파이썬 프로그래밍

22차시

중복과 순서가 없는 집합

⚠ 학습개요

- … 집합의 이해
- … 집합 생성
- … 집합 원소의 제한
- … 집합의 다양한 메소드

⚠ 학습목표

… 집합의 특징을 이해할 수 있다.

… 집합을 생성할 수 있다.

… 집합 원소는 immutable 객체여야 한다.

… 집합의 다양한 메소드를 활용할 수 있다.

… 합집합 등 집합의 기본 연산을 활용할 수 있다.

Chapter 1.

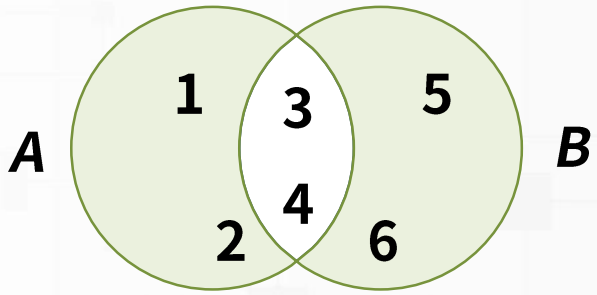
집합의 이해

PYTHON PROGRAMMING

⚠ 원소는 유일하고 순서는 의미 없는 집합

+ 집합은 중복되는 요소가 없으며, 순서도 없는 원소의 모임(collections)

- 집합 A, B의 관계를 이해하기 쉽도록 표현한 **벤 다이어그램**
- 원소를 콤마로 구분하며 중괄호로 둘러싸 표현

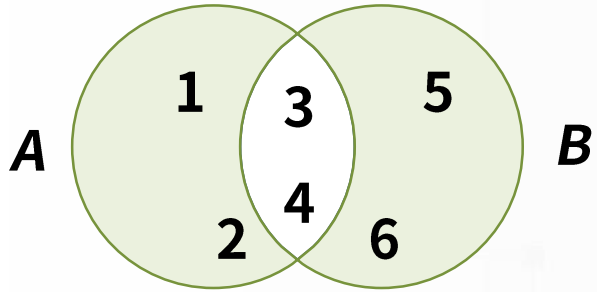


A={1, 2, 3, 4}　　B={3, 4, 5, 6}

[그림22-1] 수학의 집합과 벤 다이어그램

⚠ 원소는 유일하고 순서는 의미 없는 집합

✚ 수학의 집합과 벤 다이어그램



A={1, 2, 3, 4}　　B={3, 4, 5, 6}

{원소 1, 원소 2, ... 원소 n}

- 원소는 불변 값으로 중복될 수 없으며 서로 다른(unique) 값이어야 한다.
- 즉, 원소는 중복을 허용하지 않으며 원소의 순서는 의미가 없다.

```
s1 = {1, 2, 3, 4, 5}
s2 = {'py', 'java', 'go'}
s3 = {(1, 1), (2, 4), (3, 9)}
```

끝

Chapter 2.

집합 생성

PYTHON PROGRAMMING

⚠ 내장 함수 set()을 활용한 집합 생성

✚ 함수 set() 호출로 공집합 만들기

```
>>> s = set()
type(s)
<class 'set'>

>>> s
set()

d = {}
>>> type(d)
<class 'duct'>
```

```
>>> set([1, 2, 3])
{1, 2, 3}
>>> set([1, 2, 2])
{1, 2}
>>> set(['a', 'b']) # 'a', 'b'가 원소로 구성되는 집합
{'b', 'a'}

>>> set('abc')
{'c', 'b', 'a’}

>>> set(['abc'])
{‘abc'}
```

⚠ 내장 함수 set()을 활용한 집합 생성

✚ 수정 가능한 리스트와 딕셔너리는 집합의 원소로 사용 불가

```
>>> set([1, [2, 3]]) # 오류
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError : unhashable type: 'list'
```

⚠ 중괄호로 직접 원소를 나열해 집합 생성

✚ {원소 1, 원소 2, …}로 생성

```
>>> {1, 2, 3}
{1, 2, 3}
>>> {1, 'seoul', 'a', (1.2, 3.4)}
{1, (1.2, 3.4), 'seoul', 'a'}
```

✚ {원소 1, 원소 2, …}에서 리스트나 딕셔너리는 원소로 사용 불가

```
>>> {['a', 'b']} # 오류
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
>>> {{1:'a', 2:'b'}} # 오류
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'dict'
```

⚠ 일상 코딩: 한 글자 단어로 구성된 집합 만들기

[코딩실습] 한 글자 단어의 집합 만들기

난이도 기본

```
1  planets = set('해달별')
2  fruits = set(['감', '귤'])
3  nuts = {'밤', '잣'}
4  things = {('밤', '잣'), ('감', '귤'), '해달'}
5  # things = {['밤', '잣'], ('감', '귤'), '해달'} # 오류 발생
6
7  print(planets)
8  print(fruits)
9  print(nuts)
10 print(things)
```

주의 5번 줄의 [‘밤’, ‘잣’]과 같은 리스트는 원소로 사용할 수 없다.

⚠ 일상 코딩: 한 글자 단어로 구성된 집합 만들기

[코딩실습] 한 글자 단어의 집합 만들기 난이도 기본

결과

```
{'해', '달', '별'}
{'귤', '감'}
{'잣', '밤'}
{'해달', ('감', '귤'), ('밤', '잣')}
```

Chapter 3.

집합 원소의 제한

PYTHON PROGRAMMING

⚠ 집합의 원소 추가와 삭제

+ 집합 메소드 add(원소)로 추가

```
>>> odd = {1, 3, 5}
>>> odd.add(7)
>>> odd.add(9)
>>> print(odd)
{1, 3, 5, 7, 9}
```

+ 집합 메소드 remove(원소)

```
>>> odd = {1, 3, 5}
>>> odd.remove(3)
>>> print(odd)
{1, 5}
>>> odd.remove(9)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
keyError: 9
```

⚠ 집합의 원소 추가와 삭제

+ discard(원소)

- Remove an element from a set if it is a member.
- If the element is not a member, do nothing.

```
>>> odd = {1, 3, 5}
>>> odd.discard(3)
>>> print(odd)
{1, 5}
>>> odd.discard(9)
```

+ pop( )

```
>>> odd = {1, 3, 5}
>>> print(odd.pop())
1
>>> print(odd)
{3, 5}
```

+ clear()

```
>>> odd = {1, 3, 5}
>>> odd.clear()
>>> print(odd)
set()
```

⚠ 일상 코딩: 로또 번호를 집합을 이용해 생성

[코딩실습] 로또 번호를 randrange()와 sample() 함수로 생성 난이도 응용

```
1  from random import randrange
2  from random import sample
3
4  # randrange() 함수와 집합을 이용, 중복을 제거
5  mylotto = set()
6  while True:
7      num = randrange(1, 46)
8      print(num, end=' ')
9      mylotto.add(num)
10     if len(mylotto) == 6:
11         break
12 print()
```

⚠ 일상 코딩: 로또 번호를 집합을 이용해 생성

[코딩실습] 로또 번호를 randrange()와 sample() 함수로 생성 난이도 응용

```
13 print('집합: {}'.format(mylotto))
14 print('정렬리스트: {}'.format(sorted(mylotto)))
15 print()
16
17 # sample() 함수를 이용하면 매우 간편
18 lotto = sample(range(1, 46), 6)
19 print('sample 함수 리스트: {}'.format(lotto))
20 print('sample 함수 정렬리스트: {}'.format(sorted(lotto)))
```

주의

1~2번 줄은 모듈 random에서 함수 randrange()와 sample()을 사용하기 위한 문장이다.

⚠ 일상 코딩: 로또 번호를 집합을 이용해 생성

[코딩실습] 로또 번호를 randrange()와 sample() 함수로 생성 난이도 응용

결과

```
14 29 30 15 10 14 43
집합: {10, 43, 14, 15, 29, 30}
정렬리스트: [10, 14, 15, 29, 30, 43]

sample 함수 리스트: [1, 34, 39, 17, 29, 27]
sample 함수 정렬리스트: [1, 17, 27, 29, 34, 39]
```

Chapter 4.

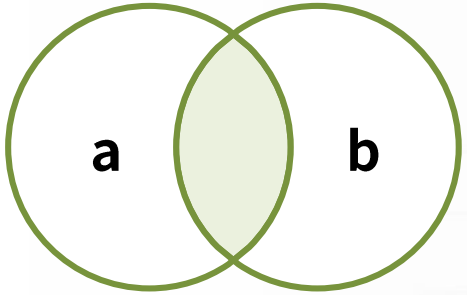
집합의 다양한 메소드

PYTHON PROGRAMMING

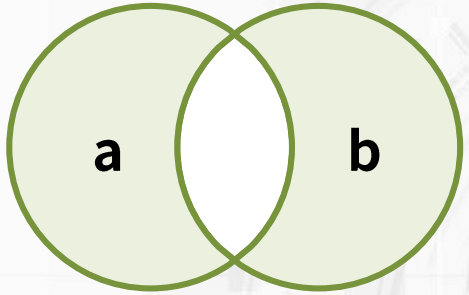
⚠ 합집합

+ 집합 연산과 벤 다이어그램

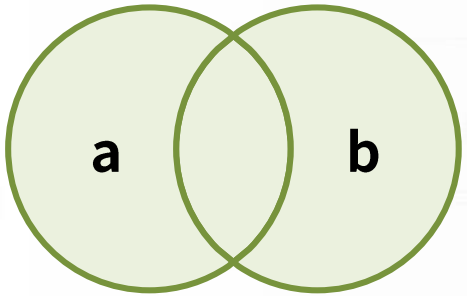
- a & b a.intersection(b)



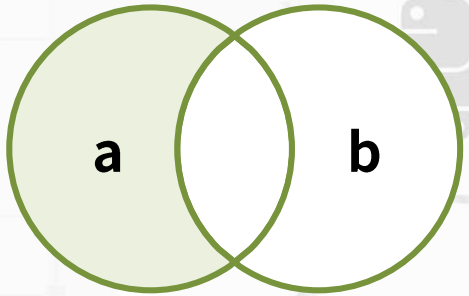
- a ^ b a.symmetric_difference(b)



- a | b a.union(b)



- a - b a.difference(b)



[그림22-2] 집합의 벤 다이어그램

⚠ 합집합

+ 합집합 연산자 | 와 메소드 union( ), update( )

```
>>> a = {4, 6, 8, 10, 12} # 4에서 12까지의 짝수
>>> b = {3, 6, 9, 12} # 3에서 12까지의 3의 배수
>>> a | b
{3, 4, 6, 8, 9, 10, 12}
>>> a.union(b)
{3, 4, 6, 8, 9, 10, 12}
>>> a
{4, 6, 8, 10, 12}

>>> a.update(b)
>>> a
{3, 4, 6, 8, 9, 10, 12}
```

- **a.update(b)**
- 합집합의 결과가 a에 반영

⚠ 교집합과 차집합

+ 교집합 연산자 &와 메소드 intersection( )

```
>>> a = {4, 6, 8, 10, 12} # 4에서 12까지의 짝수
>>> b = {3, 6, 9, 12} # 3에서 12까지의 3의 배수
>>> a &b
{12, 6}
>>> a.intersection(b)
{12, 6}

>>> a.intersection_update(b)
>>> a
{12, 6}
```

⚠ 교집합과 차집합

+ 차집합 연산자 -와 메소드 difference( )

```
>>> a = {4, 6, 8, 10, 12} # 4에서 12까지의 짝수
>>> b = {3, 6, 9, 12} # 3에서 12까지의 3의 배수
>>> a - b
{8, 10, 4}
>>> a.difference(b)
{8, 10, 4}
>>> b - a
{9, 3}
```

⚠ 여집합 등

✚ 여집합 연산자 ^와 메소드 symmetric_difference( )

```
>>> a ^ b
{3, 4, 8, 9, 10}
>>> a.symmetric_difference(b)
{3, 4, 8, 9, 10}
```

✚ 집합 연산의 축약 대입 연산자 |=, &=, -=, ^=와 메소드 intersection_update( ) 등

```
>>> A = set('abcd'); B = set('cde')
>>> A |= B #A = A | B, A.update(B)
>>> A
{'b', 'a', 'c', 'e', 'd'}

>>> A = set('abcd'); B = set('cde')
>>> A &= B #A = A & B, A. intersection_update(B)
>>> A
{'c', 'd'}
```

⚠ 여집합 등

✚ 집합 연산의 축약 대입 연산자 |=, &=, -=, ^=와 메소드 intersection_update( ) 등

```
>>> A = set('abcd'); B = set('cde')
>>> A.intersection_update(B) # A = A & B, A &= B
>>> A
{'c', 'd'}

>>> A = set('abcd'); B = set('cde')
>>> A -= B # A = A - B, A.difference_update(B)
>>> A
{'b', 'a'}

>>> A = set('abcd'); B = set('cde')
>>> A ^= B # A = A ^ B, A.symmetric_difference_update(B)
>>> A
{'b', 'a', 'e'}
```

⚠ 일상 코딩: 요일로 구성된 집합의 연산

[코딩실습] 요일 문자열 원소로 구성된 집합 연산 수행 난이도 응용

```
1  daysA = {'월', '화', '수', '목'}
2  daysB = set(['수', '목', '금', '토', '일'])
3  weekends = set(('토', '일'))
4
5  alldays = daysA | daysB
6  print(alldays)
7
8  workdays = alldays - weekends
9  print(workdays)
10
11 print(daysA & daysB)
12 print(daysA.symmetric_difference(daysB))
```

주의

1~3번 줄에서와 같이 중괄호 {…}와 원소를 사용할 때와 함수 set()을 이용할 때의 구문을 이해하자.

⚠ 일상 코딩: 요일로 구성된 집합의 연산

[코딩실습] 요일 문자열 원소로 구성된 집합 연산 수행 난이도 응용

결과

```
{'일', '토', '목', '월', '화', '금', '수'}
{'목', '월', '화', '금', '수'}
{'수', '목'}
{'월', '일', '금', '화', '토'}
```

SUMMARY

⚠집합의 이해

⚠집합 생성

⚠집합 원소의 제한

…수정 가능한 리스트와 딕셔너리는 집합의 원소로 사용 불가

⚠ 딕셔너리의 이해

… add(), remove(), discard(), pop(). clear()

… union(), update()

… intersection(), intersection_update()

… difference(), difference_update()

… symmetric_difference(), symmetric_difference_update()