데이터 과학을 위한
파이썬 프로그래밍

2판

Chapter 07
자료구조

목차

학습목표

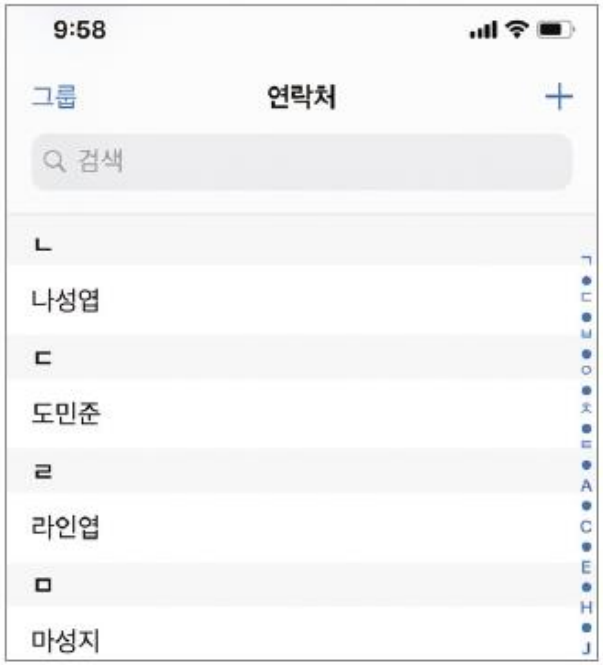
- 파이썬에서의 자료구조에 대해 이해한다.
- 스택, 큐, 튜플, 세트에 대해 학습한다.
- 파이썬에서의 딕셔너리에 대해 알아본다.
- collections 모듈에 대해 이해한다.

01
자료구조의 이해

1. 자료구조의 개념

- **자료구조(data structure):** 데이터의 특징을 고려하여 저장하는 방법
- 실생활 속 데이터 저장 사례: 전화번호부

(a) 전화번호부



(b) 휴대전화의 연락처

그림 7-1 실생활 속 자료구조

2. 파이썬에서의 자료구조

표 7-1 파이썬에서 제공하는 자료구조

자료구조명	특징
스택(stack)	나중에 들어온 값이 먼저 나갈 수 있도록 해주는 자료구조(last in first out)
큐(queue)	먼저 들어온 값이 먼저 나갈 수 있도록 해주는 자료구조(first in first out)
튜플(tuple)	리스트와 같지만 데이터의 변경을 허용하지 않는 자료구조
세트(set)	데이터의 중복을 허용하지 않고, 수학의 집합 연산을 지원하는 자료구조
딕셔너리(dictionary)	전화번호부와 같이 키(key)와 값(value)의 형태로 데이터를 저장하는 자료구조이며 여기서 키값은 다른 데이터와 중복을 허용하지 않음
collections 모듈	위에 열거된 여러 자료구조를 효율적으로 사용할 수 있도록 지원하는 파이썬 내장(built-in) 모듈

02
스택과 큐

1. 스택

- **스택(stack):** 컴퓨터공학과 학생들이 전공을 시작하면서 처음 배우는 자료구조로, 자료구조의 핵심 개념 중 하나
- **스택 자료구조(stack data structure):** 4, 10과 같은 데이터를 저장하는 공간, 리스트와 비슷하지만 저장 순서가 바뀌는 형태
- **푸시(push):** 스택에 데이터를 저장하는 것
- **팝(pop):** 데이터를 추출하는 것

1. 스택

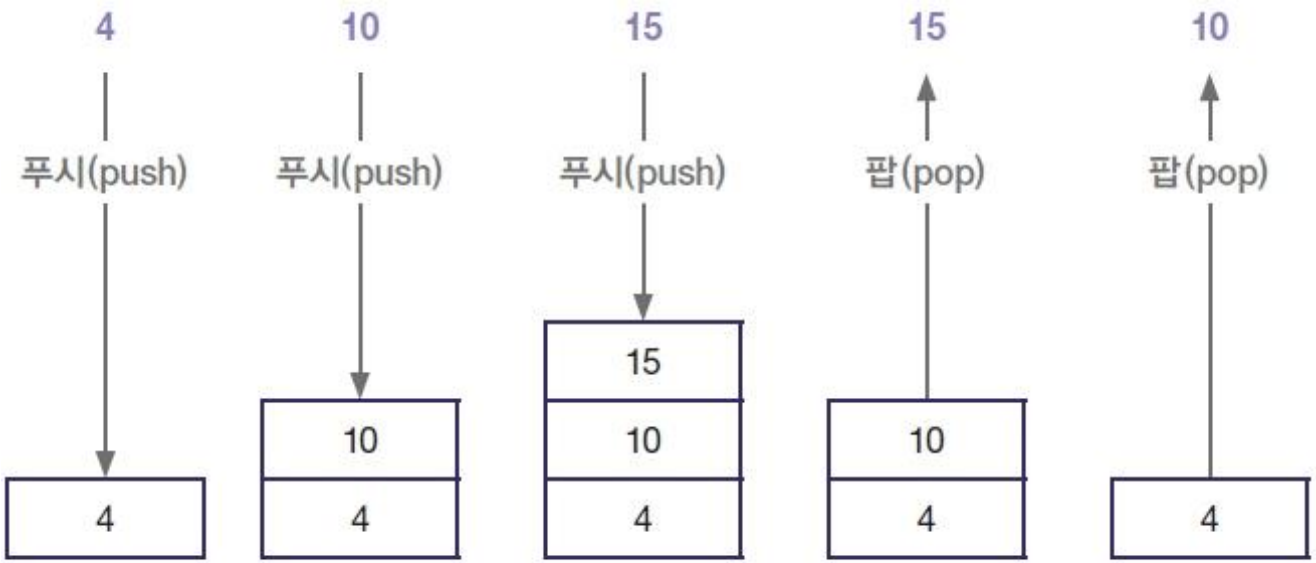


그림 7-2 스택

- 스택은 어떤 상황에서 사용할 수 있을까?

  - 택배 수화물을 저장하는 방식: 먼저 배달해야 하는 수화물은 트럭의 입구 쪽에, 나중에 배달해야 하는 수화물은 트럭 의 안쪽에 넣어야 함

1. 스택

- 파이썬에서는 리스트를 사용하여 스택을 구현

☞ 리스트라는 저장 공간을 만든 후 append( ) 함수로 데이터를 저장(push)하고 pop( ) 함수로 데이터를 추출(pop)함.

```
>>> a = [1, 2, 3, 4, 5]
>>> a.append(10)
>>> a
[1, 2, 3, 4, 5, 10]
>>> a.append(20)
>>> a
[1, 2, 3, 4, 5, 10, 20]
>>> a.pop()
20
>>> a.pop()
10
```

1. 스택

- 입력한 텍스트를 역순으로 추출하는 프로그램

[코드 7-1]

```
 1 word = input("Input a word: ")
 2 world_list = list(word)
 3 print(world_list)
 4
 5 result = [ ]
 6 for _ in range(len(world_list)):
 7     result.append(world_list.pop())
 8
 9 print(result)
10 print(word[::-1])
```

[실행결과]

```
Input a word: PYTHON                    ← 사용자 입력(PYTHON)
['P', 'Y', 'T', 'H', 'O', 'N']
['N', 'O', 'H', 'T', 'Y', 'P']
NOHTYP
```

2. 큐

- **큐(queue):** 스택과 다르게 먼저 들어간 데이터가 먼저 나오는 'Fist in First Out (FIFO)'의 메모리 구조를 가지는 자료구조

표 7-3 대학생 인적사항

학번	이름	생년월일	주소
20150230	홍길동	1995-04-03	서울시 동대문구
20150233	김영철	1995-04-20	성남시 분당구
20150234	오영심	1996-01-03	성남시 중원구
20150236	최성철	1995-12-27	인천시 계양구

2. 큐

- 파이썬에서 큐를 구현하는 법

```
>>> a = [1, 2, 3, 4, 5]
>>> a.append(10)                # a = [1, 2, 3, 4, 5, 10]
>>> a.append(20)                # a = [1, 2, 3, 4, 5, 10, 20]
>>> a.pop(0)
1
>>> a.pop(0)
2
```

- 기본적으로 스택의 구현과 같은데, pop( ) 함수를 사용할 때 인덱스가 0번째인 값을 쓴다는 의미로 pop(0)을 사용하면 됨.
- pop( ) 함수가 리스트의 마지막 값을 가져온다고 했을 때 pop(0)은 맨 처음 값을 가져온다는 뜻.

03
튜플과 세트

1. 튜플

- **튜플(tuple):** 리스트와 같은 개념이지 만 값을 변경하는 것이 불가능한 리스트로의 자료구조

```
>>> t = (1, 2, 3)
>>> print(t + t , t * 2)
(1, 2, 3, 1, 2, 3) (1, 2, 3, 1, 2, 3)
>>> len(t)
3
```

- 만약 튜플의 값을 변경하고자 한다면 다음과 같이 오류가 발생함.

```
>>> t[1] = 5
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

2. 세트

- **세트(set):** 값을 순서 없이 저장하되 중복을 불허하는 자료형

```
>>> s = set([1, 2, 3, 1, 2, 3]) # set() 함수를 사용하여 1, 2, 3을 세트 객체로 생성
>>> s
{1, 2, 3}
```

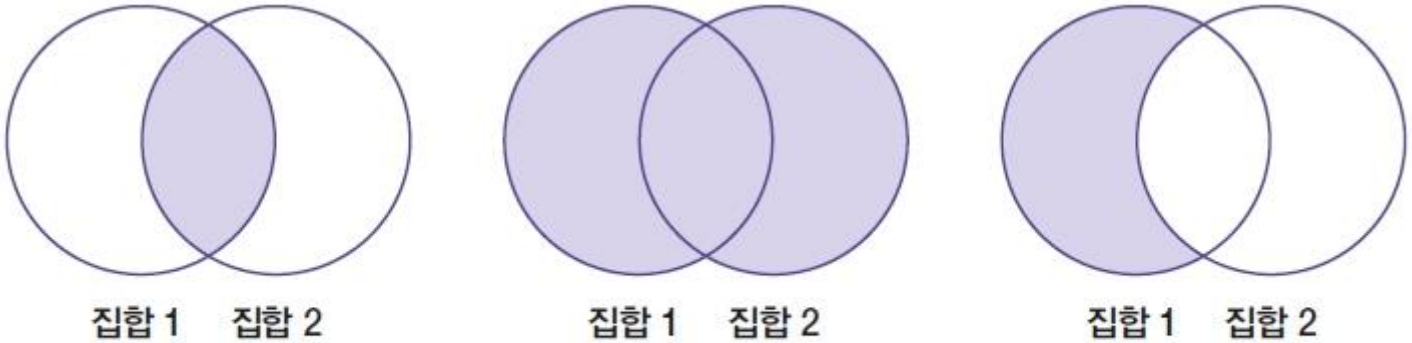
- 세트는 튜플과 다르게 삭제나 변경이 가능함.

```
>>> s
{1, 2, 3}
>>> s.add(1)              # 1을 추가하는 명령이지만 중복 불허로 추가되지 않음
>>> s
{1, 2, 3}
>>> s.remove(1)                          # 1 삭제
>>> s
{2, 3}
>>> s.update([1, 4, 5, 6, 7])           # [1, 4, 5, 6, 7] 추가
>>> s
{1, 2, 3, 4, 5, 6, 7}
>>> s.discard(3)                         # 3 삭제
>>> s
{1, 2, 4, 5, 6, 7}
>>> s.clear()                            # 모든 원소 삭제
>>> s
set()
```

2. 세트

- **세트를 지원하는 함수**

- 원소 하나를 추가하는 add( )
- 원소 하나를 제거하는 remove( ) 또는 discard( )
- 새로운 리스트를 그대로 추가하는 update( )
- 모든 변수를 지우는 clear( )



(a) 교집합 (b) 합집합 (c) 차집합

그림 7-4 집합 연산

2. 세트

- 파이썬은 교집합, 합집합, 차집합 연산을 모두 지원함.

```
>>> s1 = set([1, 2, 3, 4, 5])
>>> s2 = set([3, 4, 5, 6 ,7])
>>>
>>> s1.union(s2)                          # s1과 s2의 합집합
{1, 2, 3, 4, 5, 6, 7}
>>> s1 | s2                               # set([1, 2, 3, 4, 5, 6, 7])
{1, 2, 3, 4, 5, 6, 7}
>>> s1.intersection(s2)                   # s1과 s2의 교집합
{3, 4, 5}
>>> s1 & s2                               # set([3, 4, 5])
{3, 4, 5}
>>> s1.difference(s2)                     # s1과 s2의 차집합
{1, 2}
>>> s1 - s2                               # set([1, 2])
{1, 2}
```

2. 세트

표 7-2 세트의 집합 연산

연산	함수	기호	예시
합집합	union	|	s1.union(s2), s1 | s2
교집합	intersection	&	s1.intersection(s2), s1 & s2
차집합	difference	-	s1.difference(s2), s1 - s2

- **합집합:** 두 집합의 중복 값을 제거하고 합치는 연산
- **교집합:** 두 집합 양쪽에 모두 포함된 값만 추출하는 연산
- **차집합:** 앞에 있는 집합 s1의 원소 중 s2에 포함된 원소를 제거하는 연산

04
딕셔너리

1. 딕셔너리의 개념

딕셔너리(dictionary)

- 파이썬에서 가장 많이 사용하는 자료구조
- 영어사전에서 검색을 위해 영어 단어들을 저장해 놓은 방식과 비슷함.

  - 영어사전에서 각 단어를 검색할 수 있도록 색인(index)을 만들어 놓고 색인을 통해 그 단어를 찾아 의미를 파악함.

- 파이썬의 딕셔너리 구조에서는 데이터의 유일한 구분자인 키(key)라는 이름으로 검색할 수 있게 하고, 실제 데이터를 값(value)이라는 이름과 쌍으로 저장하여 프로그래머가 데이터를 쉽게 찾을 수 있도록 함.

1. 딕셔너리의 개념

- [표 7-3]의 대학생 인적사항에서 학번이 나머지 정보를 구분하는 키.
- 학번을 키로 하여 이름 ·생년월일 ·주소를 리스트 형태로 저장한 다음 한 번에 검색할 수 있는 형태가 되면 학번을 이용해 다른 정보에 쉽게 접근할 수 있음.

표 7-3 대학생 인적사항

학번	이름	생년월일	주소
20150230	홍길동	1995-04-03	서울시 동대문구
20150233	김영철	1995-04-20	성남시 분당구
20150234	오영심	1996-01-03	성남시 중원구
20150236	최성철	1995-12-27	인천시 계양구

2. 파이썬에서의 딕셔너리

- **딕셔너리의 선언:** 중괄호 { }를 사용하여 키와 값을 쌍으로 구성

```
딕셔너리 변수 = {키 1:값 1, 키 2:값 2, 키 3:값 3, ···}
```

예) [표 7-4]처럼 학번을 키로 사용하고, 이름을 값으로 지정할 수 있음.

표 7-4 키와 값의 샘플

학번(키)	이름(값)
20140012	Sungchul
20140059	Jiyong
20150234	Jaehong
20140058	Wonchul

2. 파이썬에서의 딕셔너리

- [표 7-4]의 정보를 간단히 파이썬으로 표현하기

```
>>> student_info = {20140012:'Sungchul', 20140059:'Jiyong', 20140058:'Jaehong'}
```

- 해당 변수에서 특정 값을 호출하는 방법

```
>>> student_info[20140012]
'Sungchul'
```

- 재할당과 데이터 추가

```
>>> student_info[20140012] = 'Sungchul'
>>> student_info[20140012]
'Sungchul'
>>> student_info[20140039] = 'Wonchul'
>>> student_info
{20140012:'Sungchul',20140059:'Jiyong',20140058:'Jaehong',20140039:'Won
chul'}
```

3. 딕셔너리 함수

- 파이썬에서는 딕셔너리를 쉽게 사용할 수 있도록 다양한 함수를 제공.

- 국가명과 국가 전화번호를 묶어 보여주는 코드 작성

```
>>> country_code = { }                                  # 딕셔너리 생성
>>> country_code = {"America": 1, "Korea": 82, "China": 86, "Japan": 81}
>>> country_code
{'America': 1, 'Korea': 82, 'China': 86, 'Japan': 81}
```

- 딕셔너리 변수 안의 키와 값을 출력하는 함수

```
>>> country_code.keys()                              # 딕셔너리의 키만 출력
dict_keys(['America', 'Korea', 'China', 'Japan'])
```

3. 딕셔너리 함수

- 값을 출력하기 위해서 values( ) 함수 사용

```
>>> country_code["German"] = 49                    # 딕셔너리 추가
>>> country_code
{'America': 1, 'Korea': 82, 'China': 86, 'Japan': 81, 'German': 49}
>>> country_code.values()                          # 딕셔너리의 값만 출력
dict_values([1, 82, 86, 81, 49])
```

- 키-값 쌍을 모두 보여주기 위해서 items( ) 함수 사용

```
>>> country_code.items()                          # 딕셔너리 데이터 출력
dict_items([('America',1),('Korea',82),('China',86),('Japan',81),
('German',49)])
```

3. 딕셔너리 함수

- for문과 함께 딕셔너리를 사용하여 키-값 쌍을 화면에 출력하기.

```
>>> for k, v in country_code.items():
...     print("Key:", k)
...     print("Value:", v)
...
Key: America
Value: 1
Key: Korea
Value: 82
Key: China
Value: 86
Key: Japan
Value: 81
Key: Gernman
Value: 49
```

3. 딕셔너리 함수

- if문을 사용하여 특정 키나 값이 해당 변수에 포함되어 있는지 확인하기

```
>>> "Korea" in country_code.keys()          # 키에 "Korea"가 있는지 확인
True
>>> 82 in country_code.values()             # 값에 82가 있는지 확인
True
```

05
collections 모듈

- **collections 모듈:** 리스트, 튜플, 딕셔너리 등을 확장하여 제작된 파이썬의 내장 모듈
- collections 모듈은 deque, OrderedDict, defaultdict, Counter, namedtuple 등을 제공함.
- 각 자료구조를 호출하는 코드

```
from collections import deque
from collections import OrderedDict
from collections import defaultdict
from collections import Counter
from collections import namedtuple
```

1. deque 모듈

- **deque 모듈:** 스택과 큐를 모두 지원하는 모듈
- **deque의 사용:** 리스트와 비슷한 형식으로 데이터를 저장해야 함

```
>>> from collections import deque
>>>
>>> deque_list = deque()
>>> for i in range(5):
... deque_list.append(i)
...
>>> print(deque_list)
deque([0, 1, 2, 3, 4])
```

1. deque 모듈

- deque_list.pop()을 수행시키면 오른쪽 요소부터 하나씩 추출됨

```
>>> deque_list.pop()
4
>>> deque_list.pop()
3
>>> deque_list.pop()
2
>>> deque_list
deque([0, 1])
```

1. deque 모듈

- deque에서 큐는 어떻게 사용할 수 있을까?

  - pop(0)을 입력하여 실행하면 deque에서는 작동하지 않음.

  - deque는 appendleft( ) 함수로 새로운 값을 왼쪽부터 입력하도록 하여 먼저 들어간 값부터 출력될 수 있도록 함.

```
>>> from collections import deque
>>>
>>> deque_list = deque()
>>> for i in range(5):
... deque_list.appendleft(i)
...
>>> print(deque_list)
deque([4, 3, 2, 1, 0])
```

1. deque 모듈

- deque 모듈의 장점:

✓ deque는 연결 리스트의 특성을 지원함.

- 연결 리스트(linked list): 데이터를 저장할 때 요소의 값을 한 쪽으로 연결한 후, 요소의 다음 값의 주소 값을 저장하여 데이터를 연결하는 기법

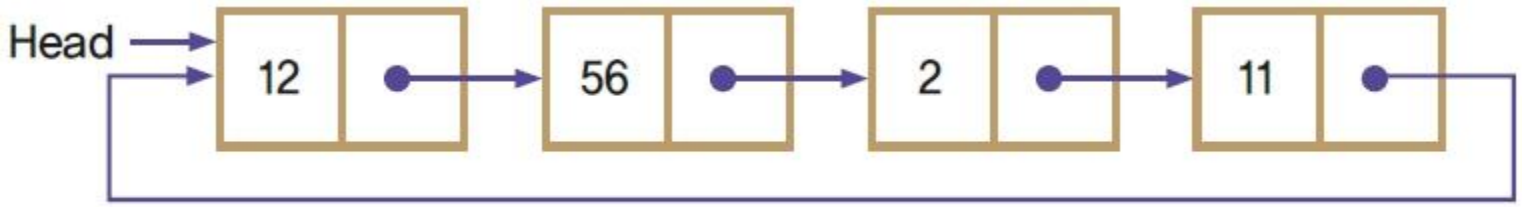


그림 7-5 연결 리스트의 형태

1. deque 모듈

✓ 연결 리스트는 다음 요소의 주소 값을 저장하므로 데이터를 원형으로 저장할 수 있음.

✓ 마지막 요소에 첫 번째 값의 주소를 저장시켜 해당 값을 찾아갈 수 있도록 연결시킴.

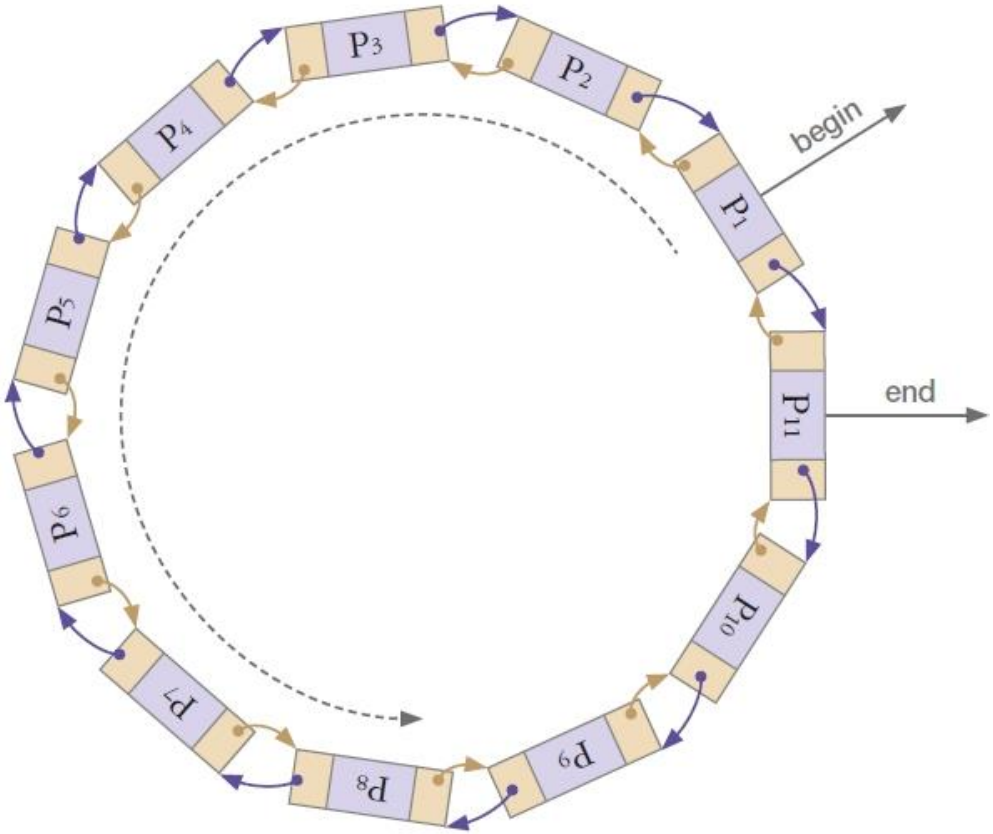


그림 7-6 원형 연결 리스트의 형태

1. deque 모듈

- **rotate( ):** 기존 deque에 저장된 요소들 값의 인덱스를 바꾸는 기법

```
>>> from collections import deque
>>>
>>> deque_list = deque()
>>> for i in range(5):
...     deque_list.appendleft(i)
...
>>> print(deque_list)
deque([0, 1, 2, 3, 4])
>>> deque_list.rotate(2)
>>> print(deque_list)
deque([3, 4, 0, 1, 2])
>>> deque_list.rotate(2)
>>> print(deque_list)
deque([1, 2, 3, 4, 0])
```

1. deque 모듈

- **reversed( ) 함수:** 기존과 반대로 데이터를 저장하는 기법.

```
>>> print(deque(reversed(deque_list)))
deque([0, 4, 3, 2, 1])
```

- **extend( ) 함수:** 리스트가 통째로 오른쪽으로 추가되는 기법
- **extendleft( ) 함수:** 리스트가 통째로 왼쪽으로 추가되는 기법

```
>>> deque_list.extend([5, 6, 7])
>>> print(deque_list)
deque([1, 2, 3, 4, 0, 5, 6, 7])
>>> deque_list.extendleft([5, 6, 7])
>>> print(deque_list)
deque([7, 6, 5, 1, 2, 3, 4, 0, 5, 6, 7])
```

2. OrderedDict 모듈

- **OrderedDict 모듈**: 순서를 가진 딕셔너리 객체

[코드 7-2]

```
1 d = { }
2 d['x'] = 100
3 d['l'] = 500
4 d['y'] = 200
5 d['z'] = 300
6
7 for k, v in d.items():
8     print(k, v)
```

[실행결과]

```
x 100
l 500
y 200
z 300
```

2. OrderedDict 모듈

- **딕셔너리:** 순서를 보장하지 않는 객체로, 딕셔너리 파일을 저장하면 키는 저장 순서와 상관없이 저장됨.

[코드 7-3]

```
1 d = { }
2 d['x'] = 100
3 d['l'] = 500
4 d['y'] = 200
5 d['z'] = 300
6
7 for k, v in d.items():
8     print(k, v)
```

[실행결과]

```
x 100
l 500
y 200
z 300
```

2. OrderedDict 모듈

- 딕셔너리로 데이터 처리 시 키나 값으로 데이터를 정렬할 때 OrderedDict 모듈을 가장 많이 사용함.
- 키를 이용하여 주민등록번호를 번호 순서대로 정렬한 후 데이터를 출력하는 경우의 코드

[코드 7-4]

```
 1 def sort_by_key(t):
 2     return t[0]
 3
 4 from collections import OrderedDict          # OrderedDict 모듈 선언
 5
 6 d = dict()
 7 d['x'] = 100
 8 d['y'] = 200
 9 d['z'] = 300
10 d['l'] = 500
11
12 for k, v in OrderedDict(sorted(d.items(), key=sort_by_key)).items():
13     print(k, v)
```

2. OrderedDict 모듈

[실행결과]

```
l 500
x 100
y 200
z 300
```

- 딕셔너리 값인 변수 d를 리스트 형태로 만든 다음, sorted( ) 함수를 사용하여 정렬.

- sorted(d.items(), key=sort_by_key)의 코드만 따로 실행하면 다음처럼 정렬되어 이차원 형태로 값이 출력됨.

```
[('l', 500), ('x', 100), ('y', 200), ('z', 300)]
```

- 만약, 값을 기준으로 정렬한다면 [코드 7-4]의 1행과 2행을 다음처럼 바꾸면 됨.

```
def sort_by_value(t):
    return t[1]
```

3. defaultdict 모듈

- 딕셔너리의 변수를 생성할 때 키에 기본값을 지정하는 방법
  - 새로운 키를 생성할 때 별다른 조치 없이 새로운 값을 생성할 수 있음.
  - 실제 딕셔너리에서는 [코드 7-5]처럼 키를 생성하지 않고 해당 키의 값을 호출하려고 할 때 오류가 발생

[코드 7-5]

```
1 d = dict()
2 print(d["first"])
```

[실행결과]

```
Traceback (most recent call last):
    File "defaultdict1.py", line 2, in <module>
        print(d["first"])
KeyError: 'first'
```

3. defaultdict 모듈

- defaultdict 모듈은 어떻게 사용할까?

[코드 7-6]

```
1 from collections import defaultdict
2
3 d = defaultdict(lambda: 0) # Default 값을 0으로 설정
4 print(d["first"])
```

[실행결과]

```
0
```

3. defaultdict 모듈

- defaultdict의 초기 값은 리스트 형태로도 설정할 수 있음

[코드 7-7]

```
1 from collections import defaultdict
2
3 s = [('yellow',1), ('blue',2), ('yellow',3), ('blue',4), ('red',1)]
4 d = defaultdict(list)
5 for k, v in s:
6     d[k].append(v)
7
8 print(d.items())
9 [('blue', [2, 4]), ('red', [1]), ('yellow', [1, 3])]
```

[실행결과]

```
dict_items([('yellow', [1, 3]), ('blue', [2, 4]), ('red', [1])])
```

4. Counter 모듈

- 시퀀스 자료형의 데이터 값의 개수를 딕셔너리 형태로 반환하는 방법

```
>>> from collections import Counter
>>>
>>> text = list("gallahad")
>>> text
['g', 'a', 'l', 'l', 'a', 'h', 'a', 'd']
>>> c = Counter(text)
>>> c
Counter({'a': 3, 'l': 2, 'g': 1, 'h': 1, 'd': 1})
>>> c["a"]
3
```

4. Counter 모듈

- Counter를 이용하면 각 문자의 개수 세는 작업을 매우 쉽게 할 수 있음

```
>>> text = """A press release is the quickest and easiest way to get
free publicity. If well written, a press release can result in
multiple published articles about your firm and its products. And
that can mean new prospects contacting you asking you to sell to
them. ....""".lower().split()
>>> Counter(text)
Counter({'and': 3, 'to': 3, 'can': 2, 'press': 2, 'release': 2,
'you': 2, 'a': 2, 'sell': 1, 'about': 1, 'free': 1, 'firm': 1,
'quickest': 1, 'products.': 1, 'written,': 1, 'them.': 1, '....': 1,
'articles': 1, 'published': 1, 'mean': 1, 'that': 1, 'prospects': 1,
'its': 1, 'multiple': 1, 'if': 1, 'easiest': 1, 'publicity.': 1,
'way': 1, 'new': 1, 'result': 1, 'the': 1, 'your': 1, 'well': 1,
'is': 1, 'asking': 1, 'in': 1, 'contacting': 1, 'get': 1})
```

4. Counter 모듈

- 딕셔너리 형태로 Counter 객체를 생성하는 방법

```
>>> from collections import Counter
>>>
>>> text = list("gallahad")
>>> text
['g', 'a', 'l', 'l', 'a', 'h', 'a', 'd']
>>> c = Counter(text)
>>> c
Counter({'a': 3, 'l': 2, 'g': 1, 'h': 1, 'd': 1})
>>> c["a"]
3
```

4. Counter 모듈

- 키워드 형태의 매개변수를 사용하여 Counter를 생성하는 방법

```
>>> from collections import Counter
>>>
>>> c = Counter(cats = 4, dogs = 8)
>>> print(c)
Counter({'dogs': 8, 'cats': 4})
>>> print(list(c.elements()))
['cats','cats','cats','cats','dogs','dogs','dogs','dogs','dogs','dogs
','dogs','dogs']
```

- Counter는 기본 사칙연산(덧셈, 뺄셈, 논리 연산 등)을 지원

```
>>> from collections import Counter
>>>
>>> c = Counter(a = 4, b = 2, c = 0, d = -2)
>>> d = Counter(a = 1, b = 2, c = 3, d = 4)
>>> c.subtract(d)                              # c - d
>>> c
Counter({'a': 3, 'b': 0, 'c': -3, 'd': -6})
```

4. Counter 모듈

```
>>> from collections import Counter
>>>
>>> c = Counter(a = 4, b = 2, c = 0, d = -2)
>>> d = Counter(a = 1, b = 2, c = 3, d = 4)
>>> print(c + d)
Counter({'a': 5, 'b': 4, 'c': 3, 'd': 2})
>>> print(c & d)
Counter({'b': 2, 'a': 1})
>>> print(c | d)
Counter({'a': 4, 'd': 4, 'c': 3, 'b': 2})
```

- + 기호: 두 Counter 객체에 있는 각 요소를 더한 것,
- & 기호: 두 객체에 같은 값이 있을 때, 즉 교집합의 경우에만 출력함.
- | 기호: 두 Counter 객체에서 하나가 포함되어 있다면, 그리고 좀 더 큰 값이 있다면 그 값으로 합집합을 적용함.

5. namedtuple 모듈

- 튜플의 형태로 데이터 구조체를 저장하는 방법

  - 특정 데이터의 규정된 정보를 하나의 튜플 형태로 구성해 손쉽게 사용할 수 있는 자료구조

```
>>> from collections import namedtuple
>>>
>>> Point = namedtuple('Point', ['x', 'y'])
>>> p = Point(11, y=22)
>>> p
Point(x=11, y=22)
>>> p.x, p.y
(11, 22)
>>> print(p[0] + p[1])
33
```

06
Lab: 텍스트 마이닝 프로그램

1. 실습 내용

- **텍스트 마이닝(text mining):** 텍스트를 분석하여 의미 있는 결과 도출하는 과정

- 이번 Lab에서는 다음 문장에 있는 단어의 개수를 파악해본다.

```
A press release is the quickest and easiest way to get free publicity.
If well written, a press release can result in multiple published
articles about your firm and its products. And that can mean new
prospects contacting you asking you to sell to them. ...
```

- 프로그램 작성하는 규칙

```
• 문장의 단어 개수를 파악하는 코드를 작성한다.
• defaultdict 모듈을 사용한다.
• 단어의 출현 횟수를 기준으로 정렬된 결과를 보여주기 위해 OrderedDict
모듈을 사용한다.
```

2. 실행 결과

[실행결과]

```
and 3
to 3
a 2
press 2
release 2
⋮
(생략)
⋮
contacting 1
asking 1
sell 1
them. 1
.... 1
```

3. 문제 해결

[코드 7-8]

```
 1 text = """A press release is the quickest and easiest way to get free
   publicity. If well written, a press release can result in multiple
   published articles about your firm and its products. And that can mean new
   prospects contacting you asking you to sell to them. ….""".lower().split()
 2
 3 from collections import defaultdict
 4
 5 word_count = defaultdict(lambda: 0) # Default 값을 0으로 설정
 6 for word in text:
 7     word_count[word] += 1
 8
 9 from collections import OrderedDict
10 for i, v in OrderedDict(sorted(word_count.items(), key=lambda t: t[1],
   reverse=True)).items():
11 print(i, v)
```

LAB 텍스트 마이닝 프로그램

- 1행: text 변수에 원하는 문장을 넣고, 이를 소문자로 바꾼 후 단어 단위로 자르는 코드

  - 이를 위해 lower( )와 split( ) 함수를 연속으로 사용함

  - 이 코드의 결과를 확인하기 위해 파이썬 셸에 다음과 같이 입력하면 리스트의 결과를 볼 수 있음.

```
>>> text = """A press release is the quickest and easiest way to get
free publicity. If well written, a press release can result in multiple
published articles about your firm and its products. And that can mean
new prospects contacting you asking you to sell to them.
....""".lower().split()
>>> print(text)
['a', 'press', 'release', 'is', 'the', 'quickest', 'and', 'easiest',
'way', 'to', 'get', 'free', 'publicity.', 'if', 'well', 'written,', 'a',
'press', 'release', 'can', 'result', 'in', 'multiple', 'published',
'articles', 'about', 'your', 'firm', 'and', 'its', 'products.', 'and',
'that', 'can', 'mean', 'new', 'prospects', 'contacting', 'you',
'asking', 'you', 'to', 'sell', 'to', 'them.', '....']
```

Thank you!

한빛아카데미 Hanbit Academy, Inc.