스택 (Stack)

01 스택 이론

02 스택의 활용

신 제 용

01 스택 이론

제한된 기능을 제공하는 스택 추상자료형을 이해하고,
이를 구현하는 자료구조를 학습합니다.

학습 키워드 - 스택, 접근 제한, 자료구조

Chapter 01
스택 이론

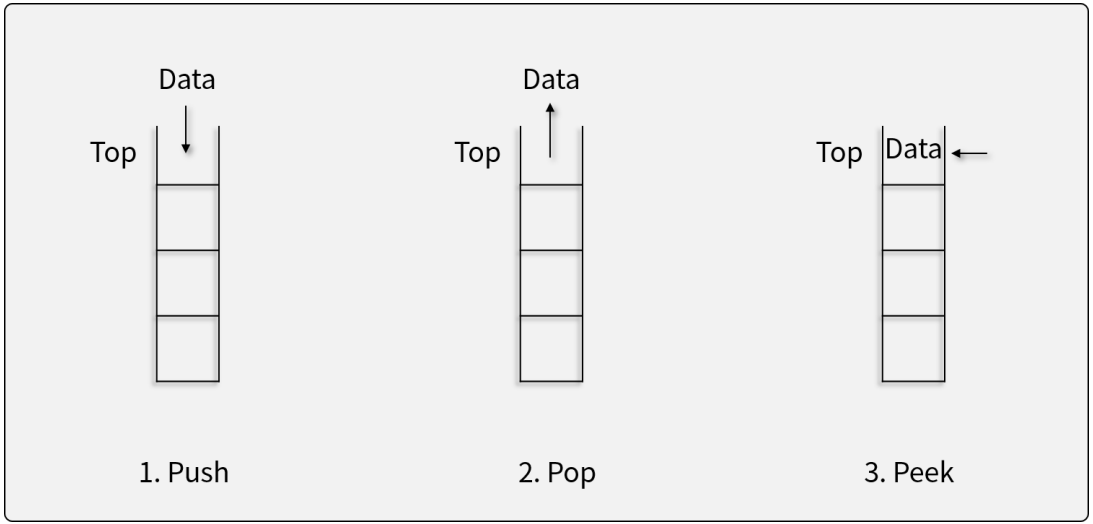
ZEROBASE SCHOOL

스택 (Stack)

- 리스트와 달리 **기능을 제한하는 추상자료형**
- 후입선출 (Last In First Out; LIFO)의 특성
- 자료가 입력된 역순으로 처리되어야 할 때 사용
  ex) 함수 콜 스택, 인터럽트 처리, 수식 계산 등

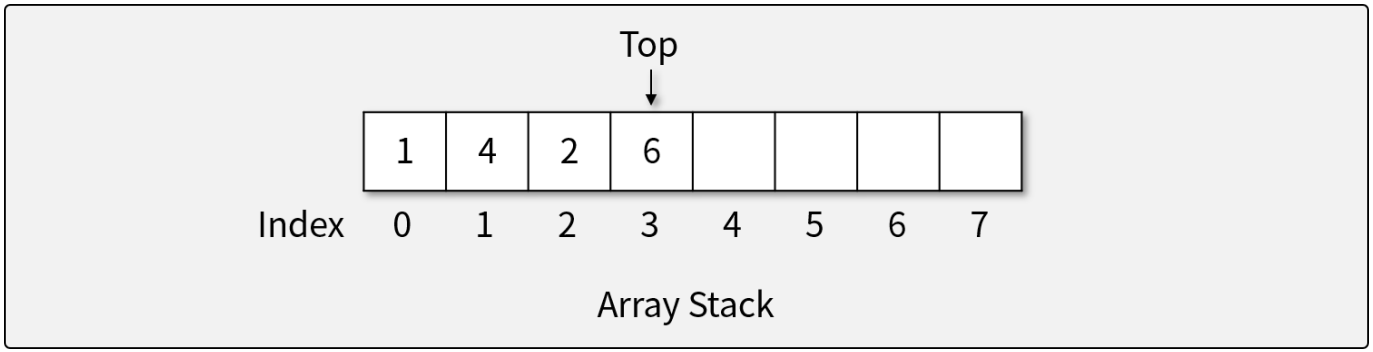
스택의 연산자

- 자료를 top 위에 삽입하는 연산자 (push())
- 자료를 top에서 꺼내는 연산자 (pop())
- top에 있는 자료를 반환하지만, 삭제하지는 않는 연산자 (peek())
- 스택이 비어있는지 확인하는 연산자 (is_empty())



스택의 구현

- 배열을 사용하므로, 크기가 정해져 있다.
- 메모리상 자료가 연속적이기 때문에 동작 속도가 빠르다.



오버플로우와 언더플로우

- 스택은 고속 동작을 위해 보통 배열로 구현

- 스택이 가득 차 있을 때 push()하면 **오버플로우** 발생
  → 오버플로우는 메모리 공간의 부족으로 발생

- 스택이 비어있을 때 pop()하면 **언더플로우** 발생
  → 언더플로우는 프로그램의 버그로 발생

02 스택의 활용

다음 챕터에서는 스택을 활용하는 예시를 확인하고 구현해봅니다.

Chapter 01
스택 이론

ZEROBASE SCHOOL

02 스택의 활용

스택을 활용하는 기초 유형 문제를 확인하고 직접 풀어봅시다.

학습 키워드 - 스택, 활용, 구현

Chapter 02
스택의 활용

ZEROBASE SCHOOL

```python
# Python에서 stack 사용하기 (리스트 응용)

stack = []

print(len(stack) == 0)  # is_empty()는 len(stack) == 0 으로 판단

for i in range(1, 11):
    stack.append(i)  # push()를 구한하고 있는 append

print(stack)
print(len(stack) == 0)  # is_empty()는 len(stack) == 0 으로 판단

print(stack[-1]) # peek()는 [-1]로 구현

for i in range(1, 11):
    val = stack.pop()  # pop()
    print(val, end=' ')
print()

print(len(stack) == 0)  # is_empty()는 len(stack) == 0 으로 판단
```

```python
# Prob1.
# 스택을 이용하여 문자열 s를 뒤집기
#
#
# 입출력 예시)
# s = "zerobase"
# 결과: "esaborez"


def solution(s):
    pass


if __name__ == '__main__':
    s = "zerobase"
    sol = solution(s)
    print(sol)
```

Chapter 02
스택의 활용

ZEROBASE SCHOOL

```python
# Prob2.
# 괄호 짝이 잘 맞는지 검사하기
# 입력받은 문자열 s의 괄호가
# 올바른 짝이면 True, 아니면 False
#
# 입출력 예시)
# s = "(())"
# 결과: True
# s = "(()"
# 결과: False
# s = "()()()"
#
# 결과: True

def solution(s):
    pass


if __name__ == '__main__':
    s = "(())"
    sol = solution(s)
    print(sol)
```

Chapter 02
스택의 활용

```python
# Prob3.
# 후위표기법으로 표기된 계산식 s 연산하기
# 후위표기법: 피연산자가 2개 나온 후에 연산자가 나오는 표기법
#             ex) "4 2 +" -> 4 + 2 = 6
#
# 입출력 예시)
# s = "2 4 +"
# 결과: 6
#
# s = "2 2 -"
# 결과: 0
#
# s = "1 3 + 5 * 2 * 5 +"
# 결과: 45
#

def solution(s):
    pass

if __name__ == '__main__':
    s = "2 4 +"
    sol = solution(s)
    print(sol)
```

Chapter 02
스택의 활용