```
# 리스트

# 리스트는 가장 자유로운 선형 자료구조 이다.

# 리스크의 구현방법
# 배열의 구조와 연결된 구조로 구현할 수 있다.

# 리스트, 선형리스트
# 순서를 가진 항목들의 모임
# 리스트와 달리 집합은 순서가 없는 항목들의 모임이다.

# 리스트의 ADT(추상)
# List() : 비어있는 새로운 리스트를 만든다.
# delete(pos) : pos위치에 있는 요소를 꼬내고(삭제) 반환한다.
# isEmpty() : 리스트가 비어있는지 검사한다.
# getEntry(pos) :  pos 위치에 있는 요소를 반환한다.
# size() : 요소의 갯수를 반환한다.
# clear() : 리스트를 초기화한다.
# find(item) : 리스트에서 item을 찾아 인덱스를 반환한다.
# replace(pos, item) : pos에 있는 항목을 item으로 바꾼다.
# sort() : 정렬
# merge(list) : list를 추가한다.
# display() : 리스트를 화면에 출력한다.
# append(e) : 리스트의 맨 뒤에 새로운 항목을 추가한다.

# 리스크의 내장 매서드
# len(list) : list의 길이를 반환
# list.insert(인덱스, 값) : 인덱스 주소에 값을 삽입한다.
# list.index(찾을값) : 찾는값이 리스크에 있으면 위치 인덱스를 반환
# list.sort() : 리스트 정렬
# list1.extend(list2) : list1뒤에 list2를 연결한다.
# list.pop(val) : 리스트 안에 val값을 뽑아서 반환하고 삭제한다.

# 배열 구조
# 구현이 간단하며 항목 접근이 O(1)이다.
# 삽입, 삭제시 오버헤드이며 항목의 개수 제한

# 연결된 구조
# 구현이 복잡하며 항목 접근이 O(n)이다.
# 삽입, 삭제가 효율적이며 크기가 제한되지 않는다.

# 자료구조 리스트 : 추상적인 의미에 자료구조 리스트를 말하며 앞에서 ADT를 정의하였다.
# 이를 구현하기 위해 배열 구조(파이썬 리스트)나 연결된 구조(연결 리스트)를 사용할 것이다.
# 파이썬 리스트 : C언어에서의 배열이 진화된 형태의 스마트한 배열로 배열 구조의 의미로 사용한다.
# 연결리스트 : 자료들이 일렬로 나열 할 수 있는 연결된 구조를 말하여, 배열 구조와 대응되는 의미로 사용한다.

# 파이썬 리스트
# 파이썬 리스트는 스마트한 배열이다.
# 항목을 추가하면서 용량을 자동으로 늘릴수 있다.
# 용량의 오버하는 값을 집어넣으면, 기존의 배열을 복사하여 크기가 2배인 새로운 배열을 만들고 그곳에 대입하고 사용한다.

# 파이썬 리스트의 시간 복잡도
# append(e) : O(1)
# insert(pos,e) : O(n)
# pop(pos) : O(n)
```

```
# 배열로 구현한 리스트
# 함수 버전 : 전역변수와 함수로 구현

# 함수 선언
items = []

def insert(pos, elem):
    items.insert(pos, elem)

def delete(pos):
    return items.pop(pos)

def getEntry(pos):
    return items[pos]

def isEmpty():
    return len(items)==0

def size():
    return len(items)

def clear():
    global items
    items=[]

def find(item):
    return items.index(item)

def replace(pos, elem):
    items[pos] = elem

def sort():
    items.sort()

def merge(lst):
    items.extend(lst)

def display(msg='ArrayList: '):
    print(msg, size(), items)

# 본문

if __name__ == "__main__":

    display('파이썬 리스트로 구현한 리스트 테스트')

    # insert(0,10); insert(0,20); insert(1,30); inser(size(),40); insert(2,50)
    # 한줄로 쓰려면 ;로 문장을 나누면 된다.
    insert(0,10)
    insert(0,20)
    insert(1,30)
    insert(size(),40)
    insert(2,50)

    display('파이썬 리스트로 구현한 List(삽입*5): ')
```

```
    sort()
    display('파이썬 리스트로 구현한 List(정렬후): ')

    replace(2,90)
    display('파이썬 리스트로 구현한 List(교체*1): ')

    delete(2); delete(size()-1); delete(0)
    display('파이썬 리스트로 구현한 List(삭제*3): ')

    lst=[1,2,3]
    merge(lst)
    display('파이썬 리스트로 구현한 List( 병합 ): ')

    clear()
    display('파이썬 리스트로 구현한 List(정리후): ')

'''
파이썬 리스트로 구현한 리스트 테스트 0 []
파이썬 리스트로 구현한 List(삽입*5):  5 [20, 30, 50, 10, 40]
파이썬 리스트로 구현한 List(정렬후):  5 [10, 20, 30, 40, 50]
파이썬 리스트로 구현한 List(교체*1):  5 [10, 20, 90, 40, 50]
파이썬 리스트로 구현한 List(삭제*3):  2 [20, 40]
파이썬 리스트로 구현한 List( 병합 ):  5 [20, 40, 1, 2, 3]
파이썬 리스트로 구현한 List(정리후):  0 []
'''
```

```
# 배열로 구현한 리스트
# 클래스 버전 : 클래스 변수 선언 및 초기화

class ArrayList:

    def __init__(self): # 생성자
        self.items=[]

    def insert(self, pos, elem):
        self.items.insert(pos, elem)

    def delete(self, pos):
        return self.items.pop(pos)

    def getEntry(self, pos):
        return self.items[pos]

    def isEmpty():
        return self.size()==0

    def size(self):
        return len(self.items)

    def clear(self):
        self.items=[]

    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)

# 본문 출력
if __name__ == "__main__":
    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(삽입*5): ')
    s.display('파이썬 리스트로 구현한 List(삽입*5): ')

    s.sort()
    s.display('파이썬 리스트로 구현한 List(정렬후): ')

    s.replace(2,90)
    s.display('파이썬 리스트로 구현한 List(교체*1): ')

    s.delete(2); s.delete(s.size()-1); s.delete(0)
    s.display('파이썬 리스트로 구현한 List(삭제*3): ')

    lst=[1,2,3]
    s.merge(lst)
    s.display('파이썬 리스트로 구현한 List( 병합 ): ')

    s.clear()
    s.display('파이썬 리스트로 구현한 List(정리후): ')

'''
파이썬 리스트로 구현한 리스트 테스트 항목수= 0 []
파이썬 리스트로 구현한 List(삽입*5):  항목수= 5 [20, 30, 50, 10, 40]
파이썬 리스트로 구현한 List(정렬후):  항목수= 5 [10, 20, 30, 40, 50]
파이썬 리스트로 구현한 List(교체*1):  항목수= 5 [10, 20, 90, 40, 50]
파이썬 리스트로 구현한 List(삭제*3):  항목수= 2 [20, 40]
파이썬 리스트로 구현한 List( 병합 ):  항목수= 5 [20, 40, 1, 2, 3]
파이썬 리스트로 구현한 List(정리후):  항목수= 0 []
'''
```

```
# 리스트의 응용 : 라인편집기

# 라인 편집기의 기능
# i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료

from Array_list_class import ArrayList

def myLineEditor(): # 라인 편집기 주 함수

    list = ArrayList() # ArrayList() 객체 list 생성
    while True:
        command = input("[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> ")

        if command =='i': # 삽입 연산
            pos = int(input("입력행 번호 :"))
            str = input("입력할 내용: ")
            list.insert(pos, str) # insert 메소드로 삽입

        elif command == 'd': # 행 삭제
            pos = int(input("삭제행 번호 :"))
            list.delete(pos) # delete 메소드로 삭제

        elif command == 'r': # 행 변경
            pos = int(input("변경행 번호 :"))
            str = input("변경할 내용: ")
            list.replace(pos, str) # replace 메소드로 변경

        elif command == 'p':
            print("line editor")
            for line in range(list.size()): # size 메소드로 크기 구하기
                print('[%2d] ' % line, list.getEntry(line)) # getEntry 메소드로 단순 값 반환
            print()

        elif command == 'q':
            return

        elif command == 'l':
            filename = input(" 읽어들일 파일 이름: ")
            infile = open(filename, "r")
            lines = infile.readlines();
            for line in lines:
                list.insert(list.size(), line.rstrip('\n'))
            infile.close()

        elif command == 's':
            filename = input("저장할 파일 이름: ")
            openfile = open(filename, 'w')
            for i in range(list.size()):
                openfile.write(list.getEntry(i)+'\n')
            openfile.close()

        elif command == 'f':
            str = input("찾는 문자열: ")
            for line in range(list.size()):
                if list.getEntry(line).find(str) >= 0:
                    print(list.getEntry(line))
```

```
# 본문
if __name__ == "__main__":
    myLineEditor()

'''
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> l
 읽어들일 파일 이름: text.txt
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, 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) : return self.items.pop(pos)
[ 6]      def getEntry(self, pos) : return self.items[pos]

[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> r
변경행 번호 :5
변경할 내용:      def delete(this, pos) : this.items.pop(pos)
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, 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 getEntry(self, pos) : return self.items[pos]

[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> q
'''

'''‘
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> i
입력행 번호 :1
입력할 내용: 자료구조와 알고리즘
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> i
입력행 번호 :2
입력할 내용: 리스트
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> p
line editor
[ 1]  자료구조와 알고리즘
[ 2]  리스트

[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> d
삭제행 번호 :1
[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> p
line editor
[ 1]  리스트

[메뉴선택] i-입력, d-삭제, r-변경, p-출력, l-파일읽기, s-저장, f-찾기, q-종료=> q
'''
```

```
# 집합이란
# 원소의 중복을 허용하지 않으며 원소들 사이의 순서가 없다는 특징을 가진다.
# 선형 자료 구조가 아니다.

# 집합은 다양한 방법으로 구현할 수 있다.
# 리스트, 비트 벡터, 트리, 해싱구조 등

# 집합의 ADT
# Set() : 비어있는 새로운 집합을 만든다.
# size() : 집합의 원소의 개수를 반환한다.
# contains(e) : 집합이 원소 e를 포함하는 지를 검사하고 반환함
# insert(e) : 새로운 우너소 e를 삽입함. 이미 e가 있다면 삽입하지 않음
# delete(e) : 원소 e를 집합에서 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.

# 집합을 class를 사용한 리스트로 구현
class Set:
    def __init__( self ):
        self.items = []

    def size( self ):
        return len(self.items)

    def display(self, msg):
        # 메시지와 함께 출력
        print(msg, self.items)

    def contains(self, item): # item이 있으면 T, 없으면 F
        for i in range(len(self.items)):
            if self.items[i] == item:
                return True
        return False
        # 혹은 return item in self.items

    def insert(self, elem):
        if elem not in self.items:
            self.items.append(elem) # 순서가 중요하지 않으니 뒤에 추가

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

    def union( self, setB):
        setC = Set()
        # self리스트를 setC으로 넘길때 캐스팅을 해야지 값이 제대로 전달된다.
        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

# 본문
if __name__ == "__main__":
    setA = Set()
    setA.insert('휴대폰')
    setA.insert('지갑')
    setA.insert('손수건')
    setA.display('SetA :')

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

    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:')

'''
SetA : ['휴대폰', '지갑', '손수건']
SetB : ['빗', '파이썬 자료구조', '야구공', '지갑']
Set A: ['휴대폰', '지갑']
Set B: ['빗', '파이썬 자료구조', '야구공', '지갑']
A U B: ['휴대폰', '지갑', '빗', '파이썬 자료구조', '야구공']
A ^ B: ['지갑']
A - B: ['휴대폰']
'''
```