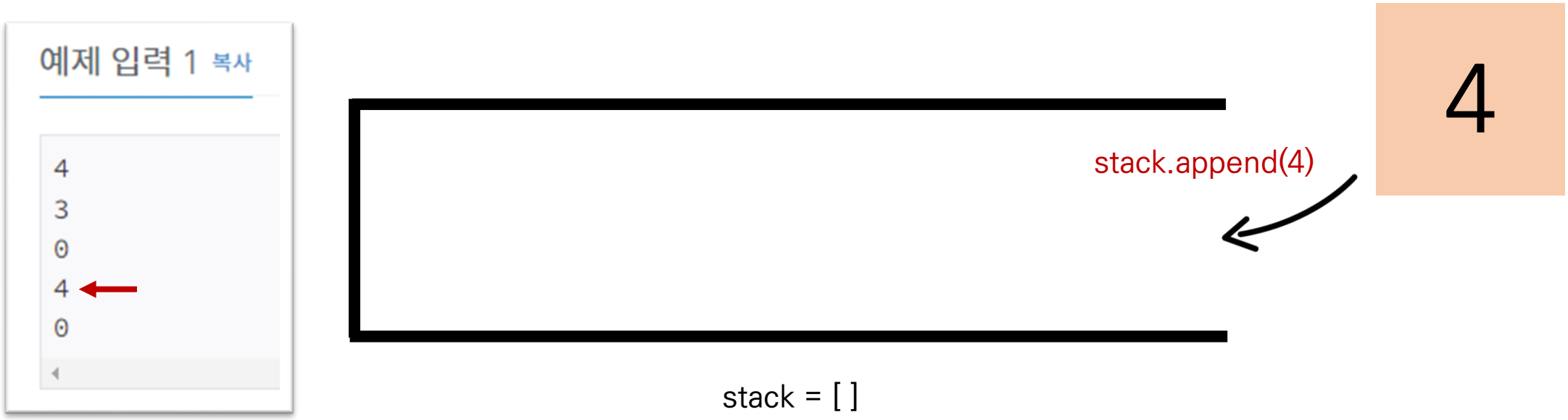
예제 입력 1 복사

```
4
3
0
4
0
```

stack = [ ]

HAPPY HACKING

‘BOJ 10773 제로’ 풀이 – 6



HAPPY HACKING

‘BOJ 10773 제로’ 풀이 – 7

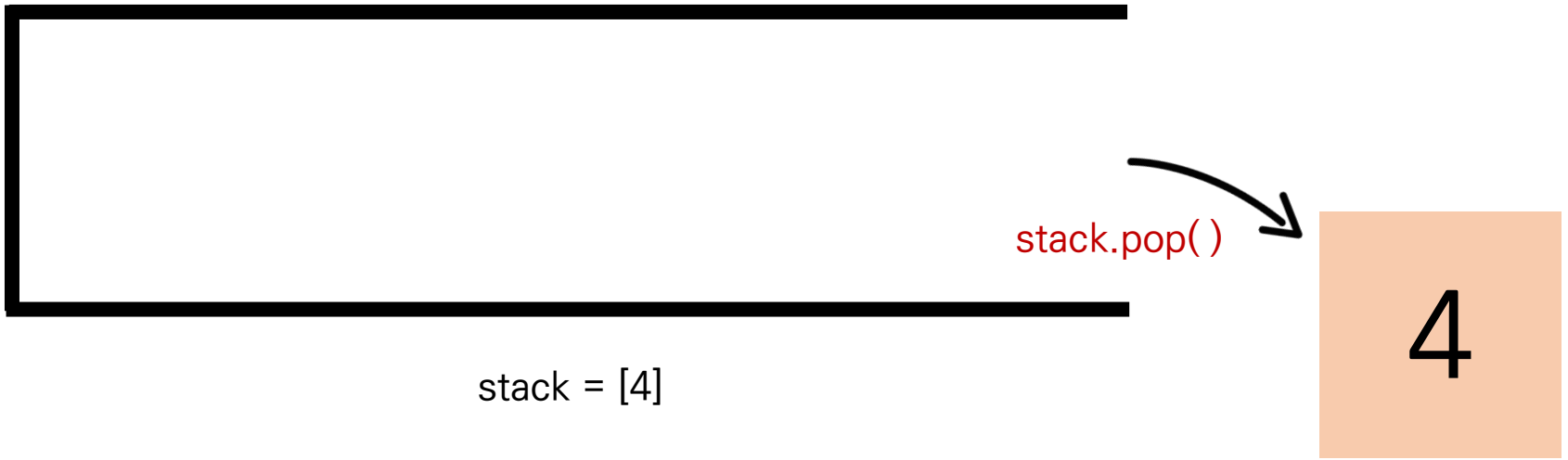
예제 입력 1 복사

```
4
3
0
4
0
```

stack = [4]

HAPPY HACKING

‘BOJ 10773 제로’ 풀이 – 8



HAPPY HACKING

‘BOJ 10773 제로’ 풀이 – 9

예제 입력 1 복사

```
4
3
0
4
0
```

stack = [ ]

HAPPY HACKING

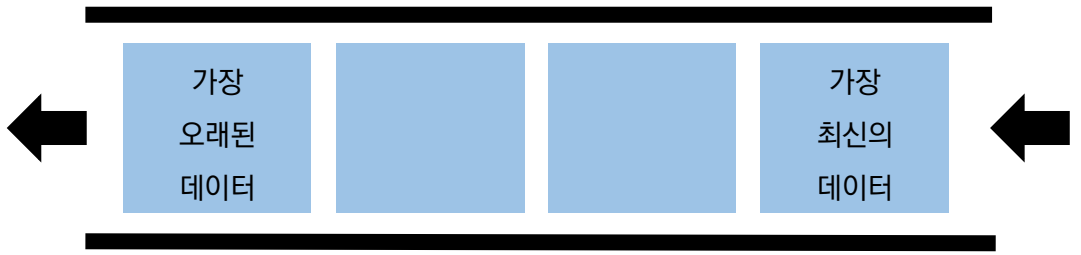
2. 큐 (Queue)

2. 큐 (Queue)

HAPPY HACKING

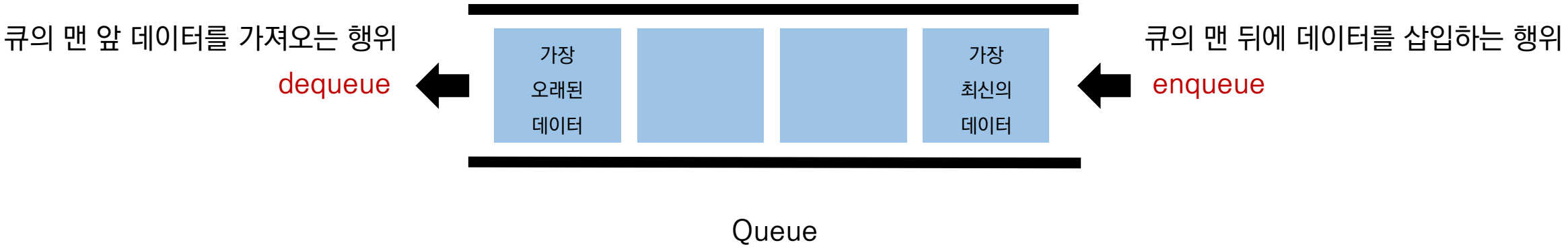
Queue는 한 쪽 끝에서 데이터를 넣고, 다른 한 쪽에서만 데이터를 뺄 수 있는 자료구조

가장 먼저 들어온 데이터가 가장 먼저 나가므로 FIFO(First-in First-out, 선입선출) 방식



큐의 동작 과정

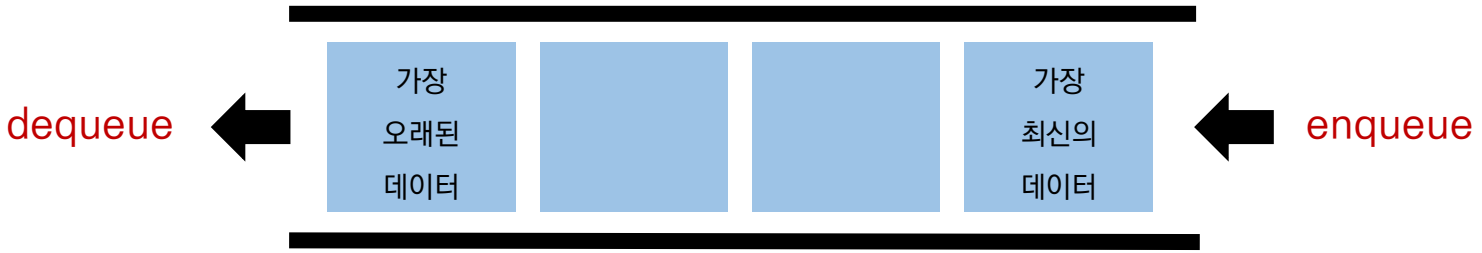
2. 큐 (Queue)


큐의 맨 앞 데이터를 가져오는 행위
dequeue
가장
오래된
데이터
가장
최신의
데이터
큐의 맨 뒤에 데이터를 삽입하는 행위
enqueue
Queue

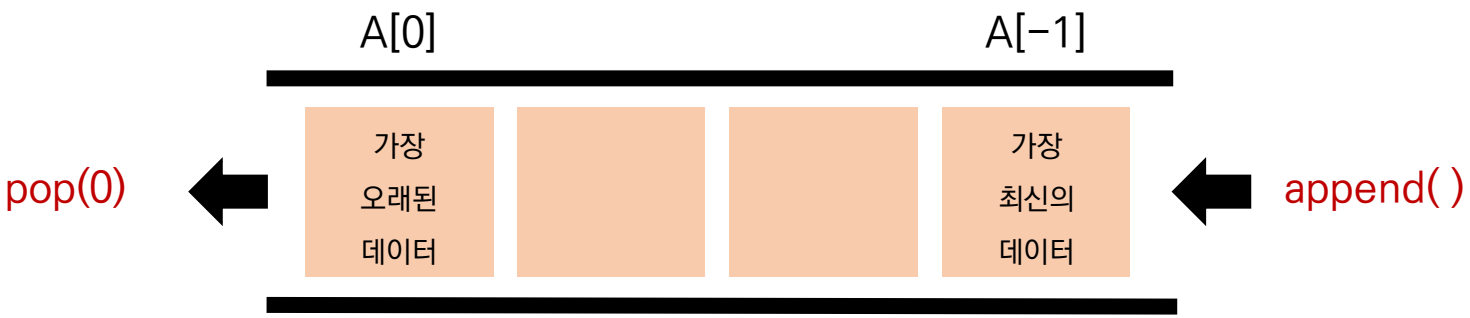
2. 큐 (Queue)

HAPPY HACKING

큐 자료구조도 파이썬 리스트(List)로 간편하게 사용할 수 있다 !



Queue



List

HAPPY HACKING

큐 연습

문제 번호	문제	링크
BOJ 2161	카드1	https://www.acmicpc.net/problem/2161

2. 큐 (Queue)

HAPPY HACKING

‘BOJ 2161 카드1’ 풀이

```python
n = int(input())
queue = list(range(1, n + 1))

while len(queue) > 1:
    print(queue.pop(0), end=" ")
    queue.append(queue.pop(0))

print(queue[0])
```

2. 큐 (Queue)

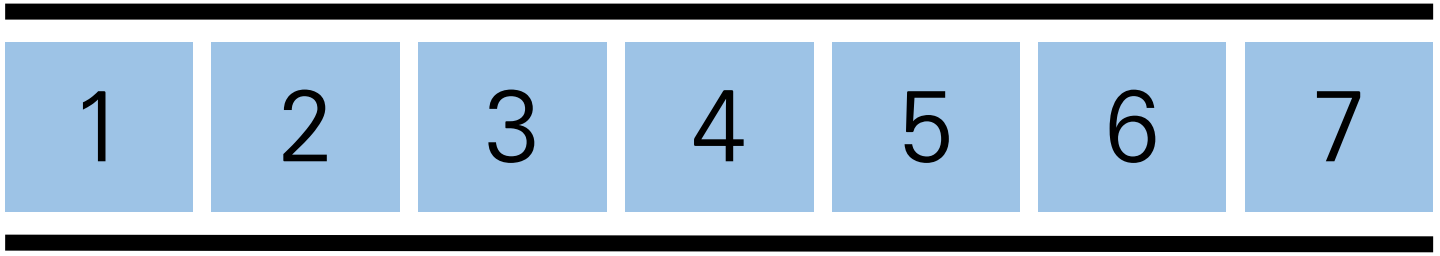
HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 1

예제 입력 1 복사

```
7
```

queue = list(range(1, n+1))



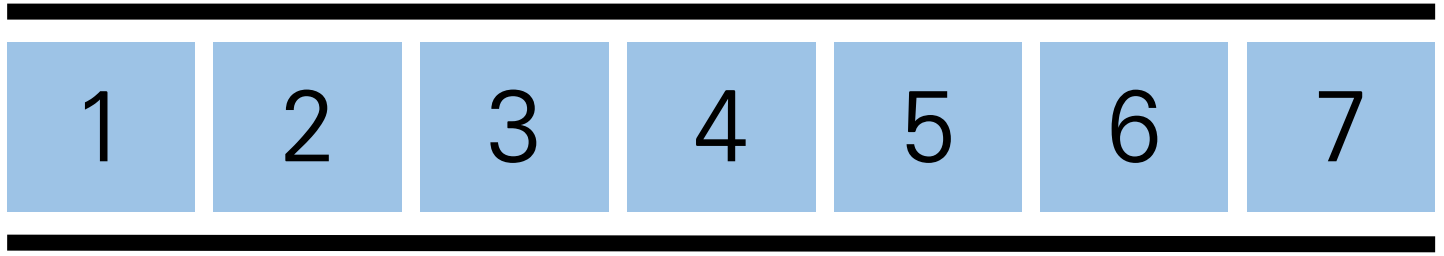
queue = [1, 2, 3, 4, 5, 6, 7]

HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 2

1) 제일 위에 있는 카드를 바닥에 버리기

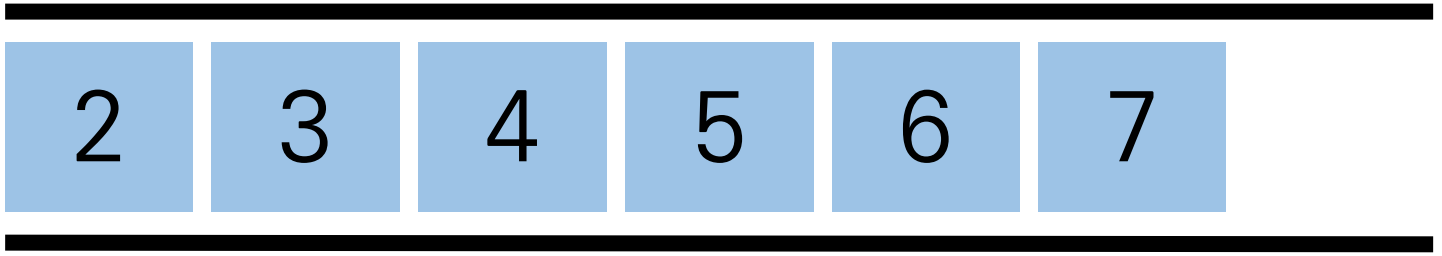
queue.pop(0)



queue = [1, 2, 3, 4, 5, 6, 7]

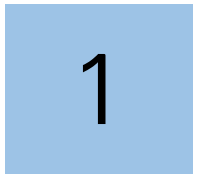
HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 3



queue = [2, 3, 4, 5, 6, 7]

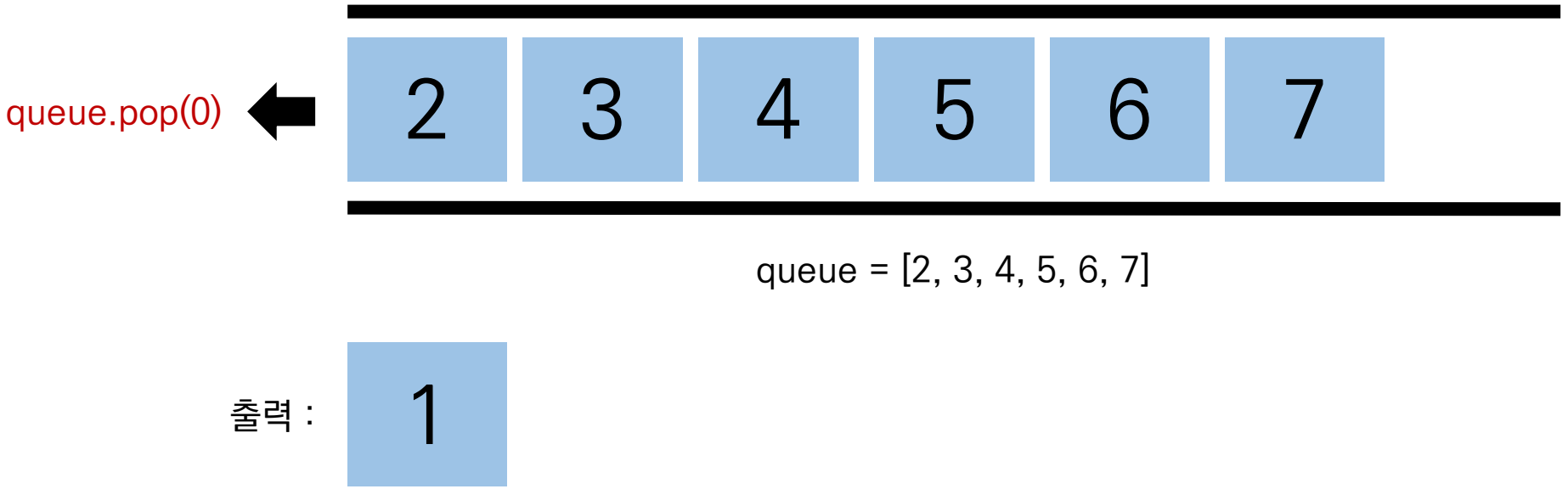
출력 :



HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 4

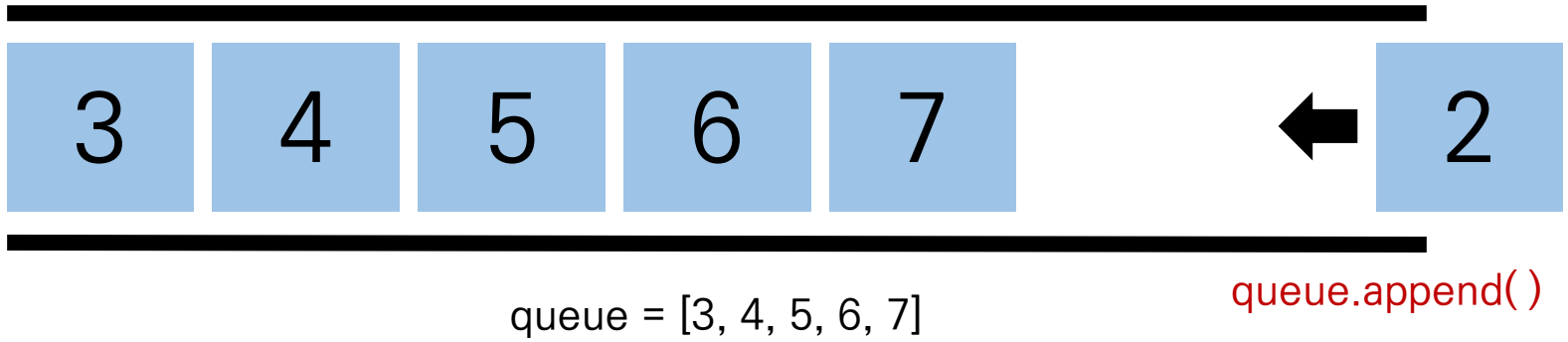
2) 제일 위에 있는 카드를 제일 아래에 있는 카드 밑으로 옮기기



HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 5

2) 제일 위에 있는 카드를 제일 아래에 있는 카드 밑으로 옮기기

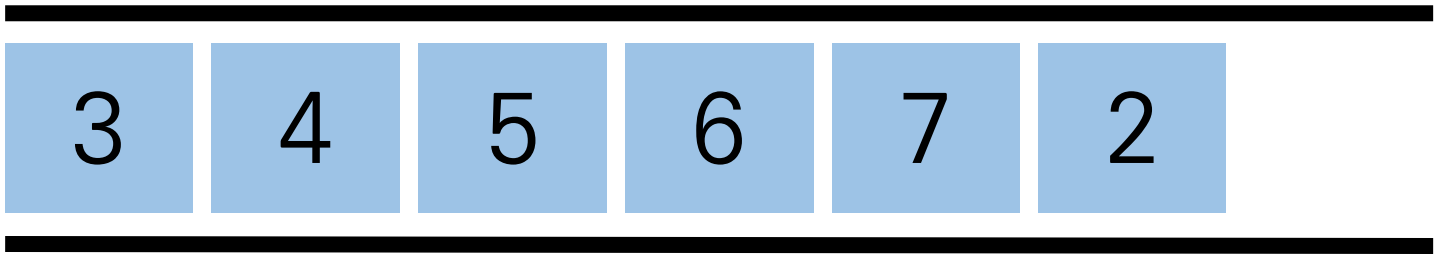


queue = [3, 4, 5, 6, 7]

출력 : 1

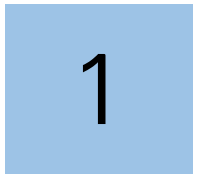
HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 6



queue = [3, 4, 5, 6, 7, 2]

출력 :

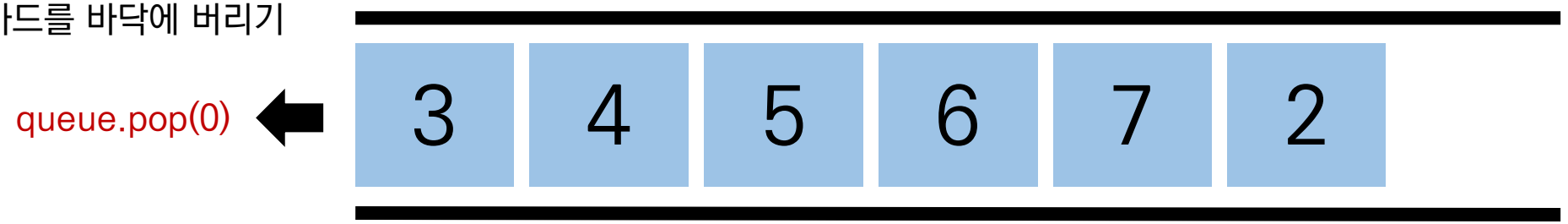


2. 큐 (Queue)

HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 7

1) 제일 위에 있는 카드를 바닥에 버리기

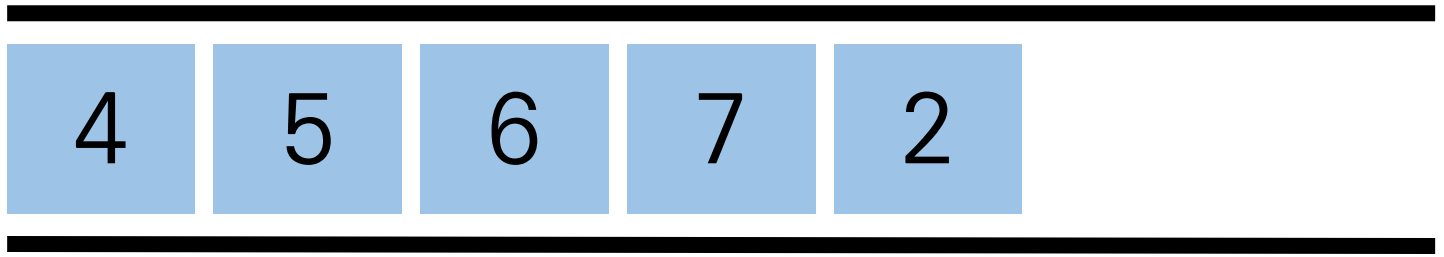


queue = [3, 4, 5, 6, 7, 2]

출력 :

HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 8



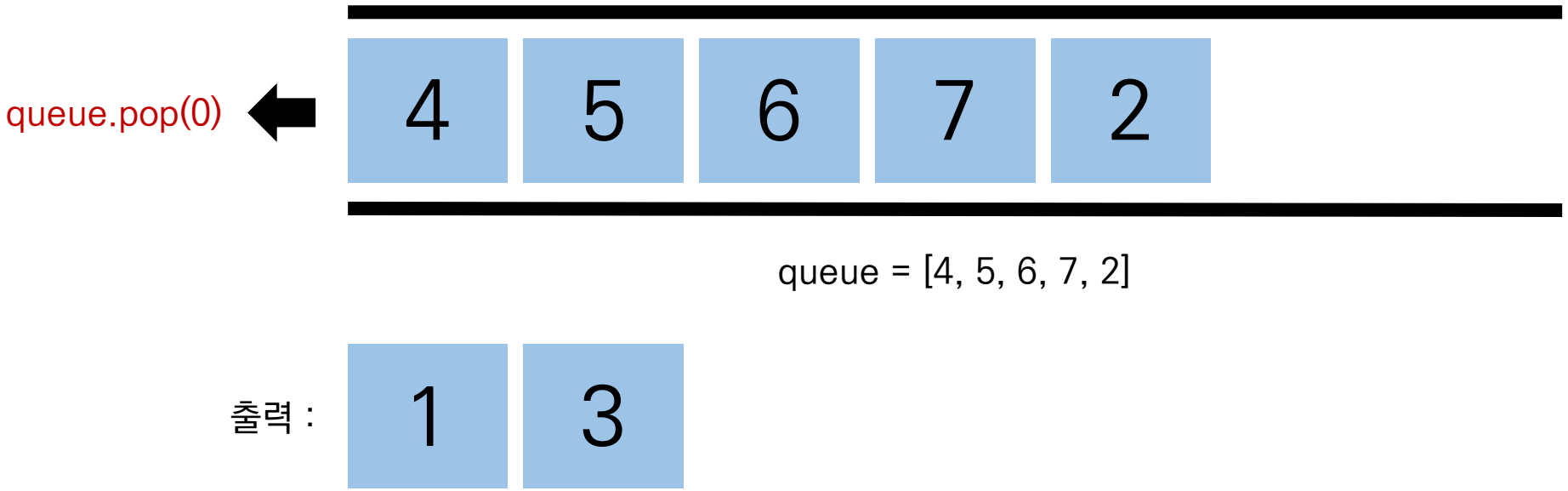
queue = [4, 5, 6, 7, 2]

출력 :

HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 9

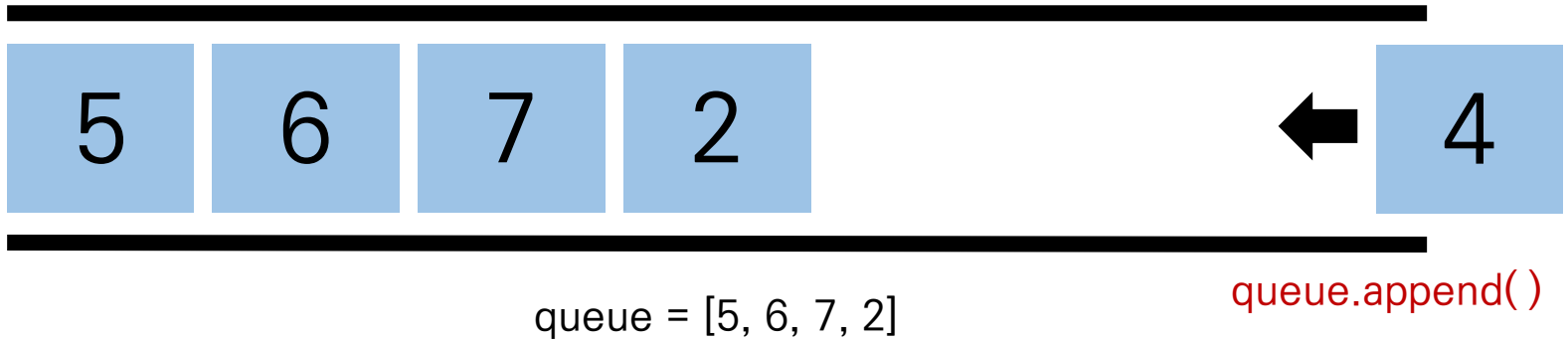
2) 제일 위에 있는 카드를 제일 아래에 있는 카드 밑으로 옮기기



HAPPY HACKING

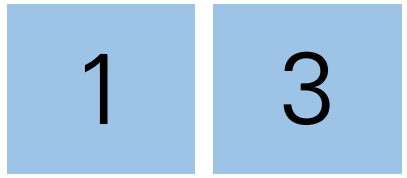
‘BOJ 2161 카드1’ 풀이 – 10

2) 제일 위에 있는 카드를 제일 아래에 있는 카드 밑으로 옮기기



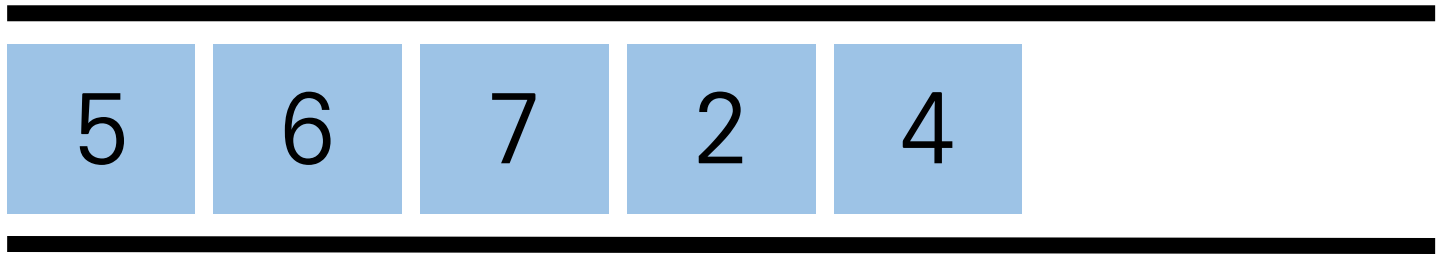
queue = [5, 6, 7, 2]

출력 :



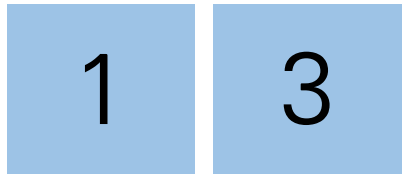
HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 11



queue = [5, 6, 7, 2, 4]

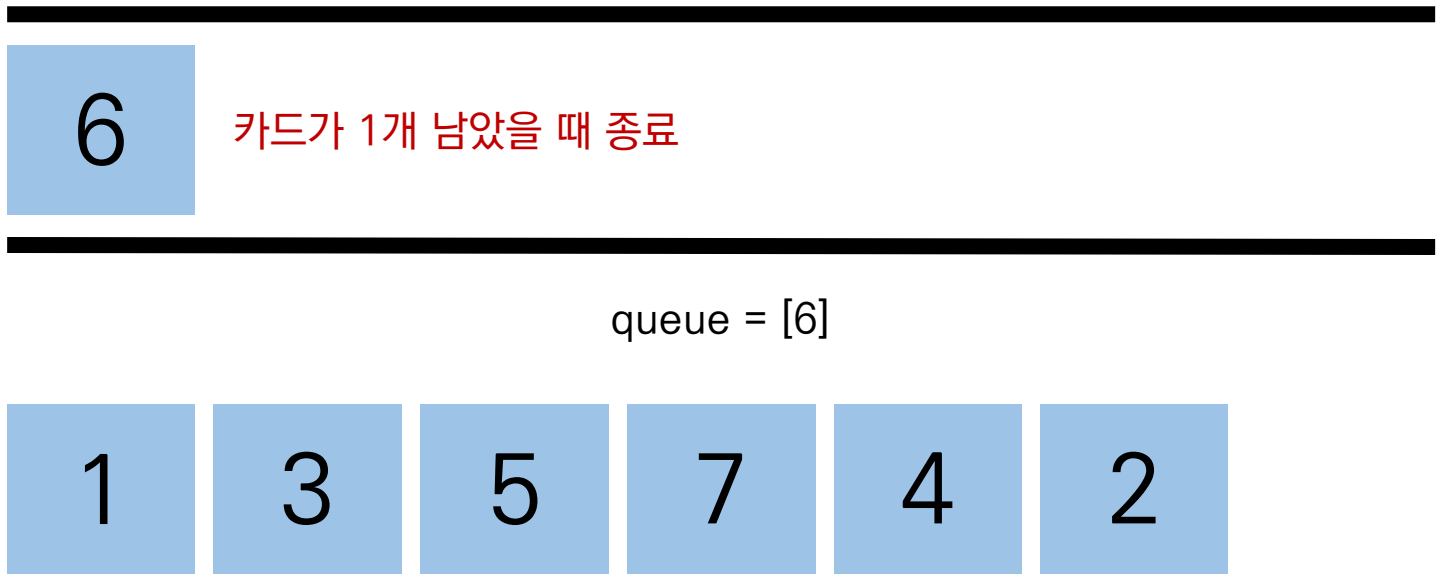
출력 :



이 과정을 계속 반복하다가 …

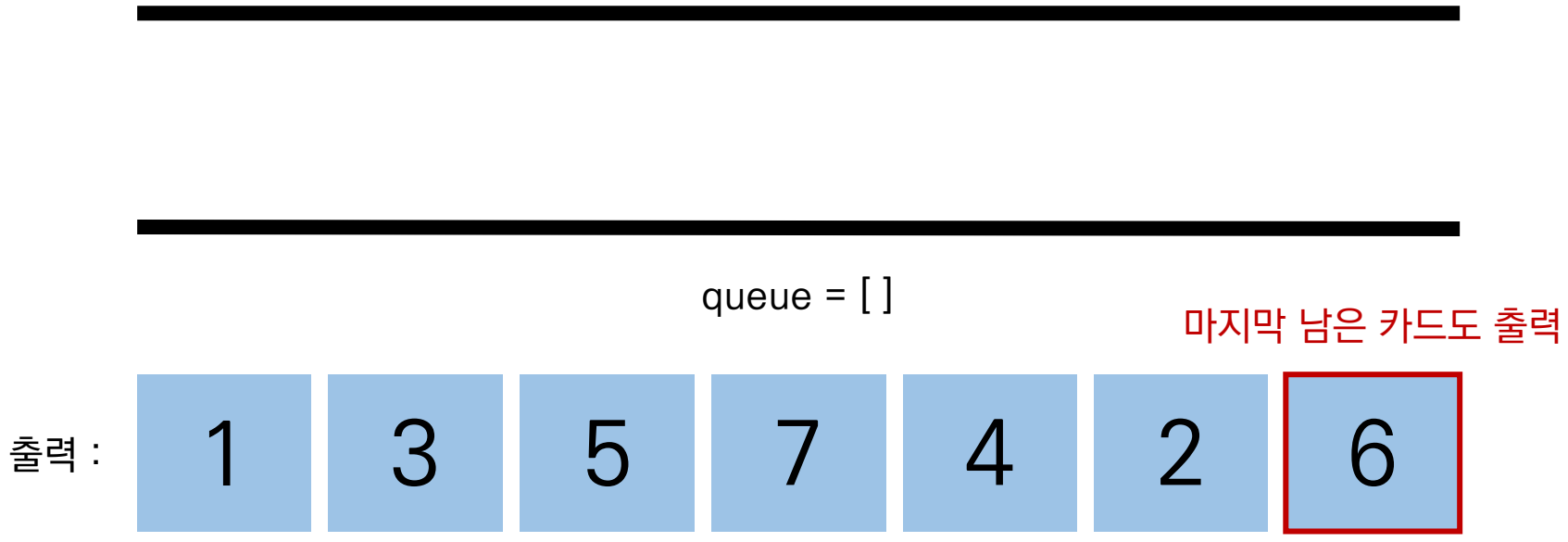
HAPPY HACKING

‘BOJ 2161 카드1’ 풀이 – 12



HAPPY HACKING

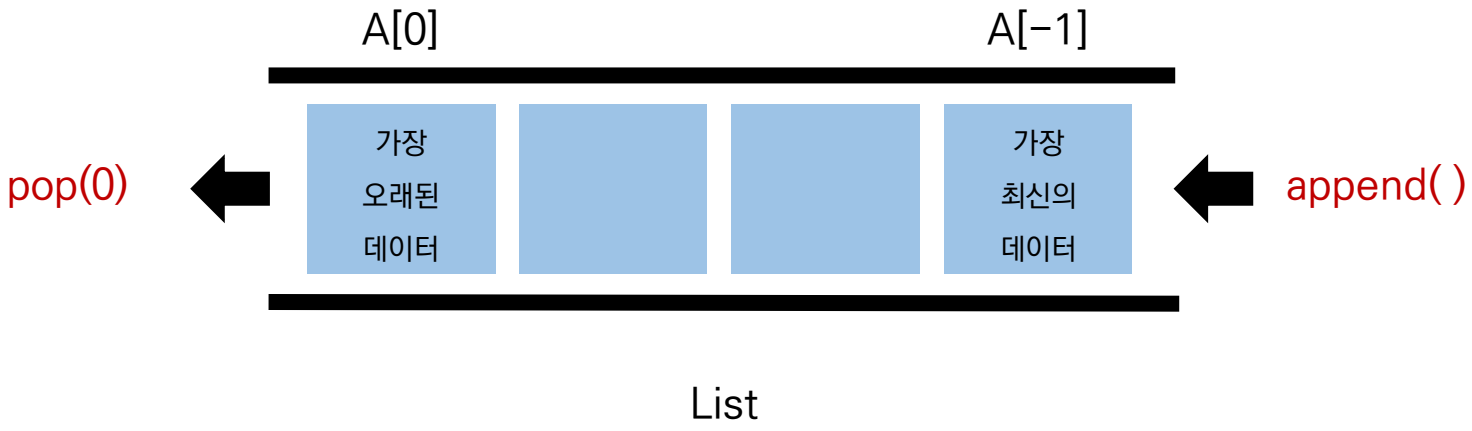
‘BOJ 2161 카드1’ 풀이 – 13



2. 큐 (Queue)

HAPPY HACKING

리스트를 이용한 큐 자료구조의 단점



데이터를 뺄 때 큐 안에 있는 데이터가 많은 경우 비효율적이다.

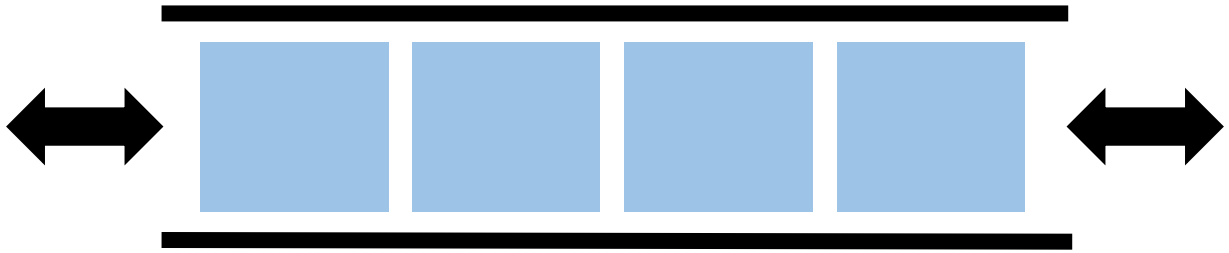
맨 앞 데이터가 빠지면서, 리스트의 인덱스가 하나씩 당겨 지기 때문이다 !

2. 큐 (Queue)

HAPPY HACKING

덱 (Deque, Double-Ended Queue) 자료구조

== 양 방향으로 삽입과 삭제가 자유로운 큐

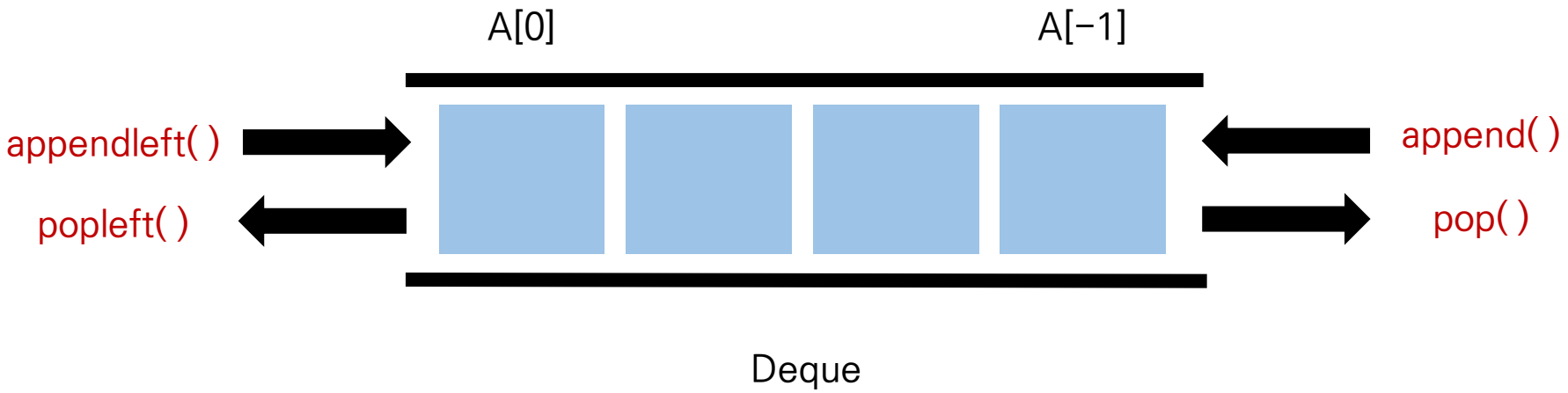


Deque

2. 큐 (Queue)

HAPPY HACKING

덱은 양 방향 삽입, 추출이 모두 큐보다 훨씬 빠르다.



따라서 데이터의 삽입, 추출이 많은 경우, 시간을 크게 단축 시킬 수 있다.

2. 큐 (Queue)

HAPPY HACKING

덱은 양 방향 삽입, 추출이 모두 큐보다 훨씬 빠르다.

수행 결과

```
list, 삽입 수행 시간, 0.5615091323852539 초
list, pop(0) 수행 시간, 4.844250679016113 초

deque, 삽입 수행 시간, 0.43166375160217285 초
deque, pop(0) 수행 시간, 0.0 초
```

1000개 요소를 삭제하는데 list는 4.8초가 걸리지만 덱은 0초에 가깝습니다.

출처: https://wellsw.tistory.com/122

2. 큐 (Queue)

HAPPY HACKING

‘BOJ 2161 카드1’ 큐와 덱 풀이 비교

```python
n = int(input())
queue = list(range(1, n + 1))

while len(queue) > 1:
    print(queue.pop(0), end=" ")
    queue.append(queue.pop(0))

print(queue[0])
```

큐를 이용한 풀이

```python
from collections import deque

n = int(input())
queue = deque(range(1, n + 1))

while len(queue) > 1:
    print(queue.popleft(), end=" ")
    queue.append(queue.popleft())

print(queue[0])
```

덱을 이용한 풀이