리스트

프로그래밍2 3주차

데이터사이언스학과

곽찬희

학습목표

- 리스트의 개념과 추상 자료형을 이해한다.
- 배열 구조와 연결된 구조의 차이를 정확히 이해한다.
- 파이썬 리스트의 내부 동작 원리를 이해한다.
- 자료구조 리스트를 파이썬 리스트로 구현하는 방법을 이해한다.
- 자료구조를 함수와 클래스로 구현하는 방법의 차이를 이해한다.
- 집합의 개념과 구현 방법을 이해한다.
- 자료구조의 각 연산들에 대한 시간 복잡도 분석 능력을 기른다.

3.1 리스트란?

- 리스트는 가장 자유로운 선형 자료구조이다.
  - ✓ 리스트의 구조
  - ✓ 리스트의 추상 자료형

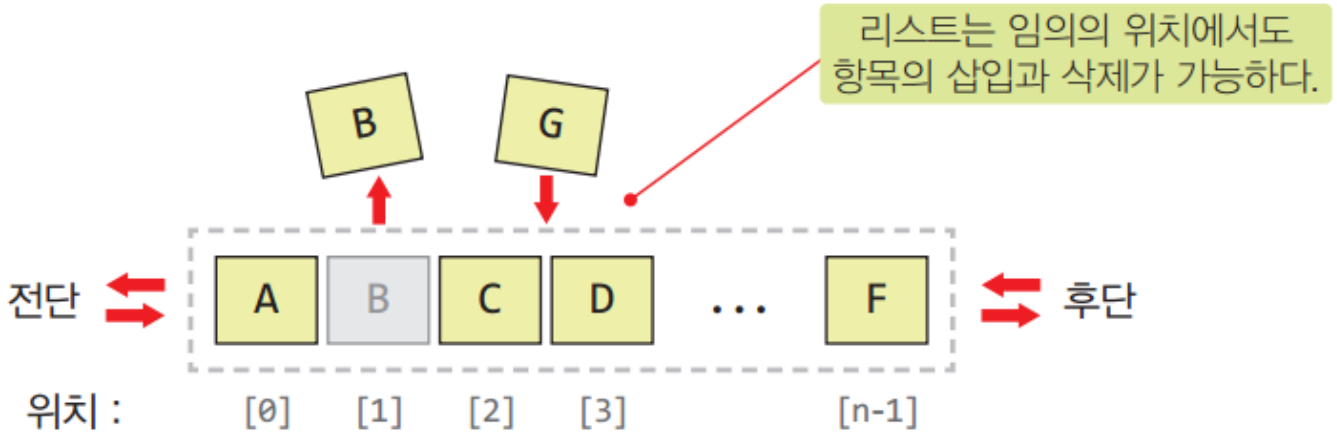
- 리스트의 구현 방법
  - ✓ 배열 구조와 연결된 구조로 구현할 수 있다.
  - ✓ 리스트와 관련된 용어의 정리

리스트란?

- 리스트(list), 선형리스트(linear list)
  - ✓ 순서를 가진 항목들의 모임
  - ✓ $L = [item_0, item_1, item_2, \cdots, item_{n-1}]$
  - ✓ 집합: 항목간의 순서의 개념이 없음

리스트의 구조

- 리스트
  - ✓ 항목들이 순서대로 나열되어 있고, 각 항목들 은 위치를 갖는다.



- Stack, Queue, Deque과의 차이점
  - ✓ 자료의 접근 위치

리스트 ADT

정의 3.1 List ADT

데이터: 같은 유형의 요소들의 순서 있는 모임

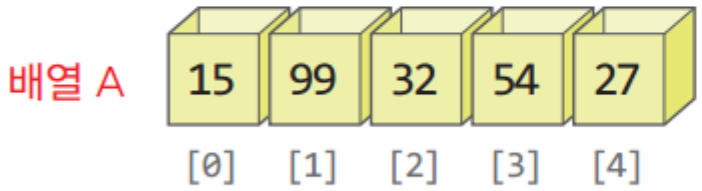
연산

- List(): 비어 있는 새로운 리스트를 만든다.
- insert(pos, e): pos 위치에 새로운 요소 e를 삽입한다.
- delete(pos): pos 위치에 있는 요소를 꺼내고(삭제) 반환한다.
- isEmpty(): 리스트가 비어있는지를 검사한다.
- getEntry(pos): pos 위치에 있는 요소를 반환한다.
- size(): 리스트안의 요소의 개수를 반환한다.
- clear(): 리스트를 초기화한다.
- find(item): 리스트에서 item이 있는지 찾아 인덱스를 반환한다.
- replace(pos, item): pos에 있는 항목을 item으로 바꾼다.
- sort(): 리스트의 항목들을 어떤 기준으로 정렬한다.
- merge(lst): 다른 리스트 lst를 리스트에 추가한다.
- display(): 리스트를 화면에 출력한다.
- append(e): 리스트의 맨 뒤에 새로운 항목을 추가한다.

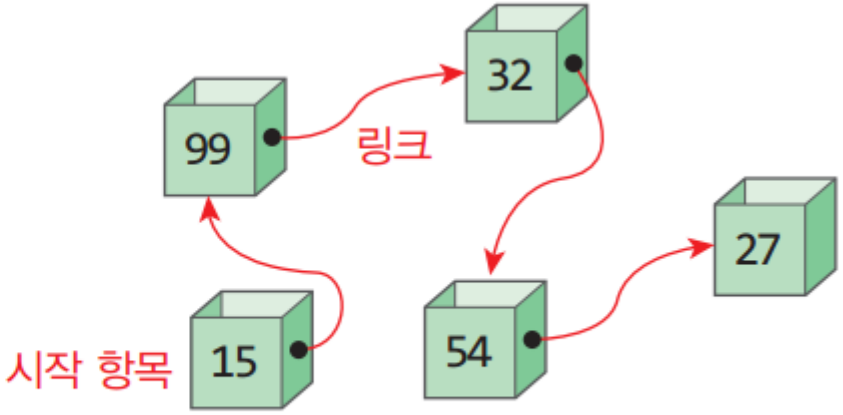
리스트 구현 방법

- 배열 구조
  - ✓ 구현이 간단
  - ✓ 항목 접근이 $O(1)$
  - ✓ 삽입, 삭제시 오버헤드
  - ✓ 항목의 개수 제한

- 연결된 구조
  - 구현이 복잡
  - 항목 접근이 $O(n)$
  - 삽입, 삭제가 효율적
  - 크기가 제한되지 않음



배열 구조의 리스트



연결된 구조의 리스트

리스트 용어 정리

파이썬 리스트	C언어에서의 배열이 진화된 형태의 스마트한 배열이다. 이 책에서는 배열 또는 배열 구조의 의미로 사용한다. 어떤 자료구조를 구현하기 위한 하나의 방법으로 사용한다.
연결 리스트	자료들이 일렬로 나열할 수 있는 연결된 구조를 말한다. 배열 구조(파이썬의 리스트)에 대응되는 의미로 사용한다.
자료구조 리스트	추상적인 의미의 자료구조 리스트를 의미한다. 앞에서 우리는 이 리스트의 ADT를 정의하였다. 이를 구현하기 위해 배열 구조(파이썬의 리스트)나 연결된 구조(연결 리스트)를 사용할 것이다.

3.2 파이썬 리스트

- 파이썬의 리스트는 스마트한 배열이다.
- 파이썬의 리스트는 동적 배열로 구현되었다.
- 파이썬 리스트의 시간 복잡도

파이썬 리스트

- 파이썬 리스트는 스마트한 배열이다
- 리스트 선언

```
A = [ 1, 2, 3, 4, 5 ]                # 파이썬 리스트 A
```

✓ C 언어의 배열 선언

```
int A[5] = { 1, 2, 3, 4, 5 };        // 정수 배열 A선언 및 초기화
```

- 항목의 수

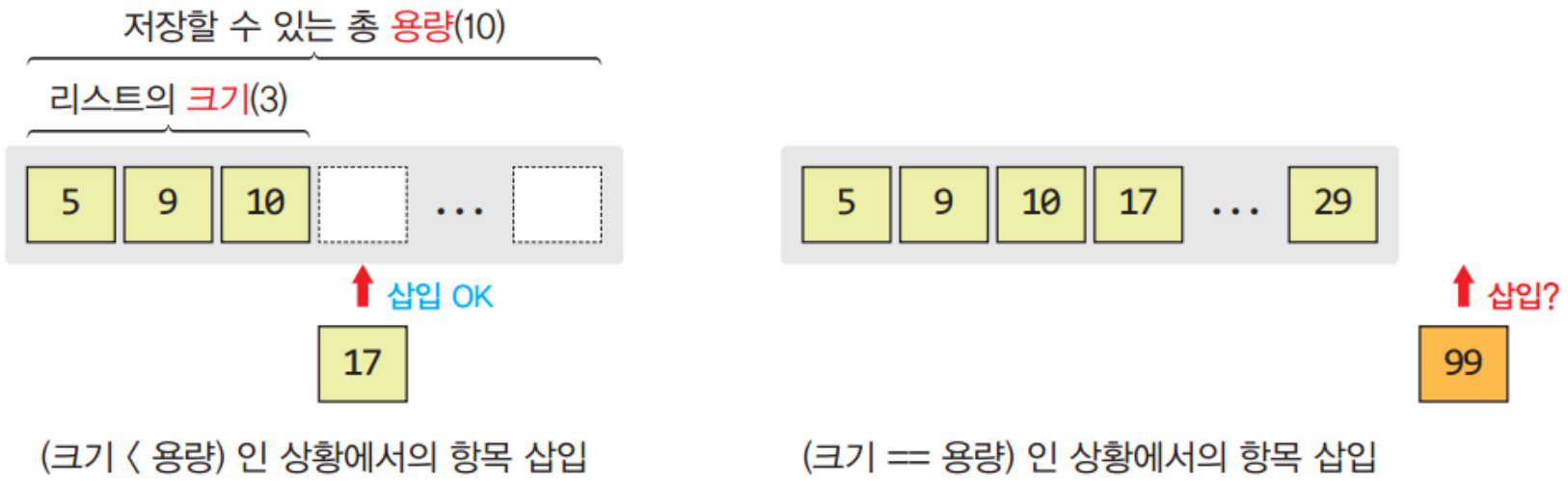
```
print('파이썬 리스트 A의 크기는 ', len(A))        # A의 크기(항목 수) 출력
```

- 항목 추가: 용량을 늘릴 수 있다.

```
A.append(6)          # A = [1, 2, 3, 4, 5, 6]
A.append(7)          # A = [1, 2, 3, 4, 5, 6, 7]
A.insert(0, 0)       # A = [0, 1, 2, 3, 4, 5, 6, 7]
```

파이썬 리스트는 동적 배열로 구현되었다

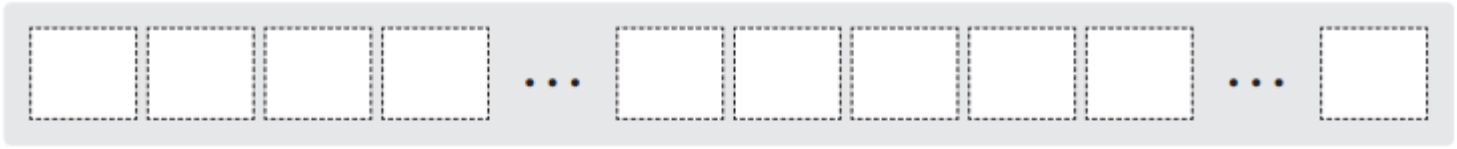
- 필요한 양보다 넉넉한 크기의 메모리를 사용!



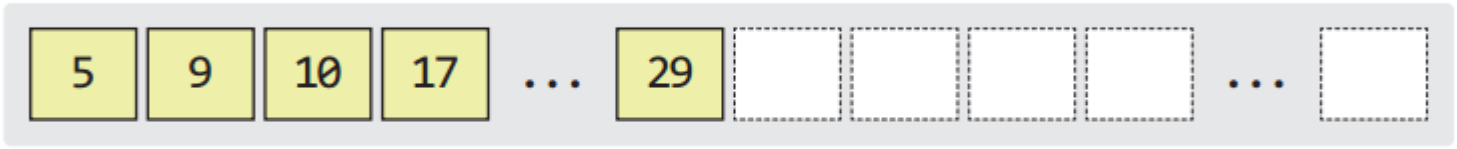
- 남은 공간이 없으면 어떻게 삽입할까?

동적 배열 구조에서의 용량 증가 과정

Step1: 용량을 확장한 새로운 배열 할당. (예: 기존 배열 용량의 2배)



Step2: 기존의 배열을 새로운 배열에 복사

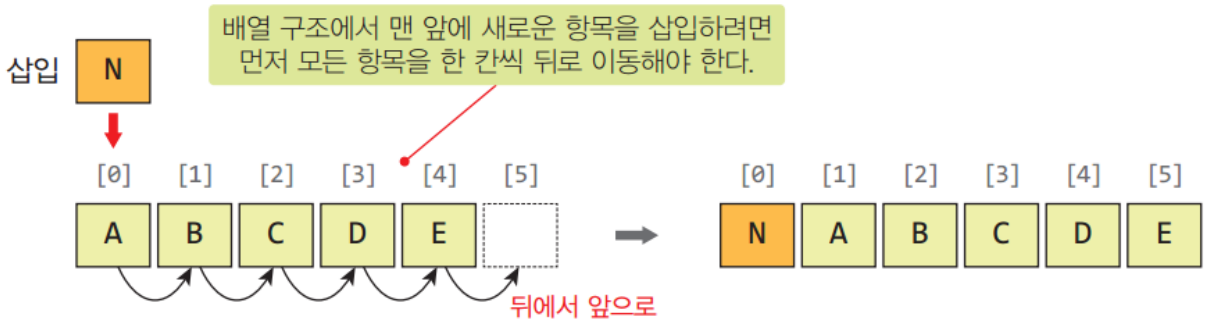


Step3: 항목을 삽입

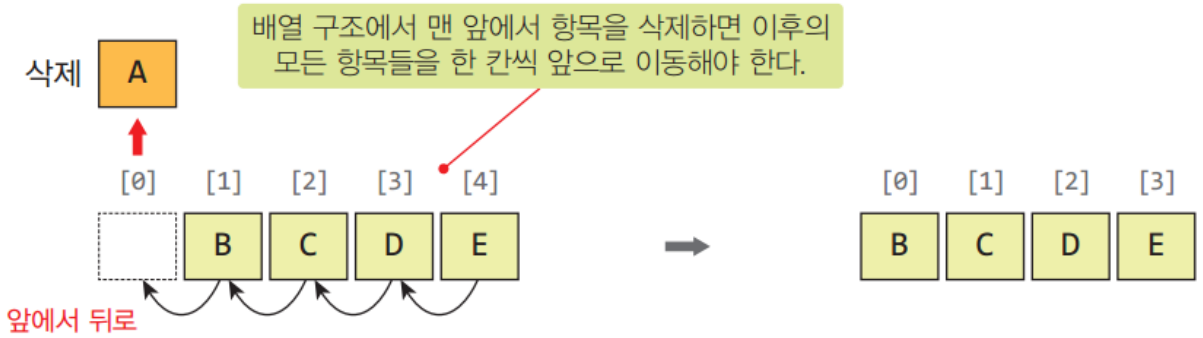
Step4: 기존 배열 해제, 리스트로 새 배열 사용

파이썬 리스트의 시간 복잡도

- **append(e)연산: 대부분의 경우** $O(1)$
- **insert(pos, e)연산:** $O(n)$



- **pop(pos)연산:** $O(n)$



3.3 배열로 구현한 리스트

- 배열 구조로 자료구조 리스트를 구현하자.
  - ✓ 전역변수와 함수로 구현한 리스트
  - ✓ 테스트 프로그램
- 클래스로 구현하는 것이 더 좋은 방법이다.
  - ✓ 클래스로 구현한 리스트
  - ✓ 테스트 프로그램
- 연산들의 시간 복잡도?

배열로 구현한 리스트(함수 버전)

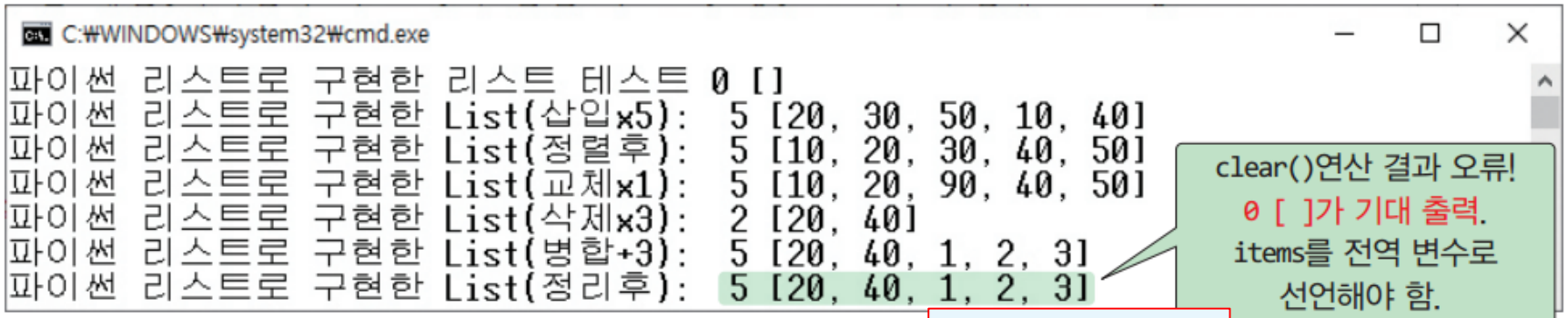
- 자료구조 리스트의 ADT 구현에 파이썬 리스트 이용
- 함수 버전: 전역변수와 함수로 구현

```
items = []                  # 리스트 항목 저장을 위한 파이썬 리스트
def insert(pos, elem) :     # pos 위치에 새로운 요소 item을 삽입한다.
  items.insert(pos, elem)   # 파이썬 리스트  클래스의 insert 연산
def delete(pos) :           # pos 위치에 있는 요소를 꺼내고 반환한다.
   return items.pop(pos)    # 파이썬 리스트 클래스의 pop 연산
def getEntry(pos): return items[pos]      # pos번째 항목 반환
def isEmpty( ): return len(items) == 0    # 크기가 0이면 True 아니면 False
def size( ):       return len(items)      # 리스트의 크기 반환. len()함수 이용
def clear( ):      items = []             # items를 초기화 --> 오류
def find(item) : return items.index(item)      # 탐색 후 인덱스 반환
def replace(pos, elem): items[pos] = elem      # pos번째 항목 변경
def sort() : items.sort()                      # 정렬(sort메소드 이용)
def merge(lst) : items.extend(lst)             # 병합(확장)
def display(msg='ArrayList:' ):                # 출력: 디폴트 인수 사용
    print(msg, size(), items)                  # 메시지+크기+배열내용 출력
```

테스트 프로그램(함수 버전)

```
display('파이썬 리스트로 구현한 리스트 테스트')
insert(0, 10);          insert(0, 20);    insert(1, 30)
insert(size(), 40);     insert(2, 50)     # 한 줄에 여러 문장
display("파이썬 리스트로 구현한 List(삽입x5): ")
sort()
display("파이썬 리스트로 구현한 List(정렬후): ")
replace(2, 90)
display("파이썬 리스트로 구현한 List(교체x1): ")
delete(2); delete(size() - 1);    delete(0)
```

```
display("파이썬 리스트로 구현한 List(삭제x3): ")
lst = [ 1, 2, 3 ]
merge(lst)
display("파이썬 리스트로 구현한 List(병합): ")
clear()
display("파이썬 리스트로 구현한 List(정리후): ")
```



clear()연산 결과 오류!
0 [ ]가 기대 출력.
items를 전역 변수로 선언해야 함.
global items 추가

```
def clear( ) :
    global items
    items = []
```

배열로 구현한 리스트(클래스 버전)

```
class ArrayList:                    # 이 책에서 클래스 코드는 살구색 바탕 이용
    def __init__( self ):           # 생성자 (2.10절 참조)
        self.items = []             # 클래스 변수 선언 및 초기화

    def insert(self, pos, elem) :
        self.items.insert(pos, elem)
    def delete(self, pos) :
        return self.items.pop(pos)
    def isEmpty( self ):
        return self.size() == 0
    def getEntry(self, pos) :
        return self.items[pos]
    def size( self ):
        return len(self.items)
    def clear( self ) :
        self.items = []             # items는
```

```
    def find(self, item) :
        return self.items.index(item)
    def replace(self, pos, elem) :
        self.items[pos] = elem
    def sort(self) :
        self.items.sort()
    def merge(self, lst) :
        self.items.extend(lst)
    def display(self, msg='ArrayList:' ):
        print(msg, '항목수=', self.size(), self.items)
```

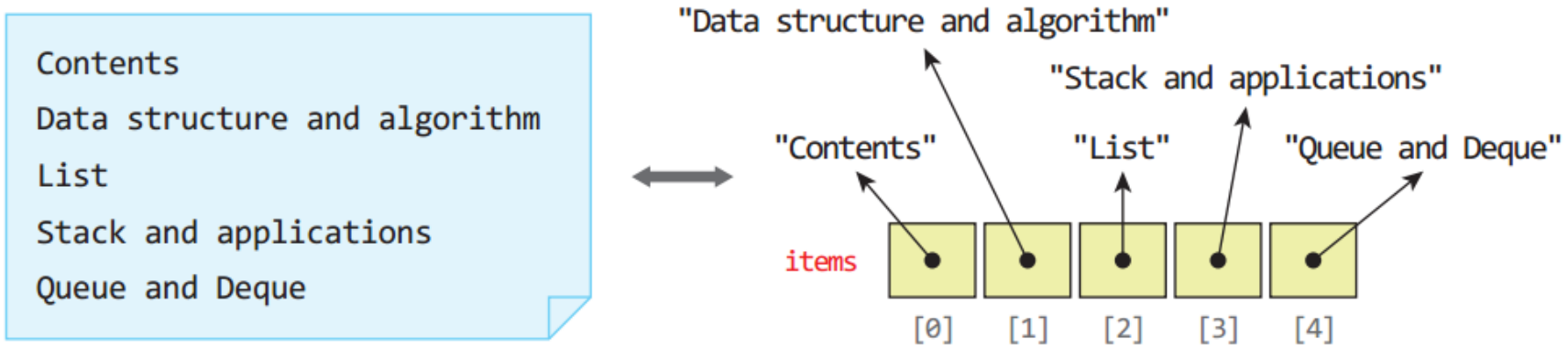
테스트 프로그램(클래스 버전)

```
s = ArrayList()
s.display('파이썬 리스트로 구현한 리스트 테스트')
s.insert(0, 10);          s.insert(0, 20);      s.insert(1, 30)
s.insert(s.size(), 40);   s.insert(2, 50)
s.display("파이썬 리스트로 구현한 List(삽입x5): ")
s.sort()
s.display("파이썬 리스트로 구현한 List(정렬후): ")
s.replace(2, 90)
s.display("파이썬 리스트로 구현한 List(교체x1): ")
s.delete(2);    s.delete(s.size() - 1);    s.delete(0)
s.display("파이썬 리스트로 구현한 List(삭제x3): ")
lst = [ 1, 2, 3 ]
s.merge(lst)
s.display("파이썬 리스트로 구현한 List(병합+3): ")
s.clear()
s.display("파이썬 리스트로 구현한 List(정리후): ")
```

3.4 리스트의 응용 : 라인 편집기

- 라인 편집기 기능
- 실행 예
- 리스트를 이용한 구현 코드

라인 편집기 기능

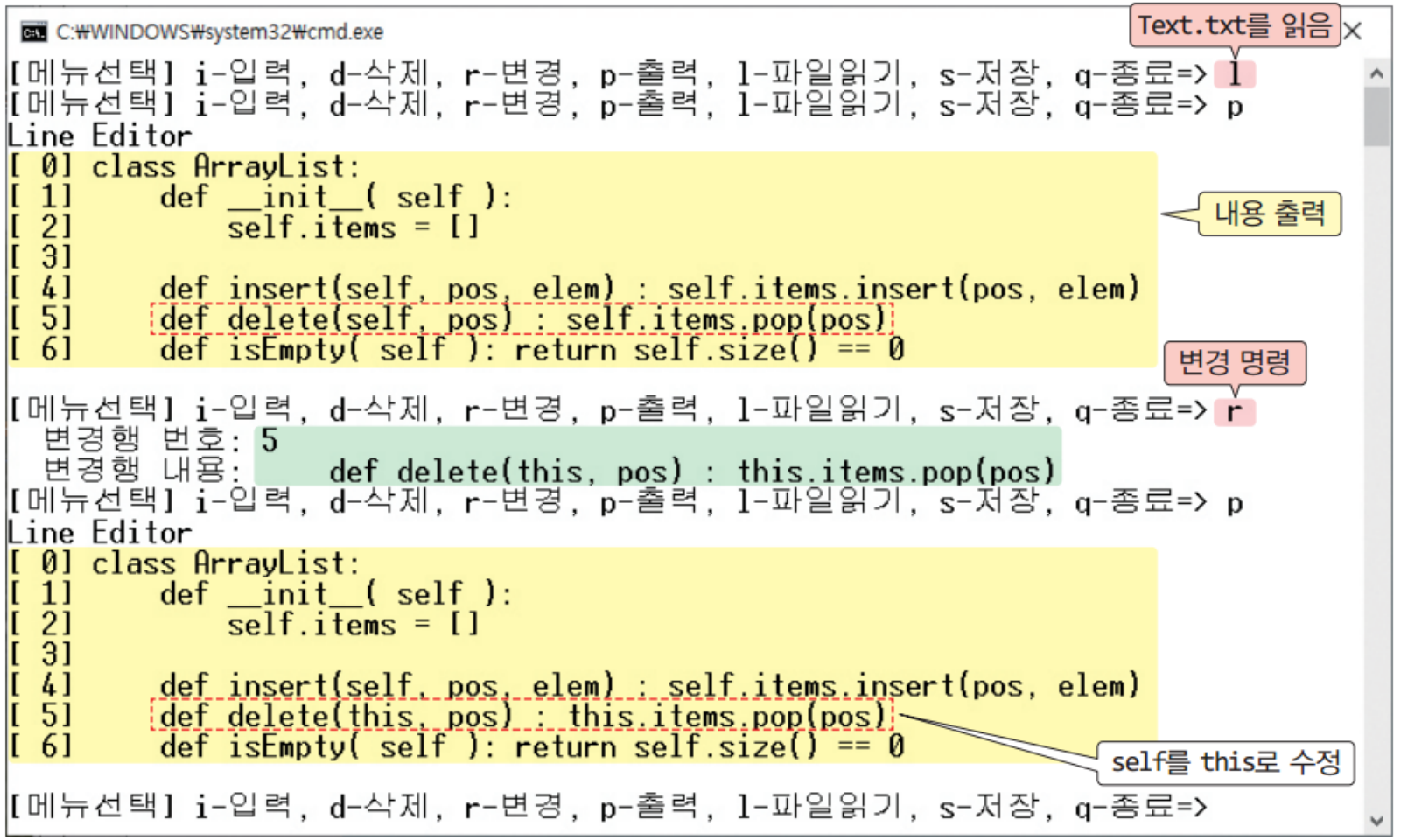


- 라인 편집기 기능
  - ✓ i: 라인 삽입. 행 번호와 문자열을 입력하면 그 행에 한 라인 추가
  - ✓ d: 한 라인 삭제. 행 번호를 입력하면 그 행을 삭제
  - ✓ r: 한 라인 변경. 행 번호와 문자열을 입력하면 그 행의 내용을 변경
  - ✓ p: 현재 내용 출력. 현재 문서의 모든 내용을 라인 번호와 함께 출력
  - ✓ l: 파일 입력. 지정된 (test.txt) 파일로부터 라인을 읽어 들임
  - ✓ s: 파일 출력. 지정된 (test.txt) 파일로 편집 내용을 저장

실행결과1


C:₩WINDOWS₩system32₩cmd.exe
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> i
입력행 번호: 0
입력행 내용: Contents
0번 줄에 삽입
라인 추가 명령
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> i
입력행 번호: 1
입력행 내용: 자료구조와 알고리즘
1번 줄에 삽입
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> i
입력행 번호: 2
입력행 내용: 리스트
2번 줄에 삽입
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> i
입력행 번호: 3
입력행 내용: 스택, 큐, 덱
3번 줄에 삽입
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> p
화면 출력 명령
Line Editor
[0] Contents
[1] 자료구조와 알고리즘
[2] 리스트
[3] 스택, 큐, 덱
내용 출력
삭제 명령
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> d
삭제행 번호: 1
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> p
Line Editor
[0] Contents
[1] 리스트
[2] 스택, 큐, 덱
1번 줄 삭제
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=>

실행결과2


C:₩WINDOWS₩system32₩cmd.exe
Text.txt를 읽음
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> l
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> p
Line Editor
[0] class ArrayList:
[1]   def __init__(self):
[2]   self.items = []
[3]
[4]   def insert(self, pos, elem) : self.items.insert(pos, elem)
[5]   def delete(self, pos) : self.items.pop(pos)
[6]   def isEmpty(self): return self.size() == 0
내용 출력
변경 명령
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> r
변경행 번호: 5
변경행 내용: def delete(this, pos) : this.items.pop(pos)
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> p
Line Editor
[0] class ArrayList:
[1]   def __init__(self):
[2]   self.items = []
[3]
[4]   def insert(self, pos, elem) : self.items.insert(pos, elem)
[5]   def delete(this, pos) : this.items.pop(pos)
[6]   def isEmpty(self): return self.size() == 0
self를 this로 수정
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=>

코드

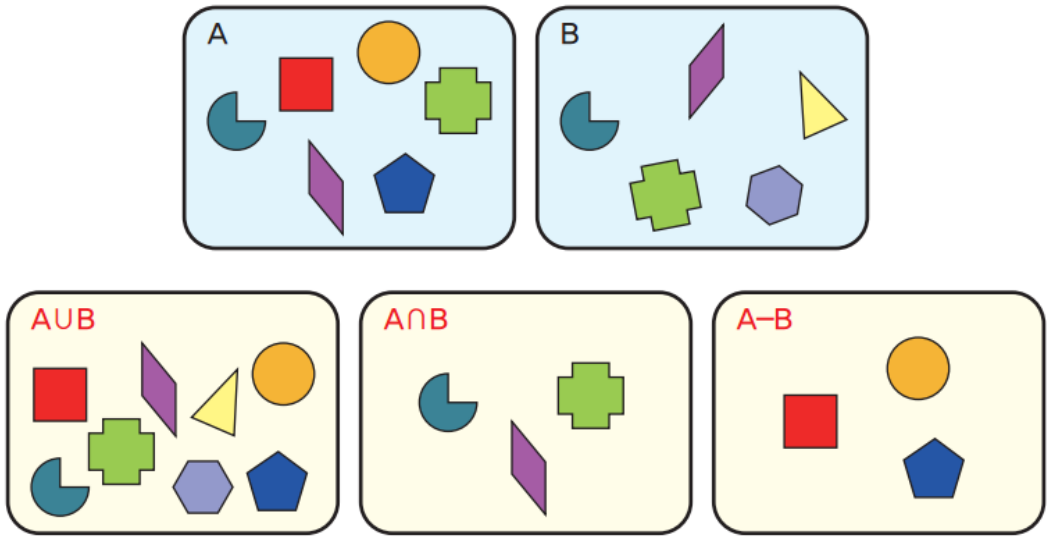
```
def myLineEditor() :                                    # 라인 편집기 주 함수
    list = ArrayList()                                  # 리스트 객체 생성
    while True :
        command = input("[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, q-종료=> ")

        if command == 'i' :                             # 삽입 연산
            pos = int( input("  입력행 번호: "))        # 삽입할 행 번호 입력
            str = input("  입력행 내용: ")              # 삽입할 행 내용 입력
            list.insert(pos, str)                       # insert 메소드로 삽입
        elif command == 'd' :                           # 행 삭제
            pos = int( input("  삭제행 번호: "))        # 삭제할 행 번호 입력
            list.delete(pos)                            # delete 메소드로 삭제
        elif command == 'r' :                           # 행 내용 변경
            pos = int( input("  변경행 번호: "))        # 변경할 행 번호 입력
            str = input("  변경행 내용: ")              # 변경할 행 내용 입력
            list.replace(pos, str)                      # replace로 변경
        elif command == 'q' : return                    # 프로그램 종료
```

3.5~3.6 집합

- 3.5 집합이란?
  - ✓ 집합 ADT
- 3.6 집합의 구현
  - ✓ 집합의 연산들
  - ✓ 테스트 코드
- 연산들의 시간 복잡도?

3.5 집합이란?

- 특징
  - ✓ 원소의 중복을 허용하지 않음
  - ✓ 원소들 사이에 순서가 없음: 선형 자료구조가 아님
  - ✓ $S = \{item_0, item_1, item_2, \cdots, item_{n-1}\}$
- 연산들



집합 ADT

정의 3.2 Set ADT

데이터: 같은 유형의 유일한 요소들의 모임. 원소들은 순서는 없지만 서로 비교할 수는 있어야 함.

연산

- Set(): 비어 있는 새로운 집합을 만든다.
- size(): 집합의 원소의 개수를 반환한다.
- contains(e): 집합이 원소 e를 포함하는지를 검사하고 반환함.
- insert(e): 새로운 원소 e를 삽입함. 이미 e가 있다면 삽입하지 않음.
- delete(e): 원소 e를 집함에서 꺼내고(삭제) 반환한다.
- equals(setB): setB와 같은 집합인지를 검사.
- union(setB): setB와의 합집합을 만들어 반환한다.
- intersect(setB): setB와의 교집합을 만들어 반환한다.
- difference(setB): setB와의 차집합을 만들어 반환한다.
- display(): 집합을 화면에 출력한다.

3.6. 집합의 구현

- 집합은 다양한 방법으로 구현할 수 있음
  - ✓ 리스트, 비트 벡터, 트리, 해싱 구조 등
- 리스트를 이용한 구현

```
class Set:                          # 집합 클래스
    def __init__( self ):           # 생성자
        self.items = []             # 원소를 저장하기 위한 리스트 생성

    def size( self ):               # 집합의 크기
        return len(self.items)      # len()함수 사용
    def display(self, msg):         # 화면에 출력
        print(msg, self.items)      # 메시지 + 집합 내용 출력
```

집합 연산

```
def contains(self, item) :
    return item in self.items
```

```
def contains(self, item) :
    for i in range(len(self.items)):
        if self.items[i] == item :
            return True
    return False
```

```
def insert(self, elem) :
    if elem not in self.items :
        self.items.append(elem)
```

```
def delete(self, elem) :
    if elem in self.items :
        self.items.remove(elem)
```

```
def union( self, setB ):
    setC = Set()
    setC.items = list(self.items)
    for elem in setB.items :
        if elem not in self.items :
            setC.items.append(elem)
    return setC
```

```
def intersect( self, setB ):
    setC = Set()
    for elem in setB.items :
        if elem in self.items :
            setC.items.append(elem)
    return setC
```

```
def difference( self, setB ):
    setC = Set()
    for elem in self.items:
        if elem not in setB.items:
            setC.items.append(elem)
    return setC
```

테스트 코드

```
setA = Set()
setA.insert('휴대폰')
setA.insert('지갑')
setA.insert('손수건')
setA.display('Set A:')

setB = Set()
setB .insert('빗')
setB .insert('파이썬 자료구조')
setB .insert('야구공')
setB .insert('지갑')
setB.display('Set B:')
```

```
setB.insert('빗')
setA.delete('손수건')
setA.delete('발수건')
setA.display('Set A:')
setB.display('Set B:')
```

```
setA.union(setB).display('A U B:')
setA.intersect(setB).display('A ^ B:')
setA.difference(setB).display('A - B:')
```