대구대학교 DAEGU UNIVERSITY

일상에서 찾아보는

자료구조의 원리

03주차 쌓아 올리기 - 스택

2차시

스택의 LIFO(Last-In First-Out) 동작 만들기

학습목표

- 스택의 동작을 만들어내는 추상자료형에 대해 설명할 수 있다.
- 십진수의 회문 검사, 이진수 변환하기 문제를 스택으로 해결할 수 있다.

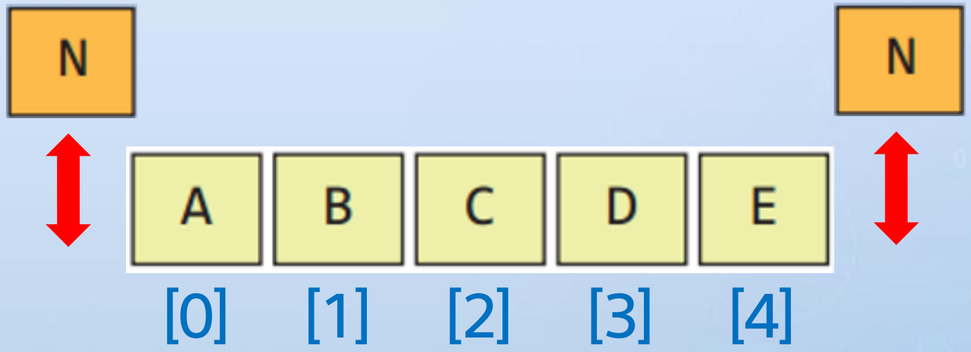
학습내용

- 파이썬 리스트 고려 사항
- 스택 클래스
- 스택 클래스 활용
- 스택 활용 실습

파이썬 리스트 고려 사항

» 스택의 저장 공간: 리스트

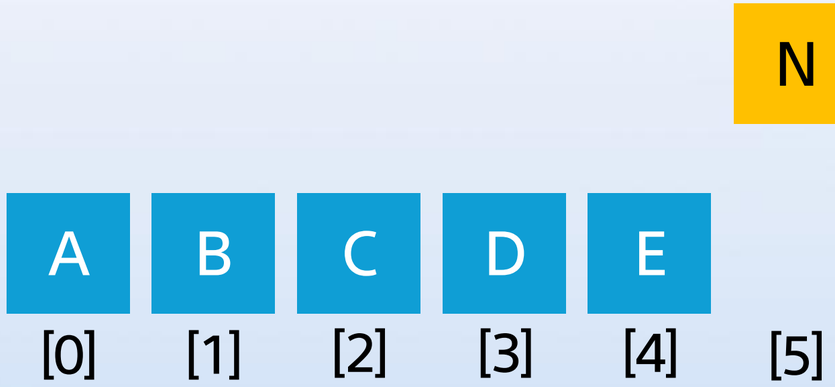
- 자료가 들어가고 나오는 입구를 정함
  - 리스트의 마지막에 자료를 넣거나 빼냄
  - 리스트의 처음에서 자료를 넣고 빼도 동작함



출처 교수자 제공

파이썬 리스트 고려 사항

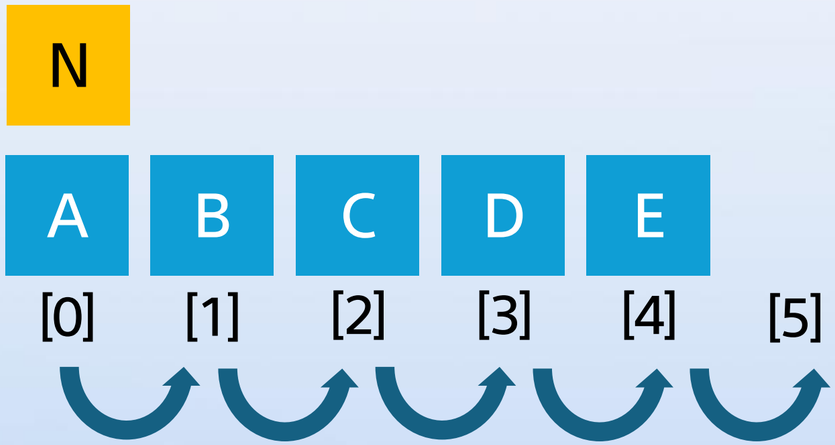
» append(N): $O(1)$



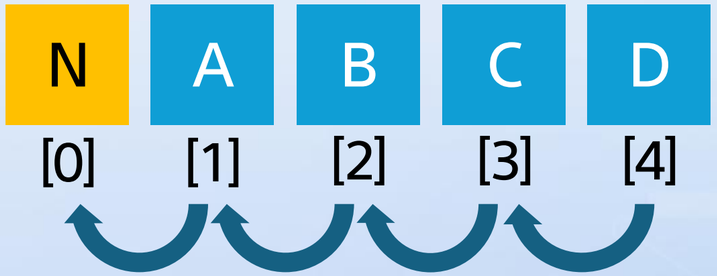
» pop(): $O(1)$

파이썬 리스트 고려 사항

» insert(0,N) : $O(n)$



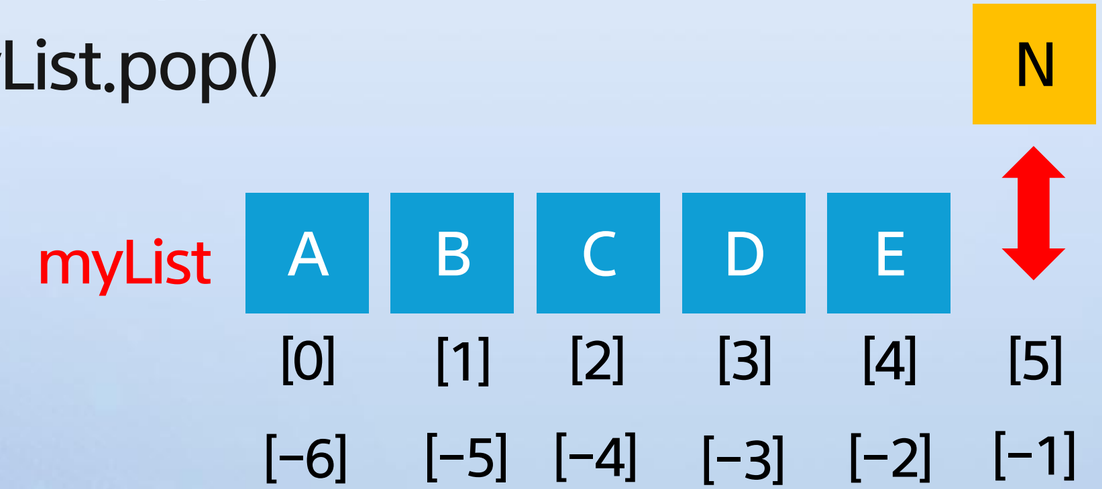
» pop(0) : $O(n)$



파이썬 리스트 고려 사항

» 파이썬 리스트의 효율적 사용

- 후단(마지막 위치) 삽입이나 삭제
- 중간이나 전단(처음 위치) 삽입이나 삭제는 비효율적임
  - myList.append(N)
  - myList.pop()



2 스택의 LIFO(Last-In First-Out) 동작 만들기

스택 클래스


```
1  class Stack :
2      def __init__( self ):
3          self.s_list = [] #스택 저장 공간을 빈
4
5      def push(self, data) :  #push는 삽입할 자료
6          self.s_list.append(data)
7      def pop(self) :
8            if not self.isEmpty() : #빈 상태가 C
9                  return self.s_list.pop(-1) #가
10     def isEmpty(self) :
11           return len(self.s_list) == 0  #s.lis
12     def size(self) :
13           return len(self.s_list)
14     def clear(self) :
15           self.s_list = []
```

스택 클래스 정의

```
1  #스택 클래스 객체 생성
2  stack=Stack()
3
4  #push 실행
5  stack.push(9)
6
7  #pop 실행
8  stack.pop()
```

- 스택 객체 생성
- push 실행
- pop 실행
- isEmpty 실행

스택 클래스

```
class Stack :
  def __init__( self ):
    self.s_list = [ ]
  def push(self, data) :
    self.s_list.append(data)
  def pop(self) :
    if not self.isEmpty() :
      return self.s_list.pop(-1)
  def isEmpty(self) :
    return len(self.s_list) == 0
```

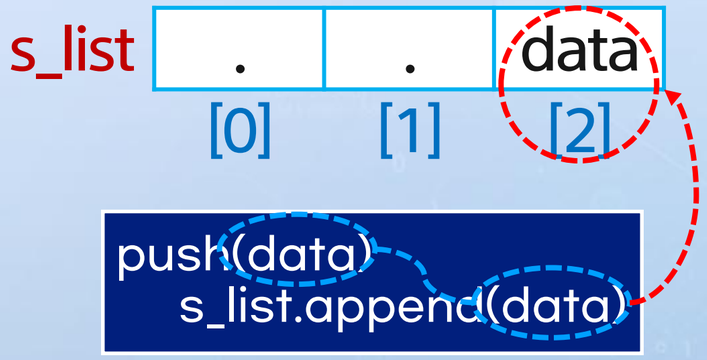
1. 생성자: 자료 저장 공간 생성
   - 파이썬 리스트를 사용하여 배열 구조로 구현
     - ➡ 빈 리스트 생성
   - 논리적 입구(스택 상단): 리스트의 맨 끝

스택 클래스

```
class Stack :
  def __init__( self ):
    self.s_list = [ ]
  def push(self, data) :
    self.s_list.append(data)
  def pop(self) :
    if not self.isEmpty() :
      return self.s_list.pop(-1)
  def isEmpty(self) :
    return len(self.s_list) == 0
```

2 push(data): #push는 삽입할 자료(값)를 필요로 한다.

- 리스트의 마지막에 data를 삽입함
  - ➡ append() 이용
- 리스트 인덱스가 1 증가

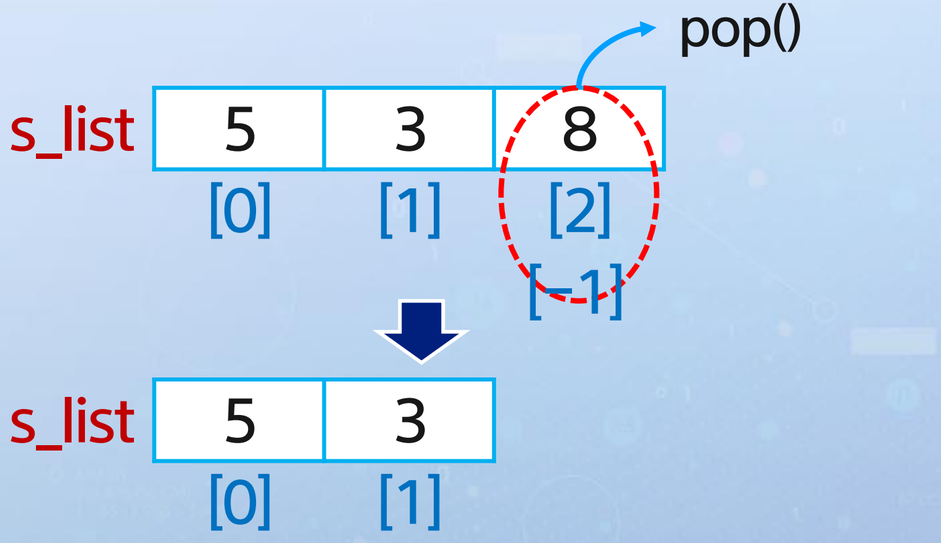


스택 클래스

```
class Stack :
  def __init__( self ):
    self.s_list = [ ]
  def push(self, data) :
    self.s_list.append(data)
  def pop(self) :
    if not self.isEmpty() :
      return self.s_list.pop(-1)
  def isEmpty(self) :
    return len(self.s_list) == 0
```

3 pop()

- 파이썬 리스트의 pop()을 이용 (마지막 항목 삭제)
- 스택이 빈 상태가 아닌 경우에 가능하므로 빈 상태인지 먼저 검사함

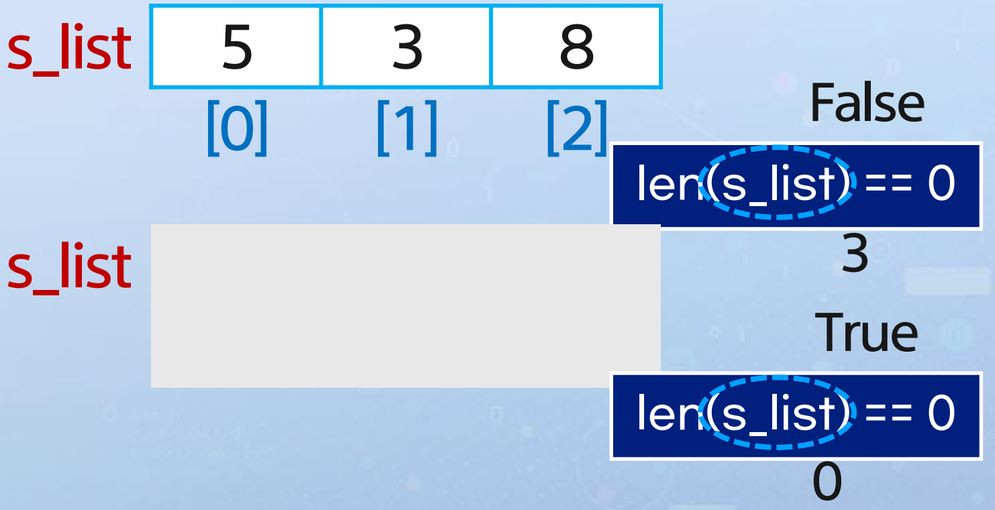


스택 클래스

```
class Stack :
  def __init__( self ):
    self.s_list = [ ]
  def push(self, data) :
    self.s_list.append(data)
  def pop(self) :
    if not self.isEmpty() :
      return self.s_list.pop(-1)
  def isEmpty(self) :
    return len(self.s_list) == 0
```

4 isEmpty()

- 스택이 비었는지 검사
  - ➡ 비었다 True, 그렇지 않다 False
- 스택(리스트)의 크기(길이)가 0인가?
  - ➡ len()을 사용



스택 클래스

```
class Stack :
  def __init__( self ):
    self.s_list = [ ]
  def push(self, data) :
    self.s_list.append(data)
  def pop(self) :
    if not self.isEmpty() :
      return self.s_list.pop(-1)
  def isEmpty(self) :
    return len(self.s_list) == 0
  def size(self) :
    return len(self.s_list)
  def clear(self) :
    self.s_list = []
```

- size()
  - 스택의 크기를 알아봄
    - ➡ 리스트의 len()을 이용
- clear()
  - 스택을 비움
    - ➡ 빈 리스트를 할당함

스택 클래스 활용

```
#스택 클래스 객체 생성
stack=Stack()

#push 실행
stack.push(9)

#pop 실행
stack.pop()
```

```
9
```

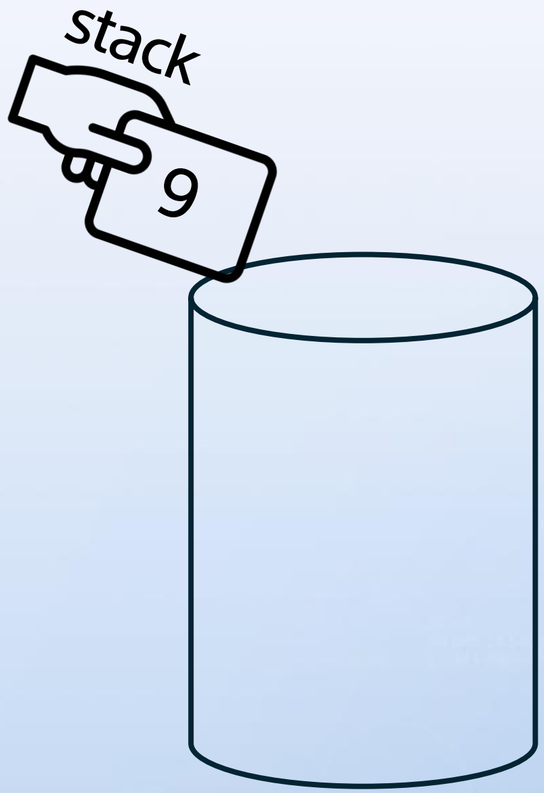
```
stack.isEmpty()
```

```
True
```

테스트

출력 결과
〉〉 9

출력 결과
〉〉 True

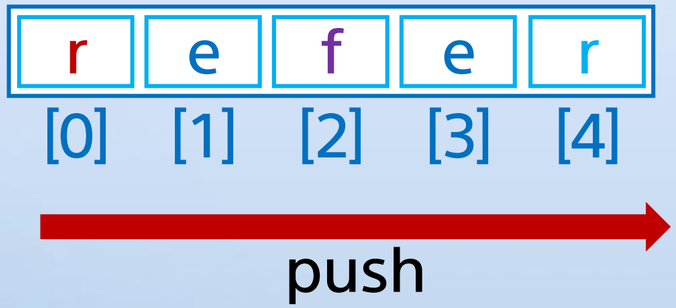


스택 활용 실습

» 회문(palindrome) 검사

- 한 문자 씩 스택에 push
- pop: 원래 문자열과 비교

“ r e f e r ”



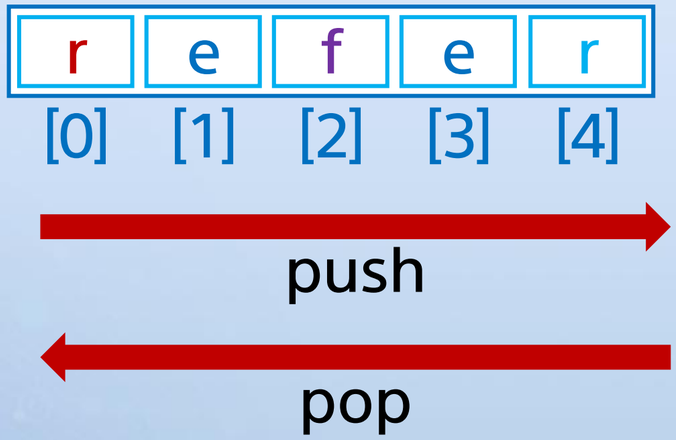

```
stack = Stack()
instr = input("문자열 입력:")
for ch in instr:
  stack.push(ch)
for ch in instr:
  if ch != stack.pop() :
    print("회문이 아님")
    break
if stack.isEmpty():
  print("회문이 맞음")
```

스택 활용 실습

» 회문(palindrome) 검사

- 한 문자 씩 스택에 push
- pop: 원래 문자열과 비교

“ r e f e r ”




```
stack = Stack()
instr = input("문자열 입력:")
for ch in instr:
  stack.push(ch)
for ch in instr:
  if ch != stack.pop() :
    print("회문이 아님")
    break
if stack.isEmpty():
  print("회문이 맞음")
```

스택 활용 실습

» 회문(palindrome) 검사

```
1  stack = Stack()
2  instr = input("문자열 입력:")
3  for ch in instr:
4      stack.push(ch)
5  for ch in instr:
6      if ch != stack.pop():
7          print("회문이 아님")
8          break
9  if stack.isEmpty():
10     print("회문이 맞음")
```

키보드에서 문자열 입력

push 실행

pop 실행

문자열 입력: rafer

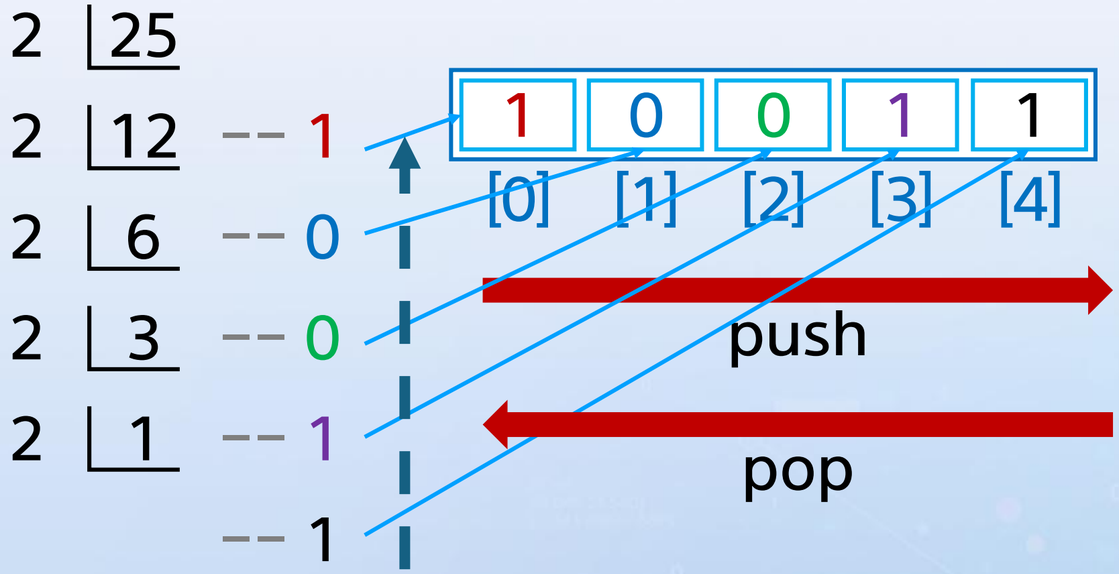
키보드 입력

“ r a f e r ”

스택 활용 실습

» 이진수 변환

```
def decimaltobinary(num):
    s = Stack()
    while num > 0:
        s.push(num%2)
        num = num//2
    while not s.isEmpty():
        print(s.pop(),end='')
```



스택 활용 실습

» 이진수 변환

Run Code

십진수를 이진수로 변환하기

```
1  def decimaltobinary(num):
2      s = Stack()
3      while num > 0:
4          s.push(num % 2)
5          num = num // 2
6      while not s.isEmpty():
7          print(s.pop(),end='')
```

함수 정의

```
1  decimaltobinary(25)
```

호출하여 테스트

```
11001
```

학습정리

Stack

- 스택은 LIFO 방식에 따라 가장 최근에 추가된 데이터가 가장 먼저 삭제되는 구조임
- 스택은 push(추가)와 pop(삭제) 연산을 통해 동작함