

Data Structure

자료구조



스택



한국기술교육대학교
온라인평생교육원

학습내용

- 스택의 개념과 동작 원리
- 배열 구조의 스택 구현
- 괄호 검사 문제와 스택

학습목표

- 스택의 개념과 동작 원리를 설명할 수 있다.
- 배열 구조로 스택을 구현할 수 있다.
- 스택을 이용해 괄호 검사 문제를 해결할 수 있다.



스택의 개념과 동작 원리

스택

스택의 개념과 동작 원리

1 스택의 개념과 동작 원리

스택(Stack)

- “Stack”: 쌓아놓은 더미 또는 무더기
- 선형 자료구조의 일종
- **자료의 입출력이 후입 선출로 일어남**

- ▶ 가장 나중에 들어온(Last-In) 데이터가 가장 먼저 나감(First-Out)
- ▶ Last-In First-Out (LIFO)
- ▶ 입력과 역순의 출력이 필요한 다양한 응용에 사용됨

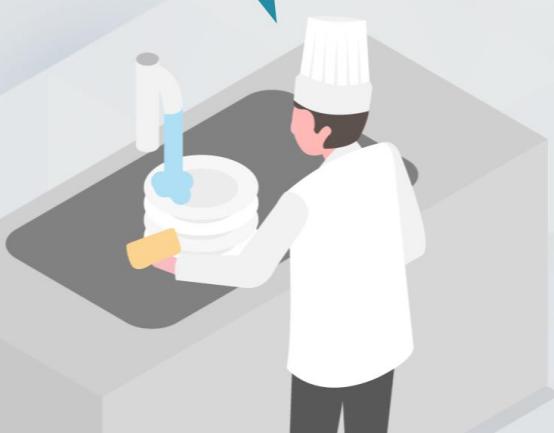
스택

스택의 개념과 동작 원리

1 스택의 개념과 동작 원리

● 스택의 예: 음식점 주방의 접시 더미

닦은 접시는
맨 위에 그냥 쌓는 것이
제일 편해요.



그건 나도 그래. 바쁘니까
그냥 맨 위에 있는 접시를 꺼내
음식을 담아야지.





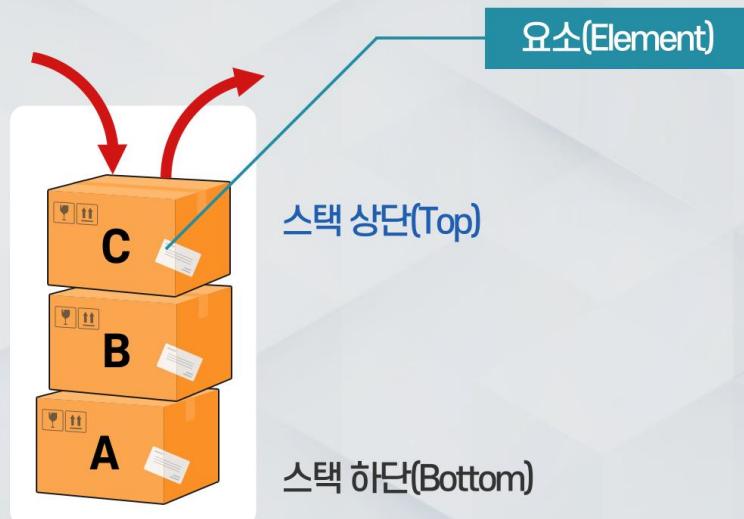
스택의 개념과 동작 원리

스택

스택의 개념과 동작 원리

1 스택의 개념과 동작 원리

● 스택의 구조



스택

스택의 개념과 동작 원리

1 스택의 개념과 동작 원리

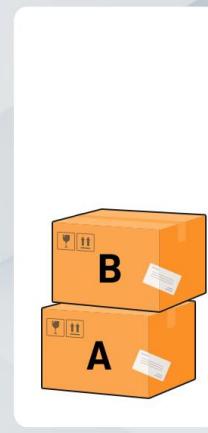
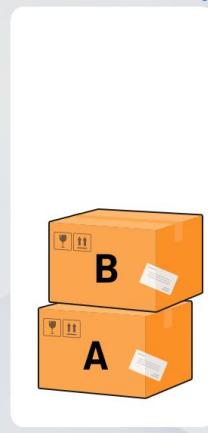
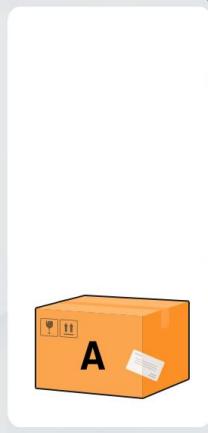
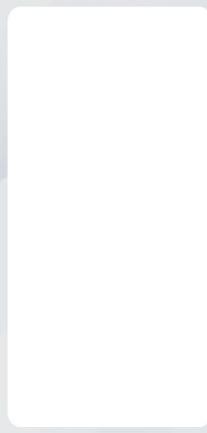
● 스택의 연산

push(A)

push(B)

push(C)

pop()





스택의 개념과 동작 원리

스택

스택의 개념과 동작 원리

2 스택의 추상 자료형

● 추상 자료형 종류

데이터	후입 선출(LIFO)의 접근 방법을 유지하는 항목들의 모음
Stack()	비어 있는 새로운 스택을 만듦
isEmpty()	스택이 공백상태이면 True를 아니면 False를 반환
isFull()	스택이 포화상태이면 True를 아니면 False를 반환
push(e)	항목 e를 스택의 맨 위에 추가
pop()	스택의 상단에 있는 항목을 꺼내 반환

스택

스택의 개념과 동작 원리

2 스택의 추상 자료형

● 추상 자료형 종류

peek()	스택의 상단에 있는 항목을 삭제하지 않고 반환
size()	모든 항목들의 개수를 반환
clear()	스택을 공백상태로 만듦
display()	스택을 화면에 보기 좋게 출력

리스트에 비해 연산들이 단순함

스택의 개념과 동작 원리



스택

스택의 개념과 동작 원리

3 스택의 용도

- ▶ 입력과 역순의 출력이 필요한 경우에 사용

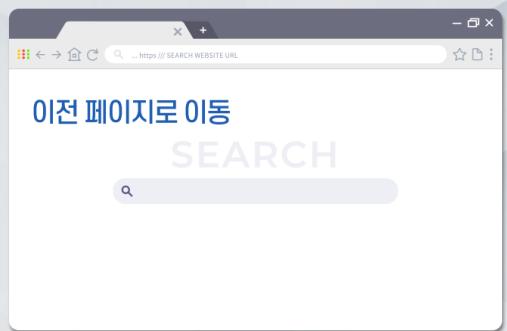
다양한 편집기에서 되돌리기(undo) 가능

웹 브라우저에서 '이전 페이지'로 돌아가기

암호 검사

계산기: 후위 표기식 계산, 중위 표기식의
후위 표기식 변환

미로 탐색



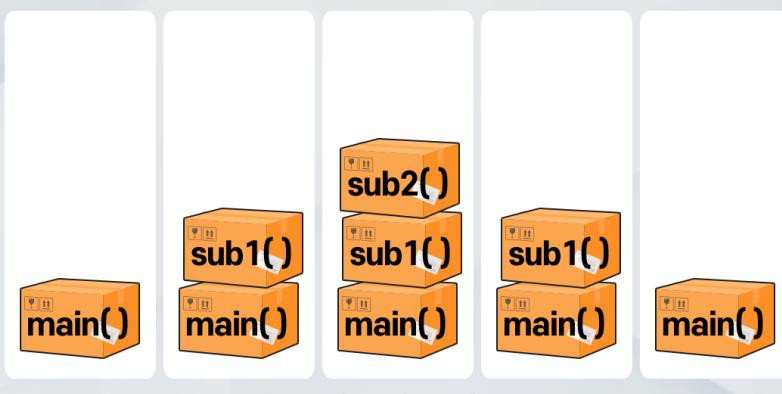
스택

스택의 개념과 동작 원리

3 스택의 용도

● 시스템 스택

- ▶ 함수 호출에서 복귀 주소 저장하는 장소
- ▶ 활성화 레코드가 저장



시스템 스택

```

1 int main()
{
    int i = 3;
20    sub1(i)
    ...
}

100 int sub1(int a)
{
    int j = 5;
150    sub2(j);
    ...
}

200 void sub2(int b)
{
    ...
}

```

배열 구조의 스택 구현



스택

배열 구조의 스택 구현

1 스택의 데이터

● 스택의 구현 방법

배열 구조

연결된 구조

▶ 배열 구조의 스택을 클래스로 구현

배열

파이썬의 리스트 이용

배열의 크기

스택의 용량 고정

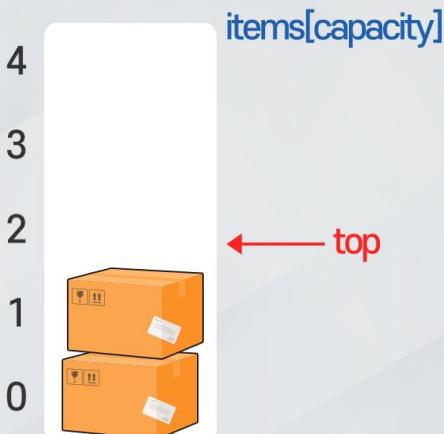
▶ 스택을 생성할 때 스택의 크기를 결정하도록 구현

스택

배열 구조의 스택 구현

1 스택의 데이터

● 배열 구조의 스택 설계



<배열을 이용한 스택의 설계>

① 배열의 이름: items

- 스택에 저장할 항목 보관
- 배열 구조 (파이썬 리스트 이용)

② 스택의 최대 용량: capacity

- 최대 저장 가능한 항목의 수를 초기에 설정

③ 스택 상단의 위치: top

- 현재 저장된 항목의 수와 동일



배열 구조의 스택 구현

스택

배열 구조의 스택 구현

1 스택의 데이터

● 클래스 정의와 생성자

스택의 데이터 멤버

스택에 저장할 자료	스택 상단을 위한 변수
<p>items</p> <ul style="list-style-type: none"> 1차원 배열에 저장 파이썬 리스트 사용 <p>capacity</p> <ul style="list-style-type: none"> 스택의 용량을 처음에 결정 	<p>top</p> <ul style="list-style-type: none"> 가장 최근에 입력된 자료의 다음 위치 기록 가장 먼저 들어온 항목: <code>items[0]</code> 가장 최근에 들어온 항목: <code>items[top-1]</code>

스택

배열 구조의 스택 구현

1 스택의 데이터

● 클래스 정의와 생성자

▶ 생성자에서 데이터 멤버 정의 및 초기화 권장

items[]	스택 항목들을 저장하는 배열
capacity	스택의 최대 용량
top	스택 상단 (가장 최근 입력 자료의 다음 위치)

- 가장 최근 입력 자료의 **다음 위치** 기록
- 가장 먼저 들어온 항목: `items[0]`
- 가장 최근에 들어온 항목: `items[top-1]`



배열 구조의 스택 구현

1 스택의 데이터

● 클래스 정의와 생성자

```
class Stack :
    def __init__( self, capacity = 10 );
        self.capacity = capacity
        self.items = [None]*self.capacity
        self.top = 0
    # 생성자 정의
    # 스택의 용량
    # 객체(데이터): 인스턴스 변수
    # 스택 상단
```

2 스택의 연산

- 스택 ADT의 연산들을 Stack class의 멤버 함수로 추가
- 스택의 크기, 초기화 연산

```
# Stack class의 멤버 함수이므로 들여쓰기
def size( self ):
    return self.top
# 스택의 크기 반환
# top은 스택의 크기(항목의 수)와 같음

def clear( self ):
    self.top = 0
# 스택을 초기화
# top만 0으로 초기화 → 스택 초기화 됨
```

배열 구조의 스택 구현



스택

배열 구조의 스택 구현

2 스택의 연산

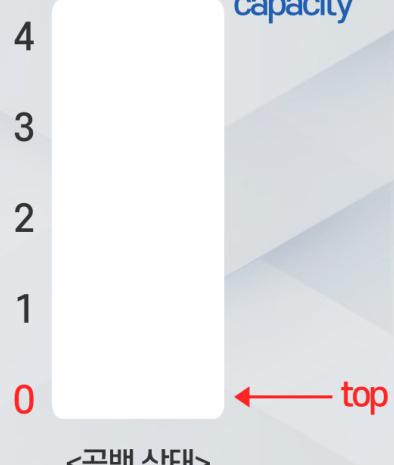
● 공백 상태 검사 (isEmpty 함수)

▶ 공백 상태: top이 0을 가리킴

```
def isEmpty( self ):
    if self.top == 0:
        return True
    else :
        return False
```

▶ 동일한 코드

```
def isEmpty( self ):
    return self.top == 0
# self.top == 0이면 True, self.top != 0이면 False 반환
```



스택

배열 구조의 스택 구현

2 스택의 연산

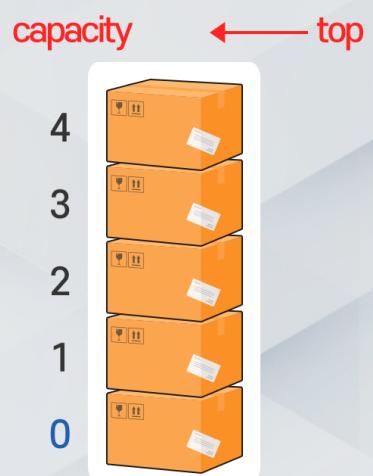
● 포화 상태 검사 (isFull 함수)

▶ 포화 상태: top은 capacity

```
def isFull( self ):
    if self.top == self.capacity:
        return True
    else:
        return False
```

▶ 동일한 코드

```
def isFull( self ):
    return self.top == self.capacity
# bool 함수(참, 거짓만 반환)인 경우 위와 같이 간결한
# 코드 사용 가능
```



capacity = 5
포화상태 → top = 5

배열 구조의 스택 구현

스택

배열 구조의 스택 구현

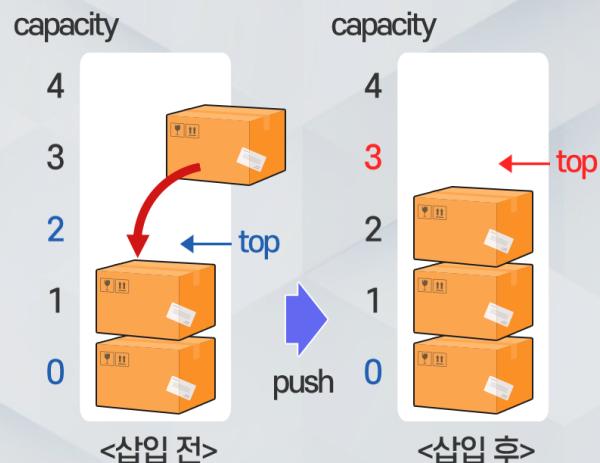
2 스택의 연산

◎ push 연산 (스택의 삽입 연산)

- 반드시 포화 상태 검사 필요(스택이 삽입 가능한 상태인지 확인)
: Overflow(스택이 꽉 차있는데 넣으려는 상태) 여부

- 삽입 후 top 한 칸 증가함

```
def push( self, item ):
    if not self.isEmpty():
        self.items[self.top] = item
        self.top += 1
    else:
        print( " overflow " )
        exit()
```



스택

배열 구조의 스택 구현

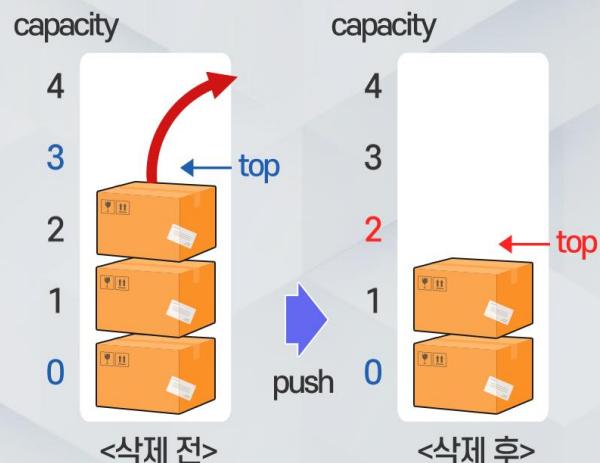
2 스택의 연산

◎ pop 연산 (스택의 삭제 연산, 가장 최근에 입력된 항목을 꺼내는 연산)

- 반드시 공백 상태 검사 필요(스택이 삭제 가능한 상태인지 확인)
: Underflow(스택이 비었는데 꺼내려는 상태) 여부

- 삭제 후 top 한 칸 감소

```
def pop( self ):
    if not self.isEmpty():
        self.top -= 1
        return self.items[self.top]
    else:
        print( " underflow " )
        exit()
```



배열 구조의 스택 구현



스택

배열 구조의 스택 구현

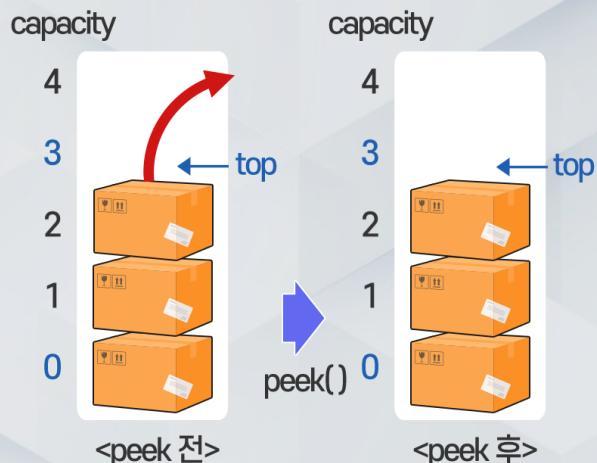
2 스택의 연산

- peek 연산: (상단의 항목을 단순 반환)

▶ 반드시 공백 상태 검사 필요: Underflow 여부

```
def peek( self ):
    if not self.isEmpty():
        return self.items[self.top-1]

    else:
        print("underflow")
        exit()
```



스택

배열 구조의 스택 구현

2 스택의 연산

- 테스트 프로그램

▶ 홀수와 짝수를 위한 스택

```
if __name__ == "__main__":
    odd = Stack()
    even = Stack()
    for i in range(10):    # i = 0, 1, 2, ..., 9
        if i%2 == 0 : even.push(i)
        else : odd.push(i)
    even.display(' 스택 even push 5회: ')
    odd.display (' 스택 odd push 5회: ')
    print(' 스택 even     peek: ', even.peek())
    print(' 스택 odd     peek: ', odd.peek())

    for _ in range(2) : even.pop()
    for _ in range(3) : odd.pop()
    even.display(' 스택 even pop 2회: ')
    odd.display (' 스택 odd pop 3회: ')

# 짝수 스택 내용 출력
# 홀수 스택 내용 출력
```

```
C:\WINDOWS\system32\cmd.exe
스택 even push 5회: = [0, 2, 4, 6, 8, ]
스택 odd  push 5회: = [1, 3, 5, 7, 9, ]
스택 even     peek: 8
스택 odd     peek: 9
스택 even pop 2회: = [0, 2, 4, ]
스택 odd  pop 3회: = [1, 3, ]
```

배열 구조의 스택 구현



파일(F) 편집(E) 보기(V) 프로젝트(P) 디버그(O) 태스크(S) 브레이크(B) 도구(T) 확장(X) 찾(W) 도움말(H) 검색(Ctrl+Q) 교재-파이썬

설명서 험세기 - 줄다 보기

stackClass.py

```

1 # 자료구조-7회차 : stackClass.py
2 #
3 # [7-2] ArrayList 클래스 구현
4 #   - 고정 크기의 스택(용량이 미리 정해진 리스트 사용)
5 =====
6
7 class Stack :
8     def __init__( self, capacity = 10 ):      # 생성자 정의
9         self.capacity = capacity             # 스택의 용량
10        self.items = [None]*self.capacity    # 객체(데이터) : 인스턴스 변수
11        self.top = 0                         # 스택 상단
12
13    def size( self ): return self.top
14    def clear( self ): self.top = 0
15
16    def isEmpty( self ): return self.top == 0
17    def isFull( self ): return self.top == self.capacity
18
19    def push( self, item ):
20        if not self.isFull():
21            self.items[self.top] = item
22            self.top += 1

```

줄다 보기 선택(S)

오류 목록: 경계 정 출처

실습 단계

- class 정의
- 삽입 연산
- 삭제 연산
- peek() 연산
- display 연산
- 테스트 코드(코드가 직접 실행될 때만 테스트 코드 실행)
- 소스 코드 실행 결과

괄호 검사 문제와 스택



스택

괄호 검사 문제와 스택

1 괄호 검사의 개념

괄호 검사

- 수식 표기, 프로그래밍 언어, HTML 문서 등 다양한 분야에서 사용
- 괄호의 종류: 대중소(' ', '), ('{', '}'), ('(, ') 등
→ 괄호 검사에 스택을 사용할 수 있음

```
int find_max(int A[], int n)
{
    int i, tmp=A[0];

    for ( i=1; i<n; i++) {
        if ( A[i] > tmp) {
            tmp = A[i];
        }
    }

    return tmp;
}
```

스택

괄호 검사 문제와 스택

1 괄호 검사의 개념

● 괄호 검사 조건

1 왼쪽 괄호의 개수와 오른쪽 괄호의 개수가 같아야 함

2 같은 타입의 괄호에서 왼쪽 괄호가 오른쪽 괄호보다 먼저 나와야 함

3 서로 다른 타입의 괄호 쌍이 서로를 교차해서는 안 됨

예 | 괄호 사용

{ A[(i+1)]=0; }	→ 오류 없음
if ((i==0) && (j==0)	→ 오류: 조건 1 위반 (')' 없음)
while (it < 10) {it--;}	→ 오류: 조건 2 위반 (')' 이 먼저 나옴)
A[(i+1)]=0;	→ 오류: 조건 3 위반 (')' 가 닫히기 전에 '] ' 를 닫음)

괄호 검사 문제와 스택



스택

괄호 검사 문제와 스택

1 괄호 검사의 개념

아이디어

가장 가까운 거리에 있는 괄호들끼리 서로 쌍을 이루어야 함

예 {{[(){}]}}}

- ▶ 열리는 괄호는 어딘가 저장하고, 닫히는 괄호는 최근에 열린 괄호와 비교하면 되지 않을까?
- ▶ **스택**을 사용
(열리는 괄호 저장, 스택에서 꺼내 닫히는 괄호와 비교)

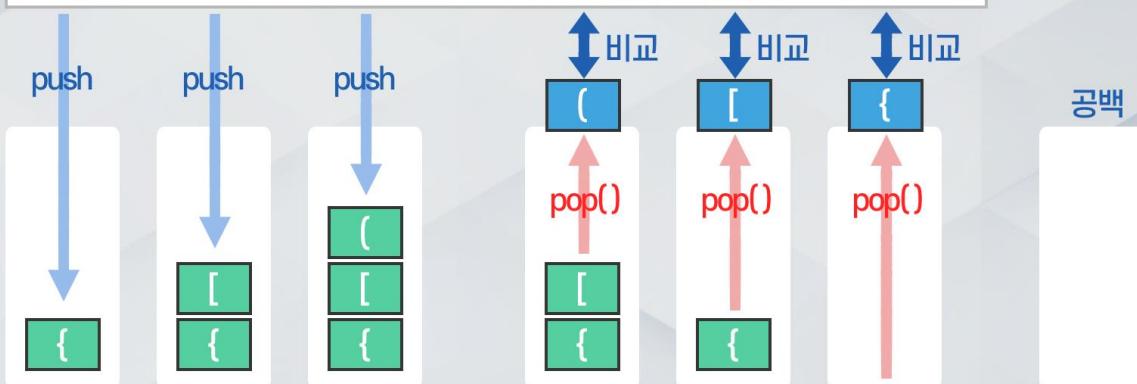
스택

괄호 검사 문제와 스택

2 스택을 이용한 괄호 검사 알고리즘

- 괄호 검사의 예 : 성공

```
// 열리는 괄호만 스택에 넣음
// 닫히는 괄호가 나오면 pop( ) 연산
{   A [     (   i+1   ) ] =0; }
```





괄호 검사 문제와 스택

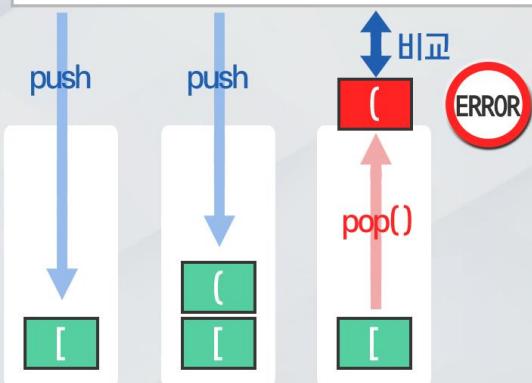
스택

괄호 검사 문제와 스택

2 스택을 이용한 괄호 검사 알고리즘

● 괄호 검사의 예 : 실패

```
// 닫히는 괄호 ']' 와 스택에서 pop()한 항목과 일치하지 않음
// 오류: 조건 3 위반
A[      ( i+1 ] ) =0;
```



스택

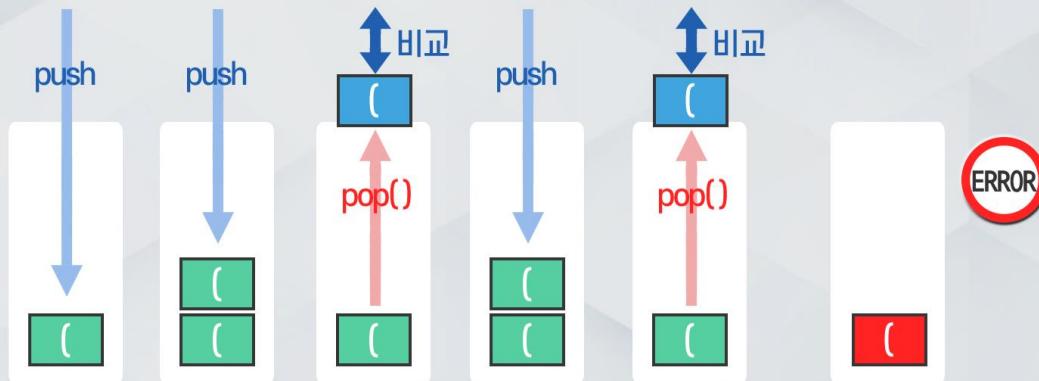
괄호 검사 문제와 스택

2 스택을 이용한 괄호 검사 알고리즘

● 괄호 검사의 예 : 실패

```
// 오류: 조건 1 위반
```

```
if(      ( i==0 ) && ( j==0 )
```



괄호 검사 문제와 스택



스택

괄호 검사 문제와 스택

2 스택을 이용한 괄호 검사 알고리즘

● 알고리즘 (자연어)

1 공백 상태의 스택을 준비

2 입력 문자열의 문자를 하나씩 읽어 왼쪽 괄호를 만나면 스택에 삽입

3 오른쪽 괄호를 만나면 가장 최근에 삽입된 괄호 꺼냄
이때 스택이 비었으면 조건 2에 위배

4 꺼낸 괄호가 오른쪽 괄호와 짝이 맞지 않으면 조건 3에 위배

5 끝까지 처리했는데 스택에 괄호가 남아있으면 조건 1에 위배

스택

괄호 검사 문제와 스택

2 스택을 이용한 괄호 검사 알고리즘

```
from stackClass import Stack # 이전에 구현한 Stack class 사용
```

```
def checkBrackets(statement):
    stack = Stack() # 공백 스택 생성
    for ch in statement:
        if ch in ('{', '[', '('): # 만약 ch가 열리는 괄호이면
            stack.push(ch) # 해당 ch를 stack에 삽입
        elif ch in ('}', ')', ']'): # 만약 ch가 닫히는 괄호이면
            if stack.isEmpty(): # 조건2 위배: 열리는 괄호 먼저
                return False
```



괄호 검사 문제와 스택

스택

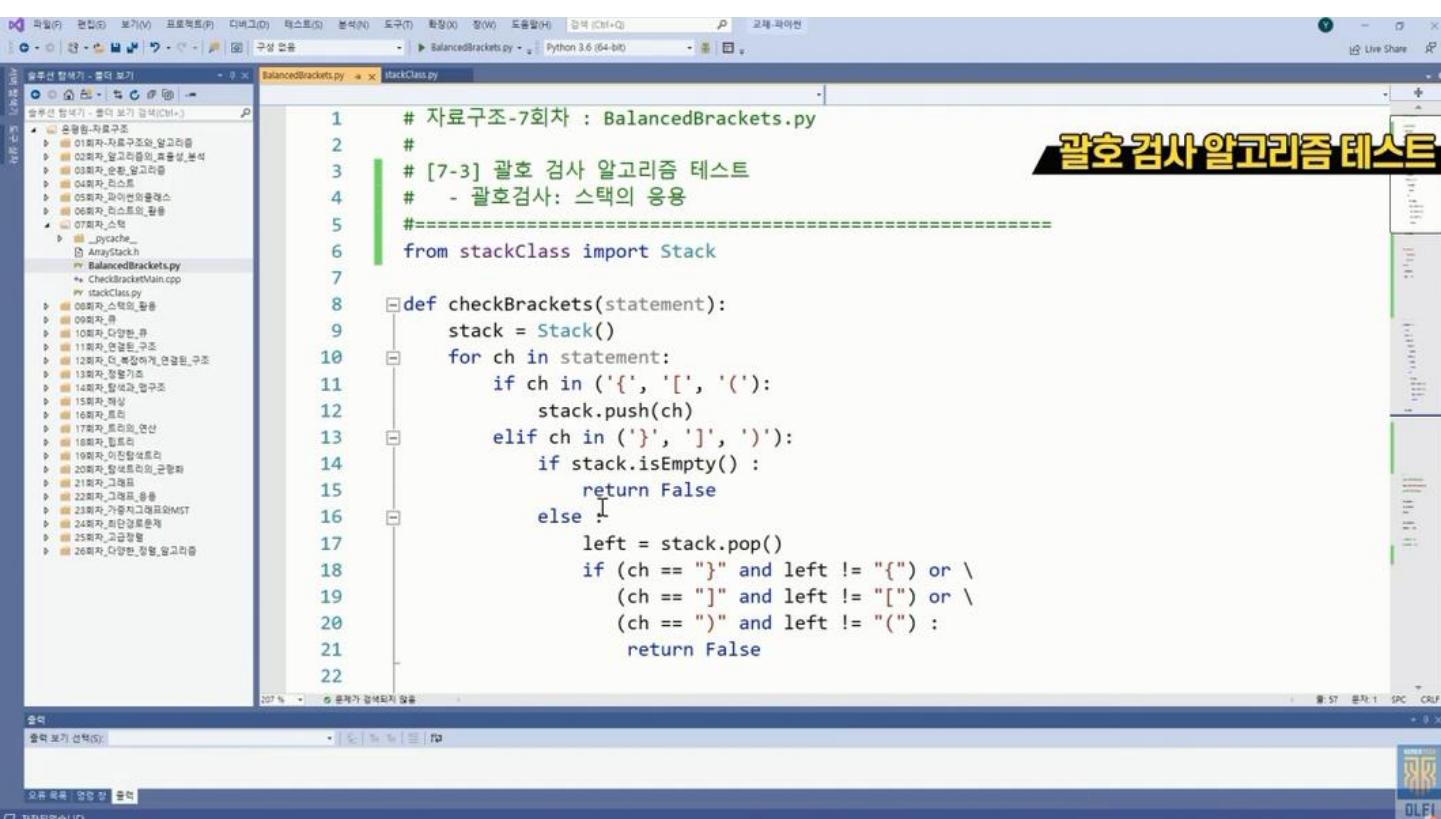
괄호 검사 문제와 스택

2 스택을 이용한 괄호 검사 알고리즘

```
else :  
    left = stack.pop()  
    if (ch == ")" and left != "(") or \  
        (ch == "]" and left != "[") or \  
        (ch == ")" and left != "(") :  
            return False      # 조건3 위배: 짹이 맞지 않음  
    return stack.isEmpty()      # 조건1 검사: 개수 동일  
                                # isEmpty() = False이면 return False
```

괄호 검사 문제와 스택

괄호 검사 알고리즘 테스트



```

1 # 자료구조-7회차 : BalancedBrackets.py
2 #
3 # [7-3] 괄호 검사 알고리즘 테스트
4 # - 괄호검사: 스택의 응용
5 =====
6 from stackClass import Stack
7
8 def checkBrackets(statement):
9     stack = Stack()
10    for ch in statement:
11        if ch in ('{', '[', '('):
12            stack.push(ch)
13        elif ch in ('}', ']', ')'):
14            if stack.isEmpty():
15                return False
16            else:
17                left = stack.pop()
18                if (ch == "}" and left != "{") or \
19                    (ch == "]" and left != "[") or \
20                    (ch == ")" and left != "("):
21                    return False
22

```

실습 단계

괄호 검사 알고리즘 테스트

checkBrackets: 괄호 검사 알고리즘 구현

테스트 코드

실행 결과 확인