11 강

알고리즘과 자료구조

큐와 스택2

서울과학기술대학교 신일훈 교수

한국방송통신대학교 프리임칼리지

학습목표

1. 스택의 개념 및 연산을 이해한다.
2. 스택을 구현할 수 있다.
3. 스택을 활용한 괄호 매치 프로그램을 작성할 수 있다.

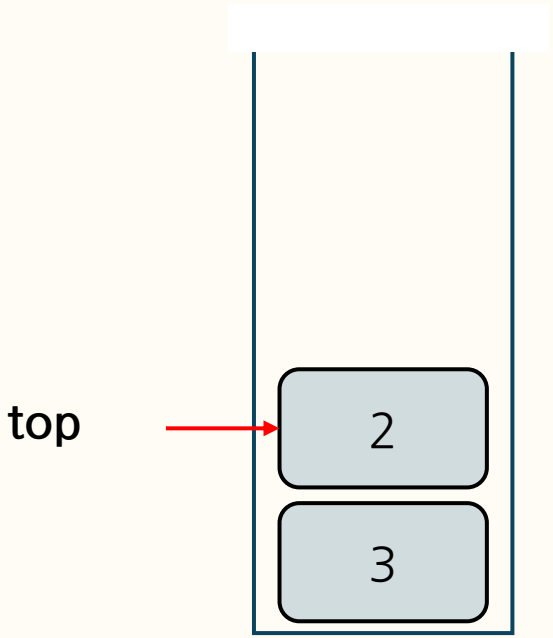
01

스택 개념

1. 스택(stack) 개념

❑ 개념

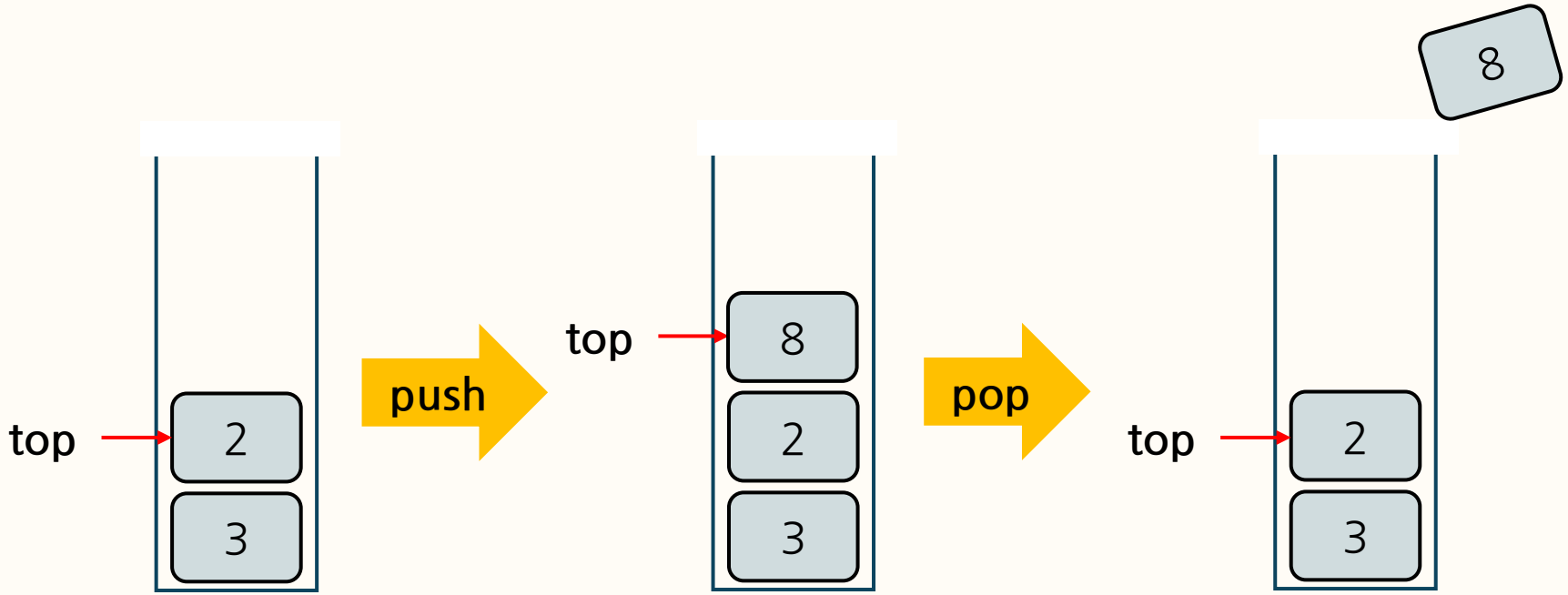
- 연결 리스트와 유사한 선형 자료구조
- 한쪽만 뚫려 있는 형태



1. 스택(stack) 개념

❑ 개념

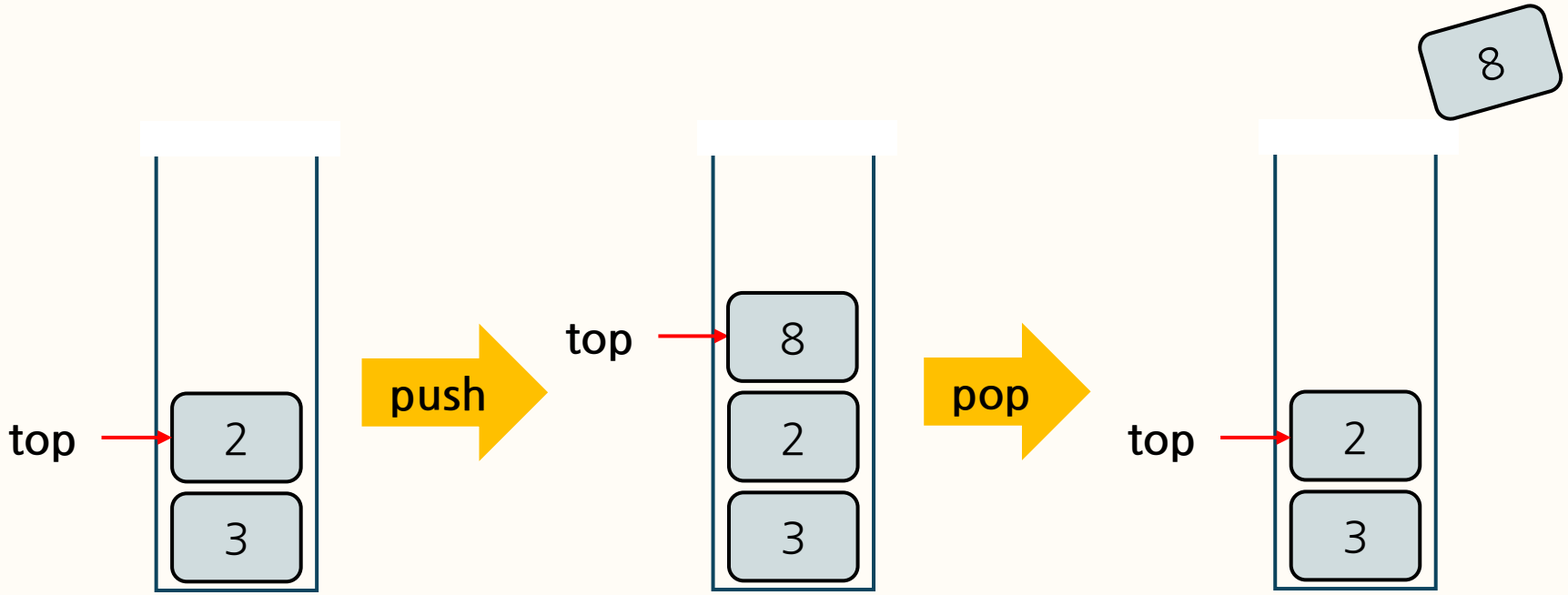
- 새로운 아이템의 추가는 스택의 전면으로만 가능
- 아이템의 제거도 스택의 전면으로만 가능



1. 스택(stack) 개념

❑ 특성

- LIFO(Last In First Out)
- FILO(First In Last Out)



1. 스택(stack) 개념

❑ 주요 연산

- 추가(push)
- 제거(pop)
- 검색
- 크기반환

1. 스택(stack) 개념

❑ 활용

- 함수호출정보 저장
- 그래프 탐색 (깊이우선탐색: death first search)
- 괄호 매치
- 회문검사
- …

02

스택 구현

2. 스택 구현

- 원형 양방향 연결리스트로 구현 가능
  - push() : insert_front()
  - pop() : delete_front()
  - …

2. 스택 구현

클래스 Stack 정의

- 멤버 변수
  - stack : 아이템들을 저장할 연결리스트 객체. 객체 생성 시에 빈 리스트로 초기화
  - count : stack에 저장된 아이템의 개수

2. 스택 구현

- 클래스 Stack 정의
  - 메서드
    - 생성자
    - push()
    - pop()
    - get_size()
    - print ()
    - search()

2. 스택 구현

클래스 Stack 정의 (생성자)

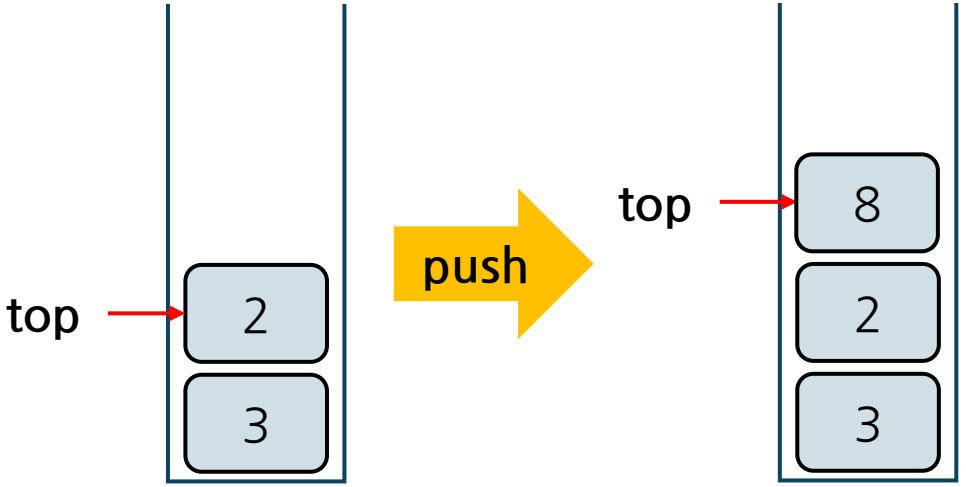
```
import CList


class Stack:
    def __init__(self):
        self.stack = CList.CList()
        self.count = 0
```

2. 스택 구현

클래스 Stack 정의 (push())

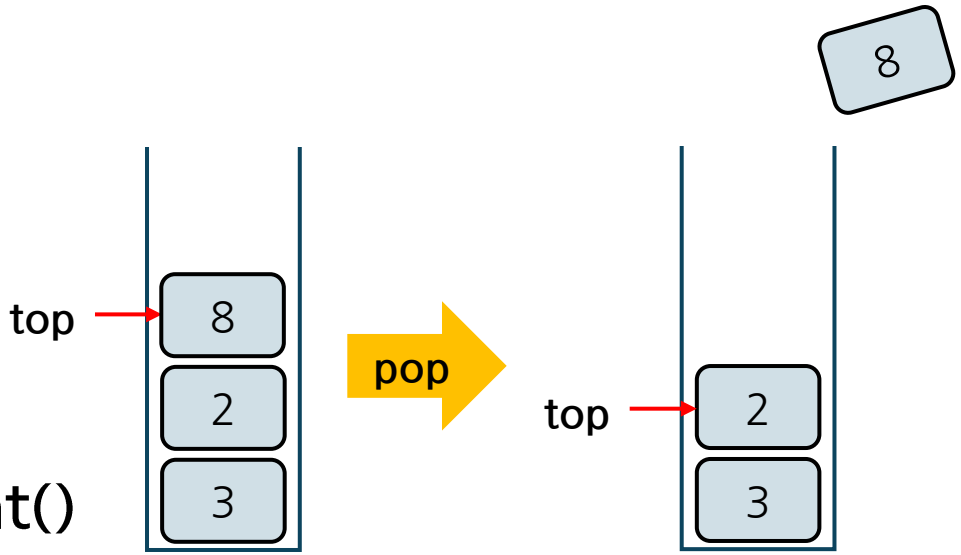
```
class Stack:
    def push(self, item):
        self.stack.insert_front(item)
        self.count += 1
```



2. 스택 구현

클래스 Stack 정의 (pop())

```
class Stack:
    def pop(self):
        if (self.count > 0):
            self.count -= 1
            item = self.stack.delete_front()
            return item
        return None
```



2. 스택 구현

클래스 Stack 정의 (print())

```
class Stack:
    def print(self):
        self.stack.print_list()
```

2. 스택 구현

클래스 Stack 정의 (get_size())

```
class Stack:
    def get_size(self):
        return self.count
```

2. 스택 구현

Stack 테스트

```
if __name__ == '__main__':
    s = Stack()

    s.push('mango')
    s.push('apple')
    s.push('orange')
    s.print()
```

2. 스택 구현

Stack 테스트

```
if __name__ == '__main__':
    ...

    for i in range(4):
        print(s.pop())

    s.push('apple')
    s.push('orange')
    s.print()
```

2. 스택 구현

Stack 테스트 실행

```
orange  <=>  apple  <=>  mango
orange
apple
mango
None
orange  <=>  apple
```

03

원형 양방향 연결 리스트 구현

3. 원형 양방향 연결 리스트 구현

- 클래스 CList 정의
  - 멤버 변수
    - head (리스트의 첫 노드를 가리킴)
  - 메서드
    - 생성자
    - insert_back()
    - delete_front()
    - print_list()
    - insert_front()

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (insert_front())

```
class CList:
    def insert_front(self, item):
        cnode = CNode(item, None, None)
        if (self.head == None):
            cnode.next = cnode
            cnode.prev = cnode
            self.head = cnode
```

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (insert_front())

```
class CList:
    def insert_front(self, item):
        ...
        else :
            first = self.head
            last = first.prev
            cnode.next = first
            cnode.prev = last
```

3. 원형 양방향 연결 리스트 구현

클래스 CList 정의 (insert_front())

```
class CList:
    def insert_front(self, item):
        ...
        else :
            ...
            first.prev = cnode
            last.next = cnode
            self.head = cnode
```

3. 원형 양방향 연결 리스트 구현

CList 테스트

```
if __name__ == '__main__':
    c = CList()
    c.insert_front('mango')
    c.insert_front('orange')
    c.insert_front('apple')
    c.print_list()
    print(c.delete_front())
    c.print_list()
    c.insert_front('banana')
    c.print_list()
```

3. 원형 양방향 연결 리스트 구현

Clist 테스트 실행

```
apple   <=>   orange   <=>   mango
apple
orange   <=>   mango
banana   <=>   orange   <=>   mango
```

04

스택 활용 프로그램 – 괄호 매치 검사

4. 스택 활용 프로그램 – 괄호 매치 검사

- 스택을 활용하여 괄호 일치 여부 검사하기
  - {{[()]}} : 일치
  - {{[])) : 일치하지 않음

4. 스택 활용 프로그램 – 괄호 매치 검사

- 스택을 활용하여 괄호 일치 여부 검사하기
  - {{[()]}} : 일치
  - {{[])) : 일치하지 않음

  - 아이디어
    - 오른쪽 괄호를 만나면 직전 왼쪽 괄호와 매치가 되어야 함.
      - {{[()

4. 스택 활용 프로그램 – 괄호 매치 검사

- 스택을 활용하여 괄호 일치 여부 검사하기
  - {{[()]}} : 일치
  - {{[])) : 일치하지 않음

  - 아이디어
    - 매치가 된 괄호들은 제거
      - {{[() => {{[
    - 이 과정을 반복하여 진행

4. 스택 활용 프로그램 – 괄호 매치 검사

- 스택을 활용하여 괄호 일치 여부 검사하기
  - {{[()]}} : 일치
  - 아이디어
    - {{[() => {{[=> {{[] => {{ => {{} => { => {} =>

4. 스택 활용 프로그램 – 괄호 매치 검사

❑ 스택을 활용하여 괄호 일치 여부 검사하기

- 알고리즘

1. 왼쪽 괄호 만나면 스택에 push
2. 오른쪽 괄호 만나면 스택에서 pop한 왼쪽 괄호와 매치하는지 비교
3. 매치하지 않으면 일치하지 않음.
4. 1 – 3 번 과정을 더 이상 괄호가 없을 때까지 반복
5. 만약 스택에 왼쪽 괄호가 남아 있으면 일치하지 않음.
6. 스택에 왼쪽 괄호가 남아 있지 않으면 모든 괄호가 일치함.

4. 스택 활용 프로그램 – 괄호 매치 검사

❑ 스택을 활용하여 괄호 일치 여부 검사하기

- 알고리즘

```
def check_parenthesis (string) :
    stack = Stack()
    for char in string :
        if (char is left_parenthesis) :
            stack.push(char)
        elif (char is right_parenthesis) :
            left = stack.pop()
            if (left is None or left not match char) :
                return False
```

4. 스택 활용 프로그램 – 괄호 매치 검사

- 스택을 활용하여 괄호 일치 여부 검사하기
  - 알고리즘

```
def check_parenthesis (string) :
    ...
    if (stack is not empty) :
        return False
    else :
        return True
```

4. 스택 활용 프로그램 – 괄호 매치 검사

- 스택을 활용하여 괄호 일치 여부 검사하기
  - 괄호 매치 여부 판단 방법
    - 괄호 매치 정보를 파이썬 dictionary를 이용하여 관리
      (key: 왼쪽 괄호, value: 오른쪽 괄호)

4. 스택 활용 프로그램 – 괄호 매치 검사

check_parenthesis() 정의

```
import Stack

def check_parenthesis(string) :
    s = Stack.Stack()
    parenthesis_dict = {
        '{':'}' ,
        '[':']',
        '(':')'
    }
```

4. 스택 활용 프로그램 – 괄호 매치 검사

check_parenthesis() 정의

```
def check_parenthesis(string) :
    ...
    left_parenthesis = parenthesis_dict.keys()
    right_parenthesis = parenthesis_dict.values()
```

4. 스택 활용 프로그램 – 괄호 매치 검사

check_parenthesis() 정의

```
def check_parenthesis(string) :
    ...
    for char in string:
        if char in left_parenthesis :
            s.push(char)
        elif char in right_parenthesis :
            left = s.pop()
            if (left == None   or   parenthesis_dict[left] != char) :
                return False
```

4. 스택 활용 프로그램 – 괄호 매치 검사

check_parenthesis() 정의

```
def check_parenthesis(string) :
    ...
    if s.get_size() > 0 :
        return False
    else :
        return True
```

4. 스택 활용 프로그램 – 괄호 매치 검사

check_parenthesis() 테스트

```
if __name__ == '__main__':
    print(check_parenthesis('{a[c]b}'))
    print(check_parenthesis('{a[c]b}}'))
    print(check_parenthesis('{[a[c]b}'))
    print(check_parenthesis('{a[c]b)'))
```

4. 스택 활용 프로그램 – 괄호 매치 검사

check_parenthesis() 실행

```
True
False
False
False
```

05

회문 프로그램

5. 회문 프로그램

❑ 스택을 활용한 회문 검사

- 회문: 좌우 대칭 문자열
  - aba, aa, a, aaccaa, …

5. 회문 프로그램

❑ 스택을 활용한 회문 검사

- 아이디어
  - 문자의 개수가 짝수인 경우:
    - 왼쪽 절반의 문자들을 차례대로 스택에 push
    - 오른쪽 절반의 문자(right)들에 대해 다음을 반복
      - left = stack.pop()
      - if (left != right)
        - return False
    - return True

정리하기

- 스택의 개념
- 스택의 구현
- 원형 양방향 연결 리스트 구현
- 스택의 활용 – 괄호 매치 검사

12 강

다음 시간 안내 ▶▶▶

해시테이블