

# 자료구조와 순차자료형

# 자료구조와 순차자료형

- ▶ 파이썬은 단순하지만 강력한 자료구조를 내부 구현하고 있음
  - ▶ 자료구조의 사용법을 숙지하고 자유롭게 활용하는 것이 파이썬을 강력한 도구로 만들어 준다
- ▶ 순차 자료형의 주요 연산

연산	설명	예제
+	연결	"Pyt" + "hon"
*	반복	"Python" * 2
len()	길이	len("Python")
in	포함 여부	"P" in "Python"
not in	불포함 여부	"r" in "Python"

# 자료구조와 순차자료형

## : 리스트(list)

### ▶ 리스트

- ▶ 파이썬의 가장 기본적인 데이터 구조
- ▶ 순서가 있는 자료의 집합, 타 언어의 배열(array)과 유사하지만, 좀 더 풍부한 기능을 제공한다

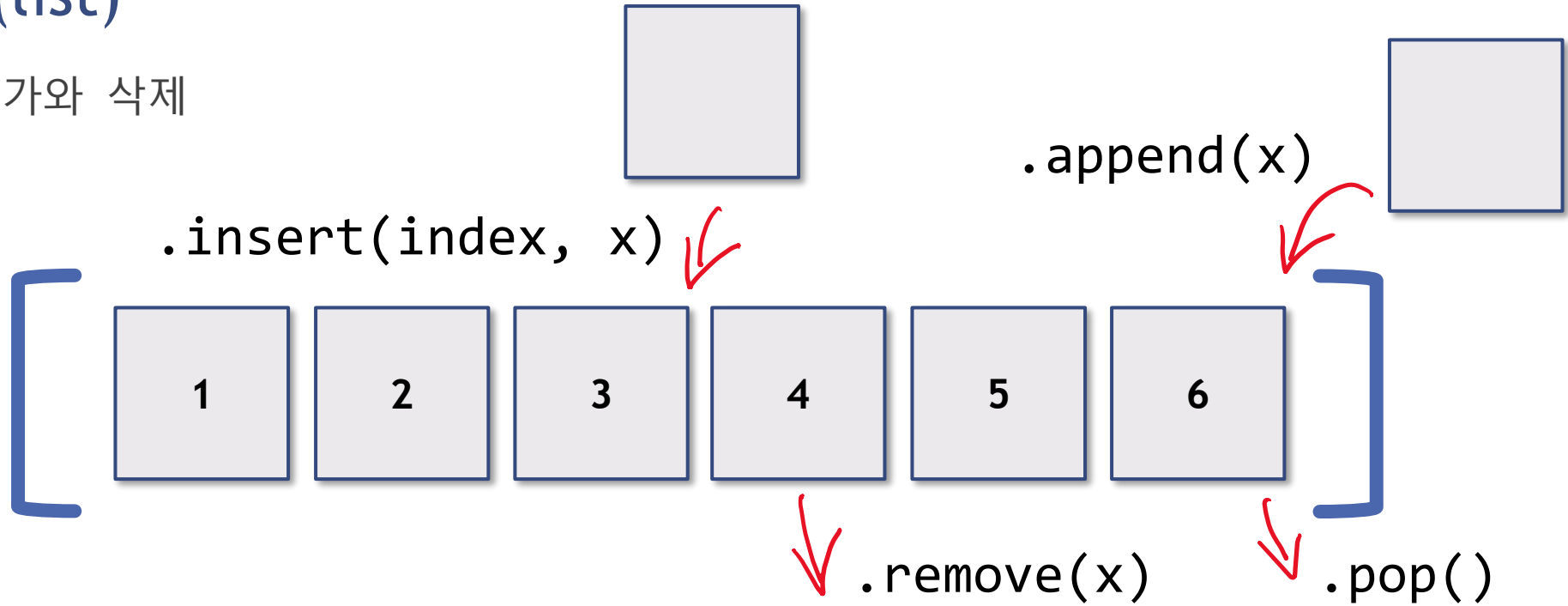
### ▶ 리스트의 특징

- ▶ `list()` 타입 함수 또는 `[]` 기호로 선언
- ▶ 순차자료형 : 모든 시퀀스 연산(`len`, `in`, `not in`, 인덱싱, 슬라이싱, 연결, 반복) 가능
- ▶ 변경 가능(`mutable`) 자료형: 항목의 추가, 변경, 삭제 모두 가능

# 자료구조와 순차자료형

## : 리스트(list)

- ▶ 요소의 추가와 삭제

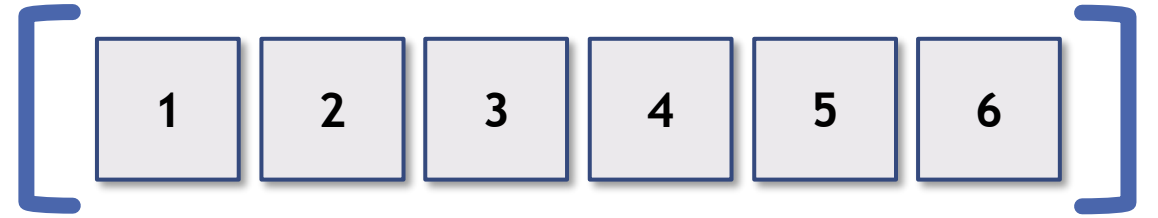
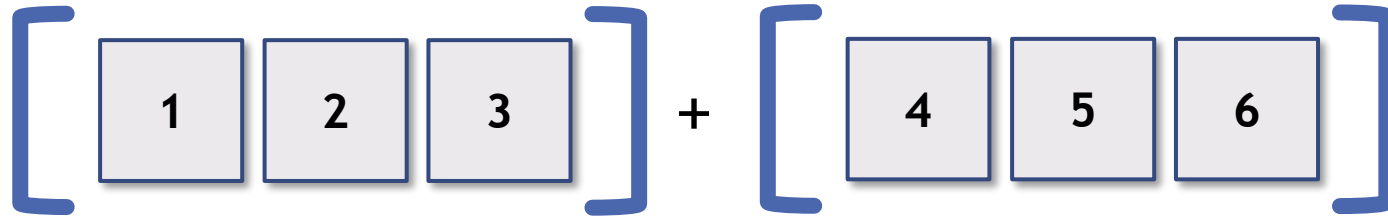


# 자료구조와 순차자료형

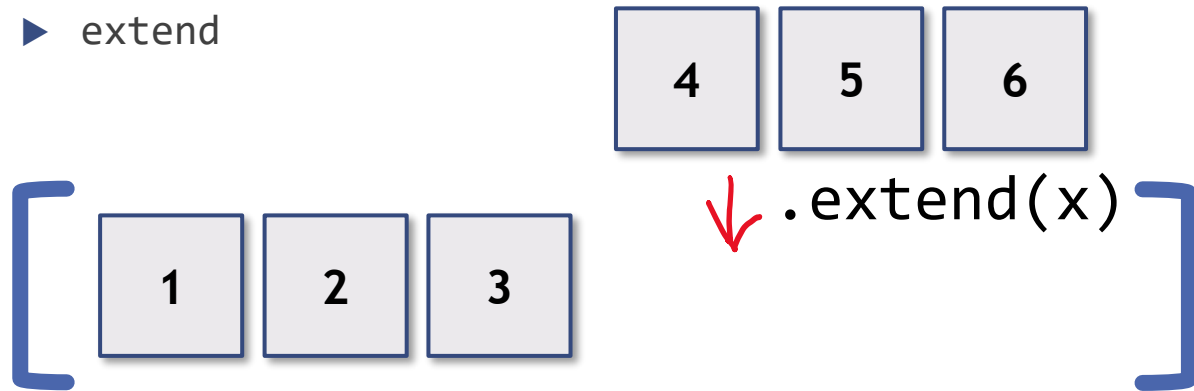
## : 리스트(list)

▶ 연결 연산자 (+) vs .extend()

▶ 연결 연산



▶ extend



# 자료구조와 순차자료형

## : 리스트(list)

- ▶ 인덱스 접근
  - ▶ `[index]`로 내부 저장 객체에 접근 가능
  - ▶ 파이썬 인덱스는 0부터 시작(zero-based index)
  - ▶ 역방향 인덱스가 가능

**`[-6] [-5] [-4] [-3] [-2] [-1]`**

a	b	c	d	e	f
---	---	---	---	---	---

**`[0] [1] [2] [3] [4] [5] [6]`**

# 자료구조와 순차자료형

## : 리스트(list)

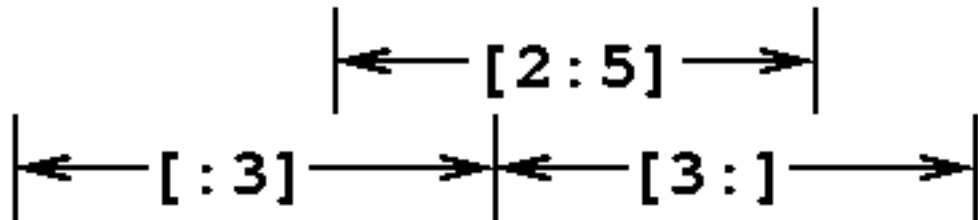
### ▶ 슬라이싱

- ▶ [시작인덱스:끝경계]
- ▶ [시작인덱스:끝경계:간격]

**[-6] [-5] [-4] [-3] [-2] [-1]**

a	b	c	d	e	f
---	---	---	---	---	---

**[0] [1] [2] [3] [4] [5] [6]**

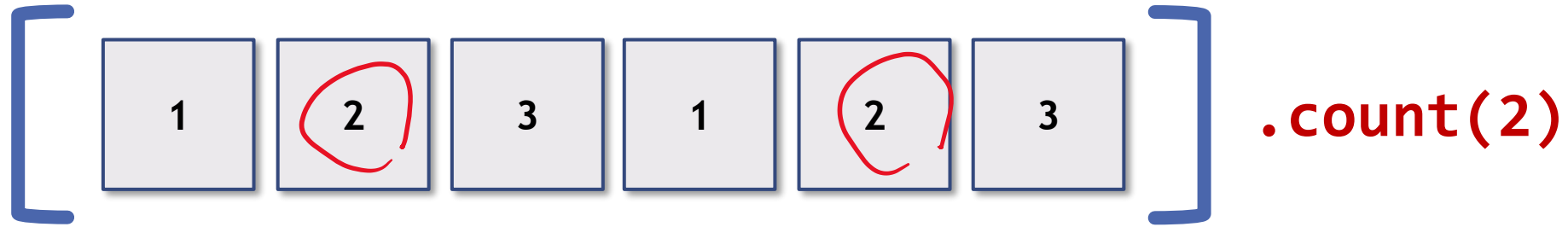


# 자료구조와 순차자료형

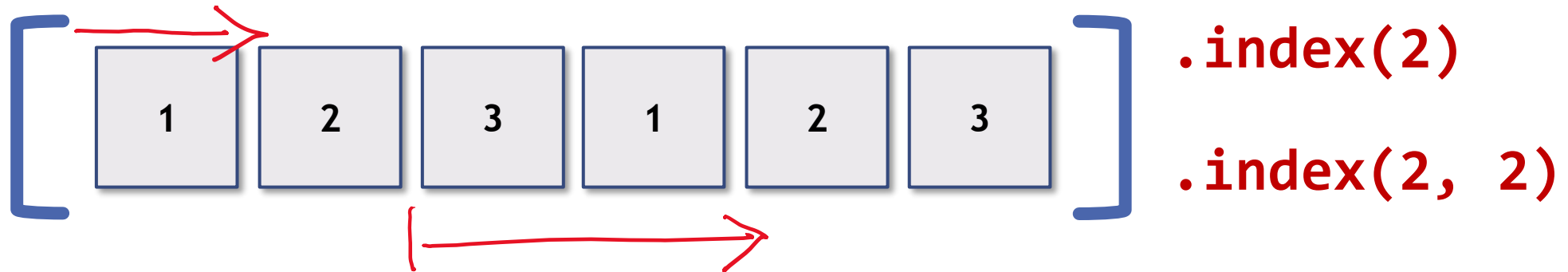
## : 리스트(list)

### ▶ 검색

▶ 내부의 요소 개수 검색: `.count()`



▶ 내부 객체의 위치(인덱스) 검색: `.index()`





# 자료구조와 순차자료형

## : 튜플(tuple)

- ▶ 변경 불가능한 리스트
  - ▶ 리스트 자료형에서 수정을 제외한 모든 기능을 튜플에 적용할 수 있음
  - ▶ tuple 타입 함수 또는 괄호(혹은 아무 기호 없이) 정의하여 사용
  - ▶ 하나의 요소만을 가질 때는 요소 뒤에 콤마(,)를 반드시 붙임

# 자료구조와 순차자료형

## : 튜플(tuple)

- ▶ 튜플의 패킹과 언패킹

- ▶ Packing : 나열된 객체를 Tuple로 변환하는 것
- ▶ Unpacking : 튜플, 리스트 내의 객체를 변수로 할당하는 것

# 자료구조와 순차자료형

## : 튜플(tuple)

### ▶ 확장 Unpacking

- ▶ Unpacking시 좌측의 변수 개수가 부족한 경우, 에러가 발생(ValueError)
- ▶ 확장 Unpacking에서는 왼쪽 변수가 적은 경우에도 적용할 수 있다

# 자료구조와 순차자료형

## : 집합(set)

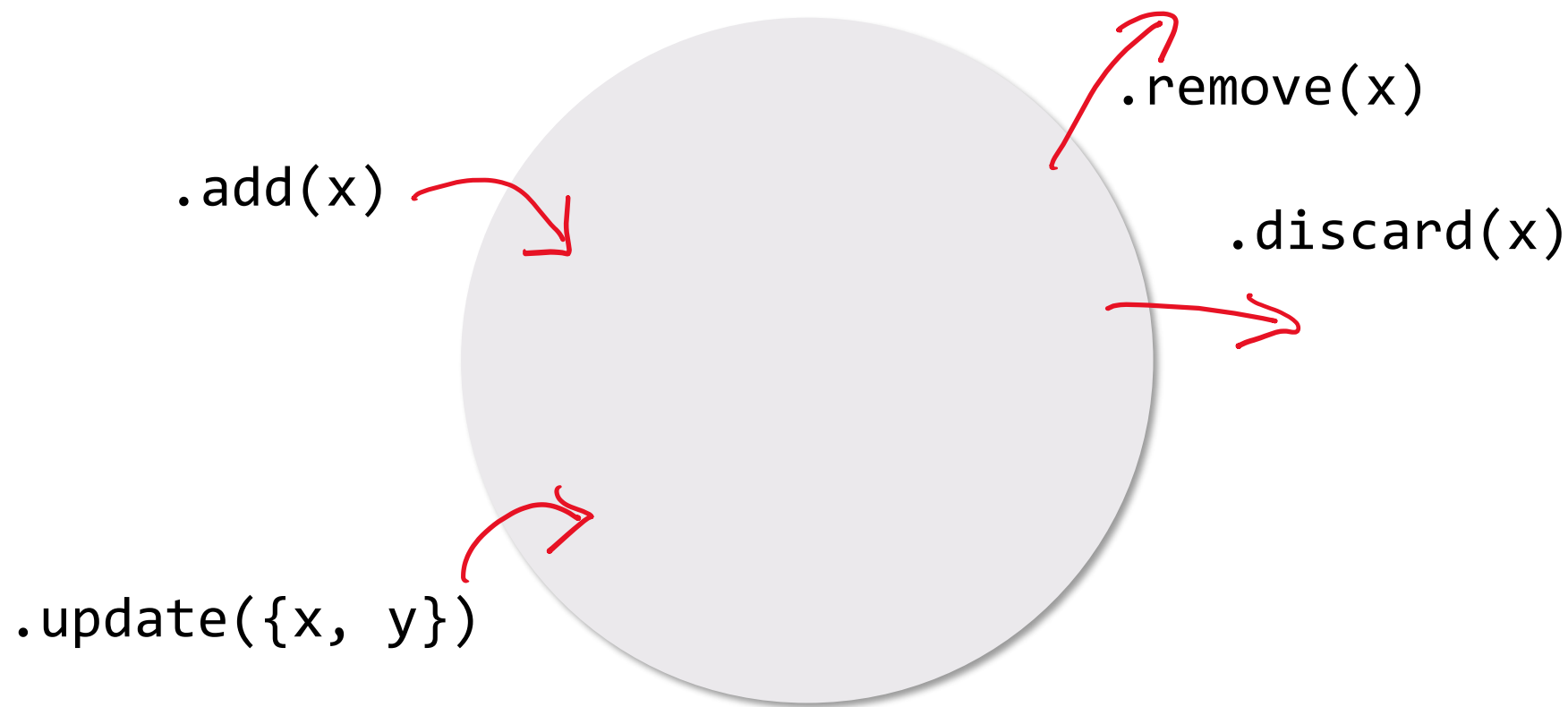
### ▶ set의 특징

- ▶ 순서가 없고 중복이 없는 객체들의 집합(non sequence)
  - ▶ len(), in, not in 정도만 가능
- ▶ set 함수 혹은 {} 기호로 정의, 단 빈 Set의 경우, {}로 정의 불가함
- ▶ 수학의 집합을 표현할 때 사용
- ▶ 세트의 내용은 불변이어야 한다(immutable)

# 자료구조와 순차자료형

## : 집합(set)

### ▶ set의 메서드



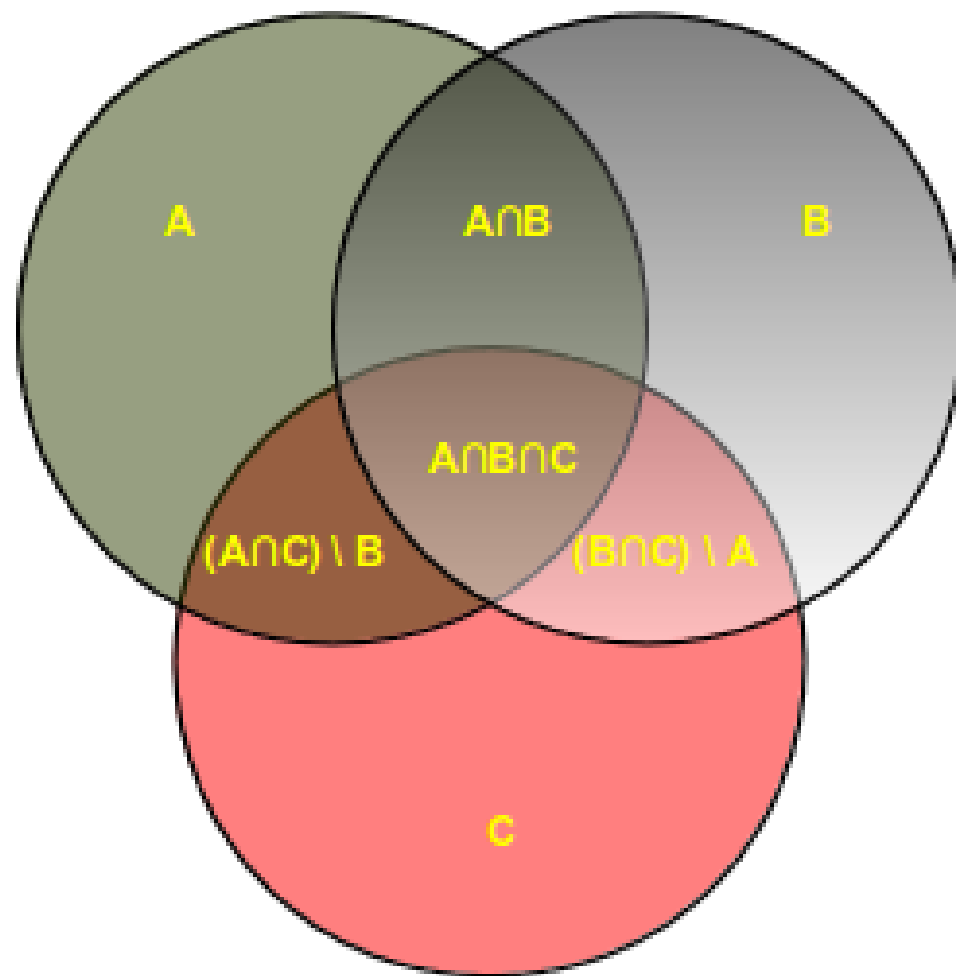
# 자료구조와 순차자료형

## : 집합(set)

### ▶ 집합의 연산

- ▶ 교집합, 합집합, 차집합
- ▶ 모집합, 부분집합 여부 확인

	연산자	메서드
교집합 (set)	$a \ \& \ b$	<code>a.intersection(b)</code>
합집합 (set)	$a \   \ b$	<code>a.union(b)</code>
차집합 (set)	$a - b$	<code>a.difference(b)</code>
모집합 (bool)		<code>a.issuperset(b)</code>
부분집합 (bool)		<code>a.issubset(b)</code>



# 자료구조와 순차자료형

## : 간단한 통계 함수

- ▶ 순차자료형의 경우, min, max, sum 등 간단한 통계 함수를 적용할 수 있다

# 자료구조와 순차자료형

## : 내장 순차 자료형 함수

### ▶ range

- ▶ 연속된 정수를 넘겨주는 이터레이터를 반환

```
range({start = 0, } stop {, step = 1})
```

- ▶ start부터 stop까지 step 간격으로 연속된 정수를 생성



# 자료구조와 순차자료형

## : 내장 순차 자료형 함수

### ▶ enumerate

- ▶ 순차자료형에서 현재 아이템의 색인을 함께 처리하고자 할 때 흔히 사용

```
for item in enumerate([P, y, t, h, o, n]):  
    item == (0, 'P')  
    item == (1, 'y')  
    ...
```

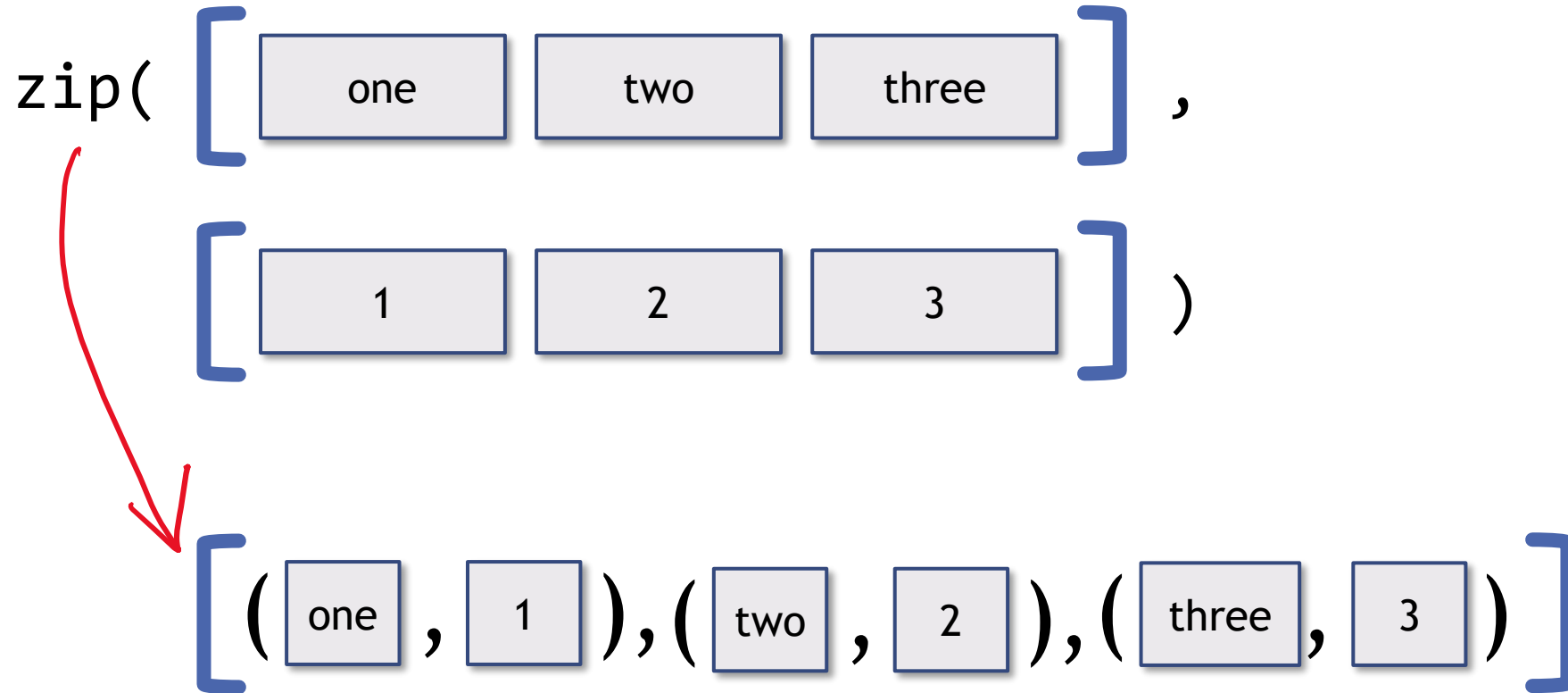
```
for i, v in enumerate([P, y, t, h, o, n]):  
    i == 0, v == 'P'  
    i == 1, v == 'y'  
    ...
```

# 자료구조와 순차자료형

## : 내장 순차 자료형 함수

### ▶ zip

- ▶ 여러 개의 리스트나 다른 순차자료형을 짝지어 튜플의 리스트를 생성

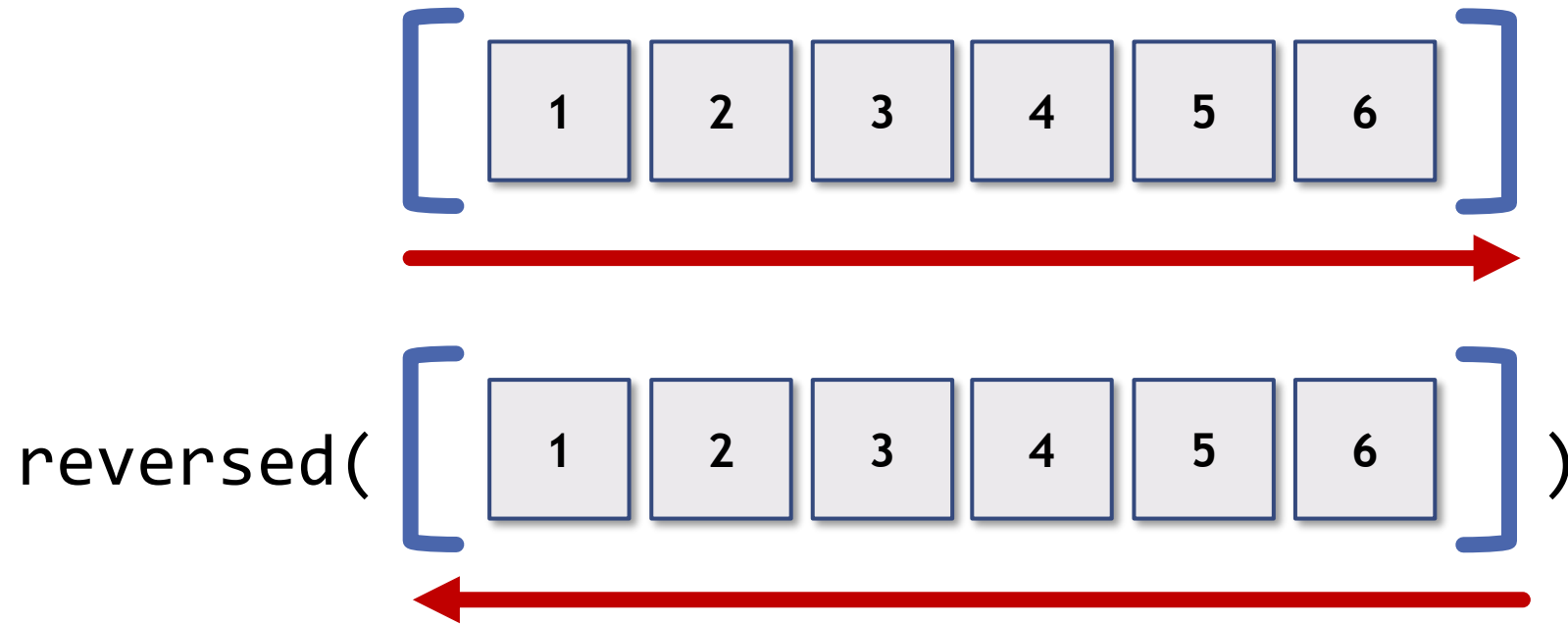


# 자료구조와 순차자료형

## : 내장 순차 자료형 함수

### ▶ reversed

- ▶ 순차 자료형을 역순으로 순회(Generator)



- ▶ `list`의 `reverse()` 메서드와 차이점은 `sorted` 함수는 원본 자료를 변경하지 않고 정렬된 새 순차자료형 객체를 반환한다는 점

# 자료구조와 순차자료형

## : 내장 순차 자료형 함수

### ▶ sorted

- ▶ 순차자료형을 정렬하여 새로운 순차자료형을 생성한다
- ▶ 정렬을 위한 기준을 정의하려면 key 인수에 정렬 기준이 되는 key 함수를 전달할 수 있다
- ▶ 기본적으로는 오름차순 정렬이지만, reverse 인수를 True로 부여하면 역순으로 정렬할 수 있다
- ▶ list의 sort 메서드와의 차이점은 sorted 메서드는 정렬된 결과를 새로운 순차형으로 생성하여 반환한다는 점

# 자료구조와 순차자료형

## : 사전(dict)

### ▶ dict

- ▶ 순서를 가지지 않는 객체의 집합(non sequence)
  - ▶ len(), in, not in 정도만 가능
  - ▶ 기준은 key
- ▶ 인덱스 기반이 아니라 key를 기반으로 값을 지정하고 참조하는 매핑형 자료형
- ▶ dict() 타입 함수 혹은 {} 기호로 정의할 수 있다

# 자료구조와 순차자료형

## : 사전(dict)

▶ dict는 키 목록(dict\_keys)과 값 목록(dict\_values)이 합쳐진 복합 자료형이다

▶ 내용 참조

▶ 키를 이용한 값 참조

```
dct['KEY1']
```

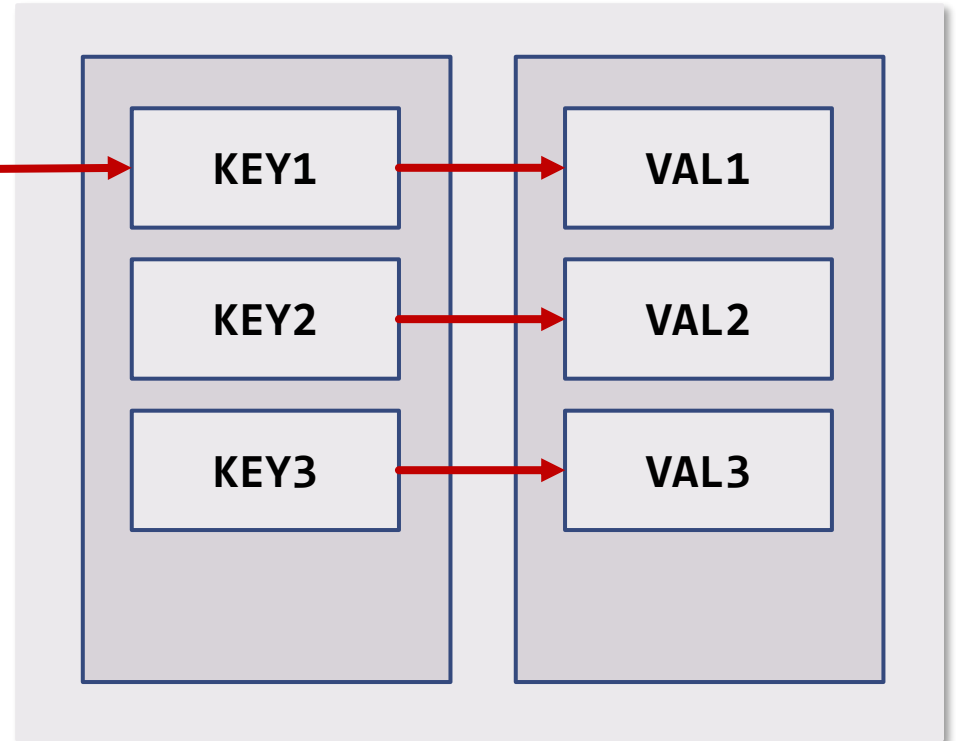
- ▶ 사전의 키는 해싱해야 하기 때문에 수정 불가(`immutable`) 객체여야 한다
- ▶ 없는 키를 참조할 경우 `KeyError`가 발생

▶ 키 삭제

▶ 키를 삭제하면 대응하는 값도 삭제된다

```
del dct['KEY1']
```

dct



# 자료구조와 순차자료형

## : 사전(dict)

### ▶ 내용 참조

#### ▶ get 메서드를 이용한 값 참조

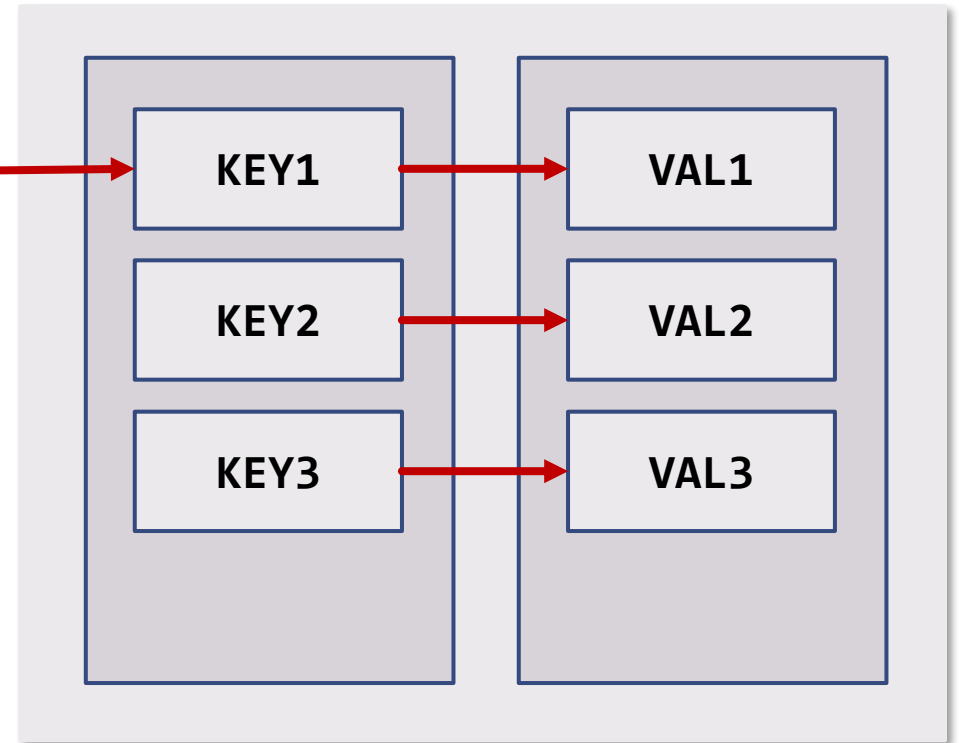
```
dct['KEY1']
```

- ▶ 키를 이용한 접근과 달리 key에 대응하는 값이 없을 경우 None을 반환
- ▶ default 값을 부여하면 key에 대응하는 값이 없을 경우 default를 반환

```
dct.get('KEY4', 'Not Found')
```

Not Found

dct



# 자료구조와 순차자료형

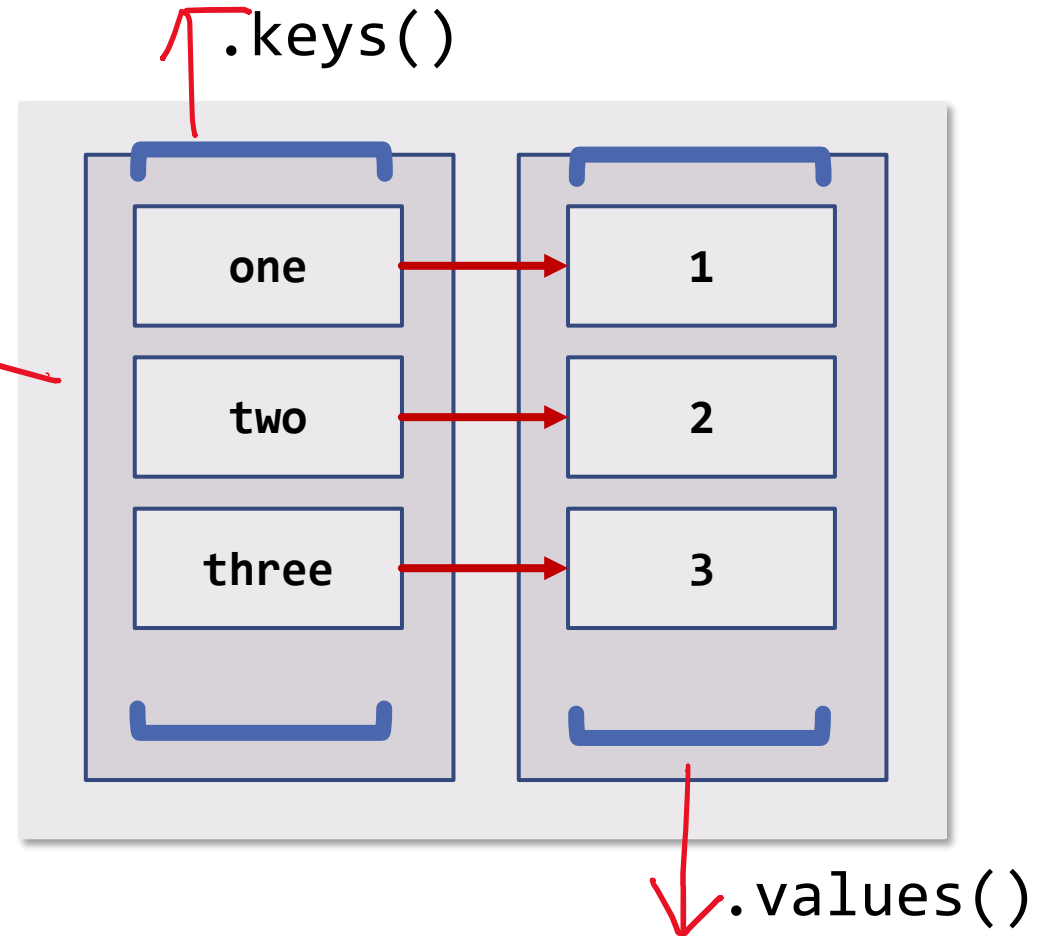
## : 사전(dict)

- ▶ 기본적으로 사전의 순회, `len()`, `in`, `not in` 등의 연산자는 키의 목록(`dict_keys`)을 기준으로 수행된다

[ (one, 1), (two, 2), (three, 3) ]

.items()

메서드	설명
<code>keys()</code>	키 목록을 반환( <code>dict_keys</code> 객체)
<code>values()</code>	값 목록을 반환( <code>dict_values</code> 객체)
<code>items()</code>	키-값 쌍 튜플 목록을 반환( <code>dict_items</code> 객체)






# 자료구조와 순차자료형

## : 축약(Comprehension)

- ▶ 리스트, 집합, 딕셔너리 축약을 이용하면 간결한 표현으로 새로운 리스트를 만들 수 있다
  - ▶ 리스트 축약 : 필터링 조건은 생략 가능



```
[ expr for val in collection if condition ]
```


필터링 조건

# 자료구조와 순차자료형

## : 축약(Comprehension)

### ▶ 셋 축약

```
{ expr for val in collection if condition }
```



필터링 조건

# 자료구조와 순차자료형

## : 축약(Comprehension)

### ▶ 딕셔너리 축약

```
{ key-expr:val-expr for val in collection if condition }
```



필터링 조건