

데이터과학을 위한

파이썬
프로그래밍



07. 자료구조

목차

1. 자료구조의 이해
2. 스택과 큐
3. 튜플과 세트
4. 딕셔너리
5. collections 모듈
6. Lab: 텍스트 마이닝 프로그램

01

자료구조의 이해

01. 자료구조의 이해

■ 자료구조의 개념

- **자료구조(data structure)** : 특징이 있는 정보를 메모리에 효율적으로 저장 및 반환하는 방법으로, 데이터를 관리하는 방식이다. 특히 대용량일수록 메모리에 빨리 저장하고 빠르게 검색하여, 메모리를 효율적으로 사용하고 실행 시간을 줄일 수 있게 해 준다.

20 YELLOW PAGE 디라이프 전화번호부

자주 찾는 한국명사전 북경대사관공사 010-6532-6774 성도출판사 028-8616-5800 흥륜출판사 00852-2529-4141 상해출판사 021-6295-5000 천도출판사 0532-8897-0001 신광출판사 024-2385-3388 광주출판사 020-3887-0555 사천출판사 029-8835-1001	가 KOTRA 중국무역관 6029-1005 중소기업진흥공단(총청) 023-6705-2399 중앙일보(사) 6382-0753 중앙일보(상) 185-8006-9033	진급 상황 전 출판/사상사 110 전파고교신 112 국제경제연구소 113 국제경제연구소 115 국제경제연구소 116 국제경제연구소 117 국제경제연구소 119 국제경제연구소 120 국제경제연구소 121 국제경제연구소 122 우원출판사 184	기업/컨설팅 커먼마켓 185-1040-3282 디앤디 185-1162-3396 비즈마켓 6734-3488 아시아-컨설팅 6796-2600 / 2033 아틀라스 컨설팅 177-8319-6880	부 상해출판사 133-2195-2345 중광출판사 187-2336-0523 중광114출판사 186-0218-9115 한신출판사 136-2165-5550 한신출판사 183-5801-1818	미용 / 패션 / 뷰티 미용 / 패션 / 뷰티 185-8019-4590
---	---	--	--	--	---

(a) 전화번호부

9:58

그룹 연락처

Q 검색

L 나성업 C 도민준 리 라인업 M 마성지	강남아저씨 186-9050-0675 고함집 (송재) 137-7318-1808 내고향 공방집 023-6721-0099 내고향 한식방집 023-6710-7519 내고향 치보집 136-1822-4115 내고향 통통집 023-6739-4999 맥분가 183-7564-8006 무궁화 (송재) 139-8338-2107 미미양육아당 136-2834-0525 전통양식 백화 023-6305-9288 복익산 136-4059-9991 서울권 186-9886-3758 스카이 피자국밥 185-8465-1153 애정양꼬치 186-1111-3050 일가 소파라/떡방 135-0127-0665 청춘 156-9210-5002 취향 중독요리 131-4021-0035 한복가 (트루) 157-3046-9099 한국강남스타일 159-9891-3032 한성관 159-2277-5826 한우 채우집 023-6703-4432
--	---

(b) 휴대전화의 연락처

01. 자료구조의 이해

■ 파이썬에서의 자료구조

자료구조명	특징
스택(stack)	나중에 들어온 값을 먼저 나갈 수 있도록 해 주는 자료구조(last in first out)
큐(queue)	먼저 들어온 값을 먼저 나갈 수 있도록 해 주는 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) 형태의 데이터를 저장하는 자료구조, 여기서 키값은 다른 데이터와 중복을 허용하지 않음
collections 모듈	위에 열거된 여러 자료구조를 효율적으로 사용할 수 있도록 지원하는 파이썬 내장(built-in) 모듈

[파이썬에서 제공하는 자료구조]

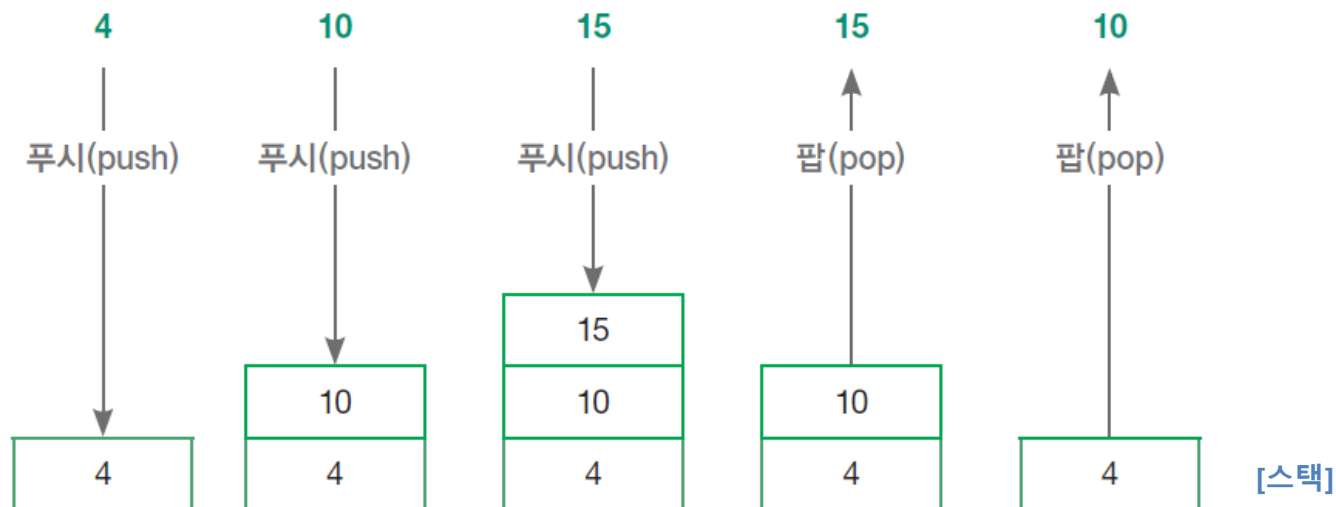
02

스택과 큐

02. 스택과 큐

■ 스택

- **스택(stack)** : 자료구조의 핵심 개념 중 하나로, 간단히 표현하면 'Last In First Out(LIFO)'으로 정의할 수 있다. 즉, 마지막에 들어간 데이터가 가장 먼저 나오는 형태로, 데이터의 저장 공간을 구현하는 것이다.
- 아래 그림에서 4, 10과 같은 데이터를 저장하는 공간으로, 리스트와 비슷하지만 저장 순서가 바뀌는 형태를 스택 자료구조(stack data structure)라고 한다. 스택에서 데이터를 저장하는 것을 푸시(push), 데이터를 추출하는 것을 팝(pop)이라고 한다.



02. 스택과 큐

■ 스택

- 파이썬에서는 리스트를 사용하여 스택을 구현할 수 있다. 리스트라는 저장 공간을 만든 후, `append()` 함수로 데이터를 저장(push)하고 추출(pop)한다. 다음 코드를 확인해 보자.

```
>>> 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
```

- 먼저 변수 `a`에는 `[1, 2, 3, 4, 5]`가 할당된다.
- 변수 `a`에 10과 20을 추가하면, 변수 `a`에는 `[1, 2, 3, 4, 5, 10, 20]`이 할당된다.
- `pop()` 함수를 처음 실행하면, 가장 마지막에 저장된 20이 추출되면서 화면에 출력되고, 동시에 변수 `a`의 값은 `[1, 2, 3, 4, 5, 10]`으로 변한다.
- 다시 `pop()` 함수를 실행하면, 마지막에 저장된 10이 추출되면서 화면에 출력되고, 동시에 변수 `a`의 값은 `[1, 2, 3, 4, 5]`로 변한다.

02. 스택과 큐

■ 스택

- 스택으로 만들 수 있는 프로그램 중 하나는 입력한 텍스트의 역순을 추출하는 프로그램을 작성하는 것이다.

코드 7-1 stack.py

```
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])
```

```
Input a word: PYTHON
['P', 'Y', 'T', 'H', 'O', 'N']
['N', 'O', 'H', 'T', 'Y', 'P']
NOHTYP
```

← 사용자 입력(PYTHON)

02. 스택과 큐

■ 스택 : [코드 7-1] 해석

- 먼저 입력한 텍스트는 변수 `word`에 저장되고, 그 값을 리스트형으로 변환한다. 그 후 값을 차례대로 추출하면, 입력한 텍스트의 역순값이 출력된다.
- [코드 7-1]에서 확인할 코드가 있다. 바로 `_` 기호이다. 일반적으로 `for`문에서 많이 쓰이는데, `for`문에 `_` 기호가 있으면 해당 반복문에서 생성되는 값은 코드에서 사용하지 않는다는 뜻이다. [코드 7-1]에서는 6행의 `range(len(world_list))`에서 생성되는 값이 반복문 내에서 사용되지 않으므로 `_`로 할당받은 것이다.

02. 스택과 큐

■ 큐

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

PUT(A)	PUT(B)	PUT(C)	GET()	PUT(D)	GET()
				D	D
		C	C	C	C
	B	B	B	B	
A	A	A			

[큐]

02. 스택과 큐

■ 큐

- 큐는 어떤 상황에서 사용할 수 있을까? 대표적인 예로, 앞서 언급한 사례 중 은행에서 대기 번호표를 뽑을 때 번호를 저장하는 방식이다. 먼저 온 사람이 앞의 번호표를 뽑고, 번호가 빠른 사람이 먼저 서비스를 받는 구조이다.
- 파이썬에서 큐를 구현하는 것은 기본적으로 스택의 구현과 같은데, `pop()` 함수를 사용할 때 인덱스가 0번째인 값을 쓴다는 의미로 `pop(0)`을 사용하면 된다. 즉, `pop()` 함수가 리스트의 마지막 값을 가져온다고 하면, `pop(0)`은 맨 처음 값을 가져온다는 뜻이다.

```
>>> 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
```

03

튜플과 세트

03. 튜플과 세트

■ 튜플

- 튜플(tuple)은 리스트와 같은 개념이지만, 데이터를 변경할 수 없는 자료구조이다

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

- ➔ 첫 번째 줄에서 튜플을 선언하는데, 튜플은 괄호를 이용하여 $t=(1, 2, 3)$ 과 같은 형태로 선언한다. 대괄호 $[]$ 를 이용하는 리스트와는 차이가 있다. 하지만 선언 외에 여러 가지 연산은 리스트와 같아, 리스트에서 사용하는 연산, 인덱싱, 슬라이싱이 모두 동일하게 적용된다. 위의 코드처럼 튜플 간의 덧셈 $t + t$ 이나 곱셈 $t * 2$, 그리고 $len()$ 과 같은 리스트형 데이터에 사용하는 함수 모두 사용할 수 있다.

03. 튜플과 세트

■ 튜플

- 튜플과 리스트의 유일하면서도 큰 차이점이 있다면, 튜플의 값은 마음대로 변경할 수 없다는 것이다. 만약 튜플의 값을 변경하고 싶다면 다음과 같이 오류가 발생한다.

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

03. 튜플과 세트

■ 튜플

- 튜플은 언제 사용할까?

사실 프로그래밍을 하다 보면 자신이 하나의 함수만 만들고, 다른 사람이 그 함수의 결과값을 사용해야 하는 경우가 발생할 수 있다. 이때 반환해 주는 타입을 튜플로 선언하여 받아서 사용하는 사람이 마음대로 데이터를 바꾸지 못하게 할 수 있다.

- 그렇다면 잘 바뀌지 않는 데이터는 어떤 것이 있을까?

학번이나 이름, 주민등록번호와 같이 변경되지 않아야 하는 정보 등이다. 프로그래머가 이러한 이해 없이 마음대로 값을 변경하려고 할 때, 튜플은 이를 방지하는 기능을 한다.

03. 튜플과 세트

■ 세트

- **세트(set)** : 값을 순서 없이 저장하면서 중복을 불허하는 자료형이다. 세트는 튜플과 다르게 삭제나 변경이 가능하며, 다양한 집합 연산을 제공한다.

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

- ➡ 세트를 사용하기 위해 `set()` 함수를 사용하여 리스트나 튜플의 데이터를 넣으면, 해당 값이 세트 형태로 변환된다. 위 코드처럼 `[1, 2, 3, 1, 2, 3]`이라는 리스트형의 값을 세트로 변환하면, 중복을 제거한 후 `{1, 2, 3}`으로 변환되어 출력된다.

03. 튜플과 세트

■ 세트

- 세트는 튜플과 다르게 삭제나 변경이 가능하다. 이 기능을 위해 다양한 함수를 다음과 같이 지원한다.

```
>>> 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()
```

03. 튜플과 세트

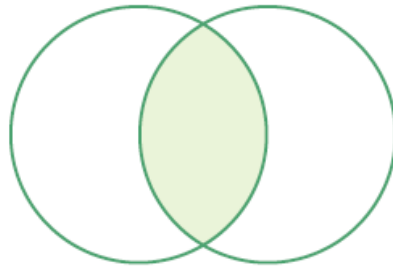
■ 세트

- **add()** : 원소 하나를 추가
- **remove()** 또는 **discard()** : 원소 하나를 제거
- **update()** : 새로운 리스트를 그대로 추가
- **clear()** : 모든 변수를 지우기

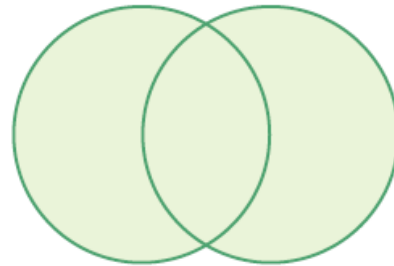
03. 튜플과 세트

■ 세트

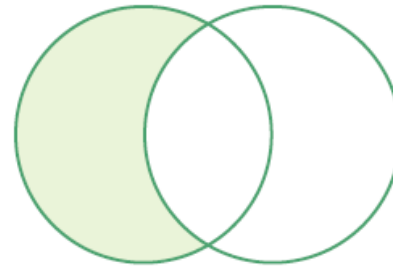
- 파이썬의 세트는 수학의 집합과 마찬가지로 다양한 집합 연산을 제공한다.



집합 1 집합 2
(a) 교집합



집합 1 집합 2
(b) 합집합



집합 1 집합 2
(c) 차집합

[집합 연산]

03. 튜플과 세트

■ 세트

[illegible]

03. 튜플과 세트

■ 세트

- 합집합은 두 집합의 중복값을 제거하고 합치는 연산이다. 위 코드에서는 `s1.union(s2)` 를 통해 `s1`과 `s2`의 합집합이 출력되었다. 합집합은 `union`과 같은 함수로도 표현할 수 있지만, `|` 기호로도 추출할 수 있다. 위 코드에서 `s1 | s2`의 결과가 `s1.union(s2)`와 동일한 것을 확인할 수 있다.
- 교집합은 두 집합 양쪽에 모두 포함된 값만 추출하는 연산이다. 위 코드에서 `s1`과 `s2`는 모두 3, 4, 5를 원소로 가지고 있다. 이 경우, `s1.intersection(s2)`나 `s1 & s2`로 교집합을 추출 할 수 있다.
- 차집합은 앞에 있는 집합 `s1`의 원소 중 `s2`에 포함된 원소를 제거하는 연산이다. 즉, `s1`에서 `s1`과 `s2`의 교집합 원소를 삭제하면 된다. 앞선 코드에서 `s1`은 [1, 2, 3, 4, 5]를 가지고 있으므로 [3, 4, 5]를 제거하면 [1, 2]만 남는다. 코드로 표현하면 `s1.difference(s2)` 또는 `s1 - s2`로 차집합을 추출할 수 있다.

03. 튜플과 세트

■ 세트

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

[세트의 집합 연산]

04

딕셔너리

04. 딕셔너리

■ 딕셔너리의 개념

- 딕셔너리(dictionary) : 전화번호부와 같이 키(key)와 값(value) 형태로 데이터를 저장하는 자료구조이다.

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

[대학생 인적사항]

04. 딕셔너리

■ 파이썬에서의 딕셔너리

- 파이썬에서 딕셔너리의 선언은 중괄호 {}를 사용하여 키와 값의 쌍으로 구성하면 된다.

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

04. 딕셔너리

■ 파이썬에서의 딕셔너리

학번(키)	이름(값)
20140012	Janhyeok
20140059	Jiyong
20150234	JaeHong
20140058	Wonchul

[키와 값의 샘플]

- 표의 정보를 간단히 파이썬으로 표현해보자. student_info라는 변수를 먼저 선언한 후, 해당 변수에 {키:값} 형태로 값을 입력한다. 그럼 해당 변수는 간단히 저장된다.

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

04. 딕셔너리

■ 파이썬에서의 딕셔너리

- 해당 변수에서 특정 값을 호출하는 방법이다. 해당 값의 키를 대괄호 [] 안에 넣어 호출할 수 있다. 변수의 자료형을 정확히 모르고 호출한다면, 리스트로 오해할 수도 있다.

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

- 재할당과 데이터 추가이다

```
>>> student_info[20140012] = 'Janhyeok'
>>> student_info[20140012]
'Janhyeok'
>>> student_info[20140039] = 'Wonchul'
>>> student_info
{20140012: 'Janhyeok', 20140059: 'Jiyong', 20140058: 'JaeHong', 20140039: 'Wonchul'}
```

04. 딕셔너리

■ 딕셔너리의 함수

- 파이썬에서는 딕셔너리를 쉽게 사용할 수 있도록 다양한 함수를 제공한다.
- 국가명과 국가 전화번호를 묶어 보여 주는 코드를 작성하면 다음과 같다.

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

04. 딕셔너리

■ 딕셔너리의 함수

- 딕셔너리 변수 안의 키와 값을 출력하는 함수에 대해 알아보자. 먼저 키만 출력하기 위해서는 `keys()` 함수를 사용한다. 이 함수를 사용하면, 키가 리스트 형태로 출력된다

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

- 값을 출력하기 위해서는 `values()` 함수를 사용한다.

```
>>> 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])
```

04. 딕셔너리

■ 딕셔너리의 함수

- 키-값 쌍을 모두 보여 주기 위해서는 items() 함수를 사용한다.

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

04. 딕셔너리

■ 딕셔너리의 함수

- 실제로 딕셔너리를 사용할 때는 for문과 함께 사용한다. 다음 코드와 같이 키-값 쌍을 화면에 출력할 수 있다.

```
>>> 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
```


04. 딕셔너리

■ 딕셔너리의 함수

- 딕셔너리를 많이 사용하는 방법 중 하나는 if문을 사용하여 특정 키나 값이 해당 변수에 포함되어 있는지 확인하는 것이다.

```
>>> "Korea" in country_code.keys()
```

```
True
```

```
# 키에 "Korea"가 있는지 확인
```

```
>>> 82 in country_code.values()
```

```
True
```

```
# 값에 82가 있는지 확인
```

05

collections 모듈

05. collections 모듈

- collections 모듈은 이미 앞에서 배운 다양한 자료구조인 리스트, 튜플, 딕셔너리 등을 확장하여 제작된 파이썬의 내장 모듈이다.
- collections 모듈은 deque, OrderedDict, defaultdict, Counter, namedtuple 등을 제공하며, 각 자료구조를 호출하는 코드는 다음과 같다.

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

05. collections 모듈

■ deque 모듈

- deque 모듈은 스택과 큐를 모두 지원하는 모듈이다.
- Deque 모듈을 사용하기 위해서는 리스트와 비슷한 형식으로 데이터를 저장해야 한다. 먼저 `append()` 함수를 사용하면 기존 리스트처럼 데이터가 인덱스 번호를 늘리면서 쌓이기 시작한다. 다음 코드를 확인하자.

```
>>> 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])
```

05. collections 모듈

■ deque 모듈

- 여기서 다음 코드와 같이 `deque_list.pop()`을 작성하면, 오른쪽 요소부터 하나씩 추출된다. 즉, 스택처럼 나중에 넣은 값부터 하나씩 추출할 수 있다

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

05. collections 모듈

■ deque 모듈

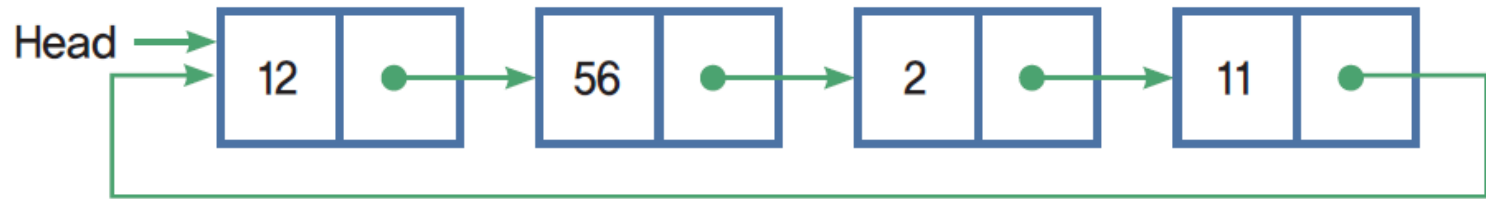
- 그렇다면 deque에서 큐는 어떻게 사용할 수 있을까? pop(0)을 입력하면 실행될 것 같지만, 이 함수는 deque에서 작동하지 않는다. 대신 deque는 appendleft() 함수로 새로운 값을 왼쪽부터 입력되게 하여 먼저 들어간 값부터 출력될 수 있도록 할 수 있다.

```
>>> 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])
```

05. collections 모듈

■ deque 모듈

- **deque 모듈의 장점** : deque는 연결 리스트의 특성을 지원한다. 연결 리스트는 데이터를 저장할 때 요소의 값을 한 쪽으로 연결한 후, 요소의 다음 값의 주소값을 저장하여 데이터를 연결하는 기법이다.

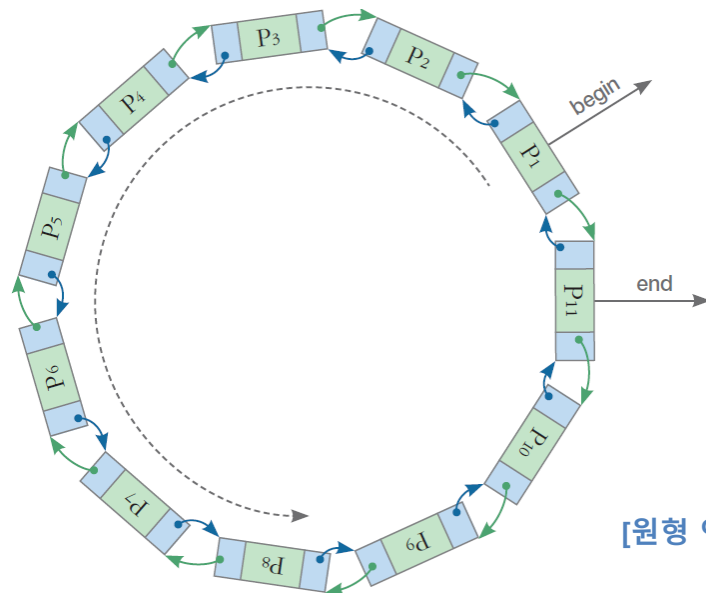


[연결 리스트의 형태]

05. collections 모듈

■ deque 모듈

- 연결 리스트는 다음 요소의 주소값을 저장하므로 데이터를 원형으로 저장할 수 있다. 또한, 마지막 요소에 첫 번째 값의 주소를 저장한다면 해당 값을 찾아갈 수 있다. 이러한 특징 때문에 가능한 기능 중 하나가 rotate() 함수이다. rotate()는 기존 deque에 저장된 요소들의 값 인덱스를 바꾸는 기법이다. 연결 리스트는 양쪽 끝의 요소들을 연결할 수 있으므로 원형의 데이터 구조를 가질 수 있다. 이러한 특징을 이용하여 각 요소의 인덱스 번호를 하나씩 옮긴다면, 실제로 요소를 옮기지 않더라도 인덱스 번호를 바꿀 수 있다.



[원형 연결 리스트의 형태]

05. collections 모듈

■ deque 모듈

- 다음 코드를 살펴보면, 기존 데이터에 rotate(2) 함수를 입력하니 3과 4의 값이 두 칸씩 이동하여 0번째, 1번째 인덱스로 옮겨진 것을 확인할 수 있다. 다시 rotate(2)를 사용하면, 1과 2가 0번째, 1번째 인덱스로 이동한다.

```
>>> 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])
```

05. collections 모듈

■ deque 모듈

- deque 모듈은 `reversed()` 함수를 사용하여 기존과 반대로 데이터를 저장할 수 있다.

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

- deque 모듈은 기존의 리스트에서 지원하는 함수도 지원한다. `extend()` 나 `extendleft()` 함수를 사용하면, 리스트가 통째로 오른쪽이나 왼쪽으로 추가된다

```
>>> 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])
```

05. collections 모듈

OrderedDict 모듈

- OrderedDict 모듈은 이름 그대로 순서를 가진 딕셔너리 객체이다. 딕셔너리 파일을 저장하면 키는 저장 순서와 상관없이 저장된다.

코드 7-2 ordereddict1.py

```
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)
```

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

05. collections 모듈

OrderedDict 모듈

코드 7-3 ordereddict2.py

```
1 from collections import OrderedDict    # OrderedDict 모듈 선언
2
3 d = OrderedDict()
4 d['x'] = 100
5 d['y'] = 200
6 d['z'] = 300
7 d['l'] = 500
8
9 for k, v in d.items():
10     print(k, v)
```

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

05. collections 모듈

OrderedDict 모듈

코드 7-4 ordereddict3.py

```
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)
```

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

05. collections 모듈

■ OrderedDict 모듈 : [코드 7-4] 해석

- [코드 7-4]를 보면 딕셔너리의 값인 변수 d를 리스트 형태로 만든 다음, sorted() 함수를 사용하여 정렬한다. sorted(d.items(), key=sort_by_key)의 코드만 따로 실행하면 다음처럼 정렬되어 이차원 형태로 출력되는 값을 확인할 수 있다.

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

- 값을 기준으로 정렬한다면 [코드 7-4]의 1행과 2행을 다음처럼 바꾸면 된다. 참고로 t[0]과 t[1]은 위 리스트 안의 튜플 값 중 0번째 인덱스(l, x, y, z)와 1번째 인덱스(500, 100, 200, 300)를 뜻한다.

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

05. collections 모듈

■ defaultdict 모듈

- defaultdict 모듈은 딕셔너리의 변수를 생성할 때 키에 기본 값을 지정하는 방법이다.
- ➔ 실제 딕셔너리에서는 [코드7 -5]처럼 키를 생성하지 않고 해당 키의 값을 호출하려고 할 때, 오류가 발생한다. 즉, 코드에서 first의 키값을 별도로 생성하지 않은 채 바로 호출하여 오류가 발생하였다.

코드 7-5 defaultdict1.py

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

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

05. collections 모듈

■ defaultdict 모듈

- 그렇다면 defaultdict 모듈은 어떻게 작동할까?

코드 7-6 defaultdict2.py

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

0

- ➔ 핵심은 3행의 `d = defaultdict(lambda: 0)`이다. defaultdict 모듈을 선언하면서 초깃값을 0으로 설정한 것이다. 현재 `lambda()` 함수를 배우지 않아 코드를 정확히 이해하기 어렵겠지만, 'return 0'이라고 이해하면 된다. 어떤 키가 들어오더라도 처음 값은 전부 0으로 설정한다는 뜻이다.

05. collections 모듈

■ defaultdict 모듈

- defaultdict의 초깃값은 [코드 7-7]처럼 리스트 형태로도 설정할 수 있다.

코드 7-7 defaultdict3.py

```
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])]
```

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

05. collections 모듈

■ Counter 모듈

- Counter 모듈은 시퀀스 자료형의 데이터 요소 개수를 딕셔너리 형태로 반환하는 자료구조이다. 즉, 리스트나 문자열과 같은 시퀀스 자료형 안의 요소 중 값이 같은 것이 몇 개 있는지 반환해 준다.

```
>>> 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
```

05. collections 모듈

■ Counter 모듈

- 기존 문자열값인 'gallahad'를 리스트형으로 변환한 후, text 변수에 저장하였다.
- c라는 Counter 객체를 생성하면서 text 변수를 초깃값으로 설정하고 이를 출력하면, 위 결과처럼 각 알파벳이 몇 개씩 있는지 쉽게 확인할 수 있다.
- c["a"]처럼 딕셔너리 형태의 문법을 그대로 이용해 특정 텍스트의 개수도 바로 출력할 수 있다.
- 앞서 defaultdict를 사용하여 각 문자의 개수를 셸는데, Counter를 이용하면 그런 작업을 매우 쉽게 할 수 있다.

05. collections 모듈

■ Counter 모듈

- 다음과 같이 코드를 작성하면 정렬까지 끝낸 결과물을 확인할 수 있는데, 이전 Lab에서 수행한 작업을 단 한 줄의 코드로 작성한 것을 확인할 수 있다.

```
>>> 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})
```

05. collections 모듈

■ Counter 모듈

- Counter 모듈은 단순히 시퀀스 자료형의 데이터를 세는 역할도 있지만, 딕셔너리 형태나 키워드형태의 매개변수를 사용하여 Counter를 생성할 수 있다.
- ➡ 먼저 딕셔너리 형태로 Counter 객체를 생성하는 방법이다. 다음 코드를 보면, {'red': 4, 'blue': 2}라는 초깃값을 사용하여 Counter를 생성한 것을 확인할 수 있다. 또한, elements() 함수를 사용하여, 각 요소의 개수만큼 리스트형의 결과를 출력하는 것을 확인할 수 있다.

```
>>> from collections import Counter
>>>
>>> c = Counter({'red': 4, 'blue': 2})
>>> print(c)
Counter({'red': 4, 'blue': 2})
>>> print(list(c.elements()))
['red', 'red', 'red', 'red', 'blue', 'blue']
```

05. collections 모듈

■ Counter 모듈

- 키워드 형태의 매개변수를 사용하여 Counter를 생성하는 방법이다. 매개변수의 이름을 키(key)로, 실제 값을 값(value)으로 하여 Counter를 생성할 수 있다.

```
>>> 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']
```

05. collections 모듈

■ Counter 모듈

- Counter는 기본 사칙연산을 지원한다. 파이썬에서 지원하는 기본 연산인 덧셈, 뺄셈, 논리 연산 등이 가능하다.

```
>>> 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})
```

05. collections 모듈

■ Counter 모듈

- + 기호는 두 Counter 객체에 있는 각 요소를 더한 것이고, & 기호는 두 객체에 같은 값이 있을 때, 즉 교집합의 경우에만 출력하였다. 반대로 | 기호는 두 Counter 객체에서 하나가 포함되어 있다면, 그리고 좀 더 큰 값이 있다면 그 값으로 합집합을 적용하였다.

```
>>> 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})
```


05. collections 모듈

■ namedtuple 모듈

- namedtuple 모듈은 튜플의 형태로 데이터 구조체를 저장하는 방법이다.

```
>>> 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

06. Lab: 텍스트 마이닝 프로그램

■ 실습 내용

- 앞에서 배운 딕셔너리와 Collections 모듈을 이용하여 텍스트 마이닝 프로그램을 만들어 보자.

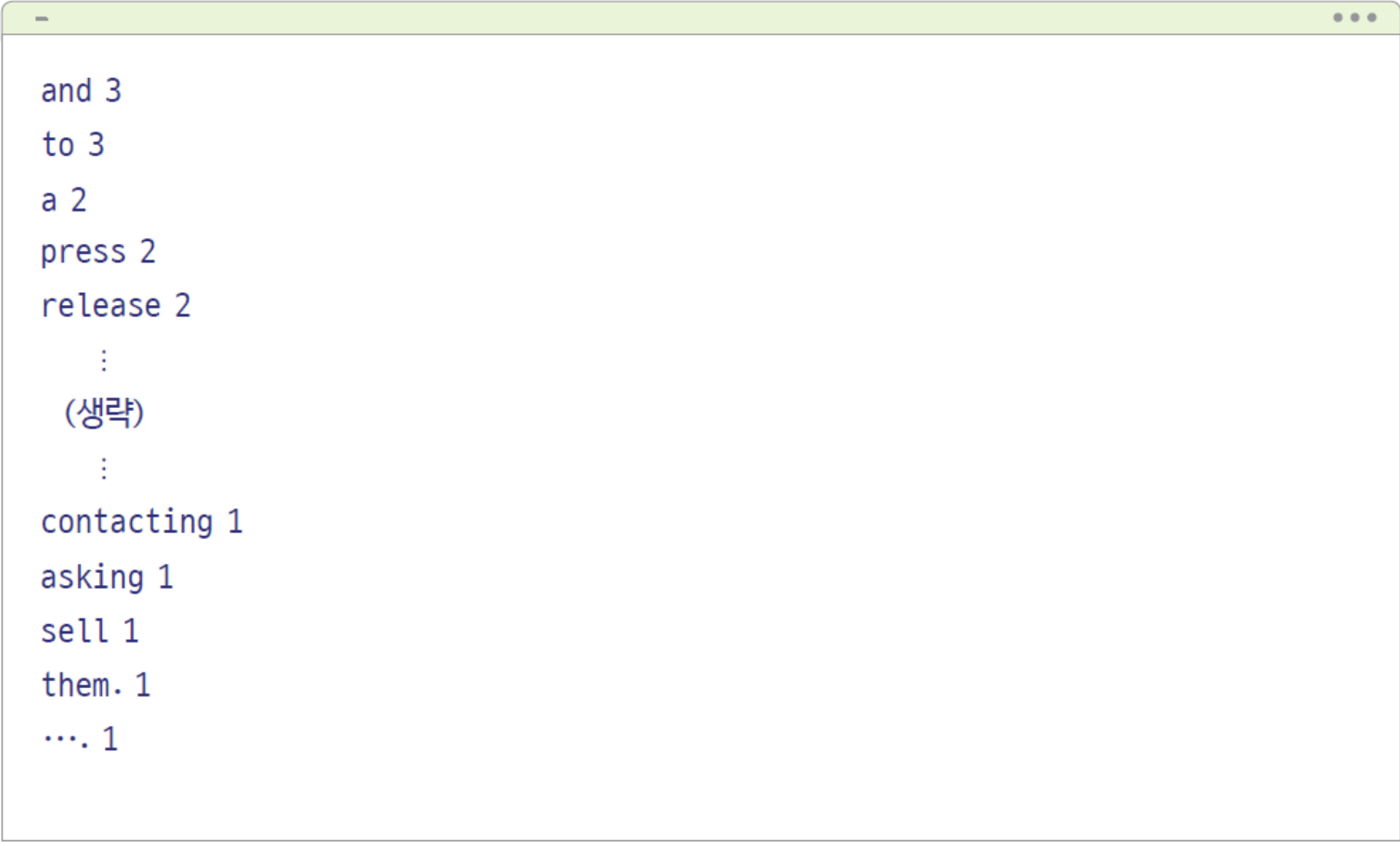
A press release is the quickest and easiest way to get free publicity. If well written, a press release can result in multiple published articles about your firm and its products. And that can mean new prospects contacting you asking you to sell to them. ...

- 이 프로그램을 작성하는 규칙은 다음과 같다.

- 문장의 단어 개수를 파악하는 코드를 작성한다.
- defaultdict 모듈을 사용한다.
- 단어의 출현 횟수를 기준으로 정렬된 결과를 보여 주기 위해 OrderedDict 모듈을 사용한다.

06. Lab: 텍스트 마이닝 프로그램

■ 실행 결과



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

06. Lab: 텍스트 마이닝 프로그램

■ 문제 해결

코드 7-8 textmining.py

```
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)
```

06. Lab: 텍스트 마이닝 프로그램

■ 문제 해결 : [코드 7-8] 해석

- 1행은 text 변수에 문장을 넣고, 이를 소문자로 바꾼 후 단어 단위로 자르는 코드이다. 이를 위해 lower()와 split() 함수를 연속으로 사용하였다. 이 코드의 결과를 확인하기 위해 파이썬 셸에 다음과 같이 입력하면 리스트의 결과를 볼 수 있다.

```
>>> 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.', '...']
```

06. Lab: 텍스트 마이닝 프로그램

■ 문제 해결 : [코드 7-8] 해석

- 다음으로 이 리스트에서 각각의 단어가 몇 개 있는지 헤아리는 코드가 필요하다. 3~7행을 보면 defaultdict 모듈을 사용하여 딕셔너리의 키값을 설정 없이 단어가 출현할 때마다 `word_count[word] += 1`을 통해 단어의 수를 증가시키는 것을 확인할 수 있다.
- 다음으로 단어의 출현 횟수를 기준으로 정렬된 결과를 보여 주고 싶다면, 9~13행과 같이 `OrderedDict` 모듈을 사용하여 코드를 구성할 수 있다.

Thank You !