



# Python으로 배우는 소프트웨어 원리

## Chapter 13. 데이터 수집과 시각화

# 목차

1. 라이브러리 개념
2. 데이터 시각화
3. 웹 데이터 수집

# 01

## 라이브러리 개념

# 01. 라이브러리 개념

## [라이브러리의 역할]

- 라이브러리는 방대한 자료를 모아 놓은 도서관처럼 많은 정보를 사용자가 이용하기 쉽게 모아둔 형태이다.
- 모듈과 라이브러리를 잘 활용할수록 효율이 높아진다.

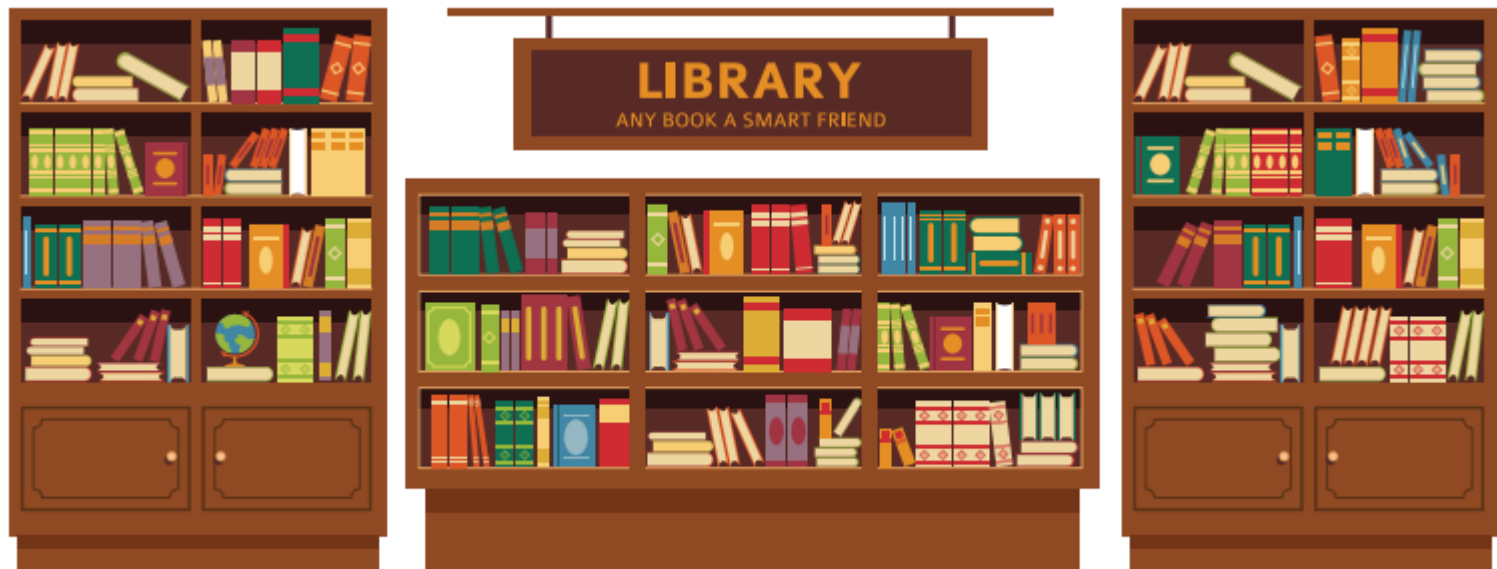


그림 13-1 라이브러리

# 01. 라이브러리 개념

## I. 모듈과 라이브러리

- 방대한 자료를 모아 놓은 도서관처럼 많은 정보를 사용자가 이용하기 쉽게 모아 둔 형태
- 표준 라이브러리는 파이썬에서 기본으로 제공 / 외부 라이브러리는 특정 용도에 맞게 필요한 기능을 제공
- 모듈은 변수나 함수, 클래스를 저장하고 있는 파일(라이브러리는 이러한 모듈들을 여러 개 묶어 놓은 형태)
- 이미지 처리나 데이터 시각화, 통계나 데이터 수집과 분석, 게임 등 필요한 영역의 용도에 알맞게 만들어 놓은 외부 라이브러리를 다양하게 사용 가능



그림 13-2 라이브러리 구조

# 01. 라이브러리 개념

## I. 모듈과 라이브러리

여기서 잠깐

라이브러리와 패키지의 차이

- 패키지는 여러 개의 모듈을 디렉토리에 모아 놓은 것
- 모듈을 넣어둔 디렉토리 명이 패키지명
- 패키지 안에 정의된 모듈을 사용할 때는 패키지 이름과 모듈 이름 사이에 '.'을 붙여 '패키지명.모듈명'과 같이 표시
- 라이브러리는 이러한 패키지와 모듈을 모두 포함하는 개념

표 13-1 패키지의 구조

최상위 디렉토리	하위 디렉토리나 모듈	모듈	
lives	animals	fish.py	→ <code>import lives.animals.fish</code>
		cat.py	
	plants	pine.py	
		rose.py	
	mushroom.py		→ <code>import lives.mushroom</code>

fish, cat 모듈을 참조하려면

`import lives.animals.fish`

mushroom 모듈을 참조하려면

`import lives.mushroom`

# 01. 라이브러리 개념

## II. 라이브러리 설치

- 필요한 외부 라이브러리를 설치하려면 pip라는 도구를 사용
- 윈도우의 명령 프롬프트를 실행한 후에 'pip'를 입력했을 때 다음과 같이 표시되면 사용 가능



```
명령 프롬프트
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Wjykim>pip

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze            Output installed packages in requirements format.
  list              List installed packages.
  show              Show information about installed packages.
  check             Verify installed packages have compatible dependencies.
  config            Manage local and global configuration.
  search            Search PyPI for packages.
```

그림 13-3 pip 입력

# 01. 라이브러리 개념

## II. 라이브러리 설치

실습 13-1

matplotlib 라이브러리 설치하기

- ① 'pip list' 라고 입력하면 표준 라이브러리를 제외하고 설치되어 있는 라이브러리들을 보여줌



```
명령 프롬프트
--use-deprecated <feature> Enable deprecated functionality, that will be removed in the future.

C:\Users\jykim>pip list
Package      Version
-----
pip          20.2.3
setuptools   49.2.1
WARNING: You are using pip version 20.2.3; however, version 21.0.1 is available.
You should consider upgrading via the 'c:\users\jykim\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.

C:\Users\jykim>
```

그림 13-5 pip list 입력





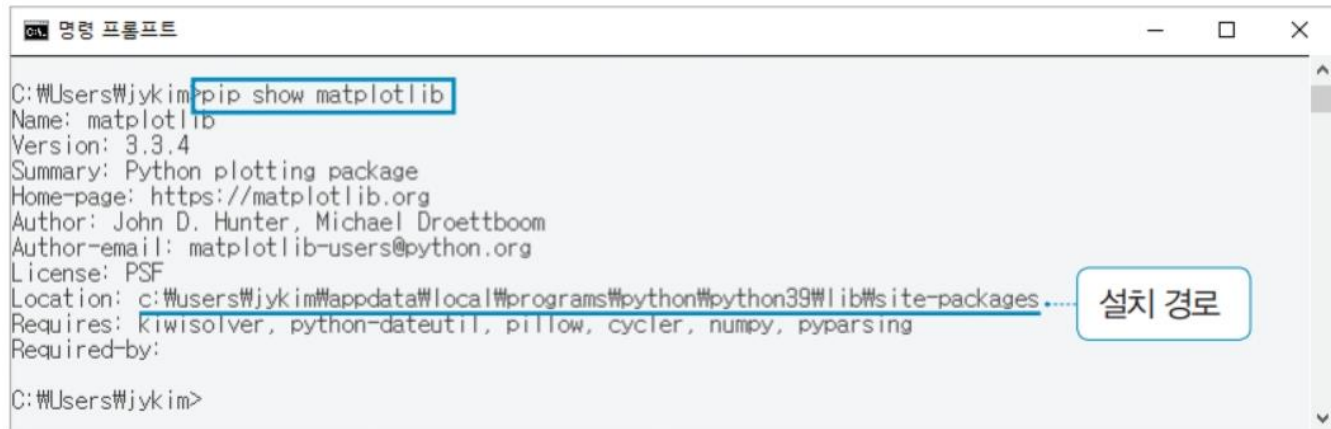
# 01. 라이브러리 개념

## II. 라이브러리 설치

실습 13-1

matplotlib 라이브러리 설치하기

- ③ 설치가 끝나면 'pip show matplotlib'으로 matplotlib 라이브러리가 설치된 곳을 확인



```
C:\Users\jykim> pip show matplotlib
Name: matplotlib
Version: 3.3.4
Summary: Python plotting package
Home-page: https://matplotlib.org
Author: John D. Hunter, Michael Droettboom
Author-email: matplotlib-users@python.org
License: PSF
Location: c:\users\jykim\appdata\local\programs\python\python39\lib\site-packages
Requires: kiwisolver, python-dateutil, pillow, cycler, numpy, pyparsing
Required-by:

C:\Users\jykim>
```

그림 13-7 pip show matplotlib 입력

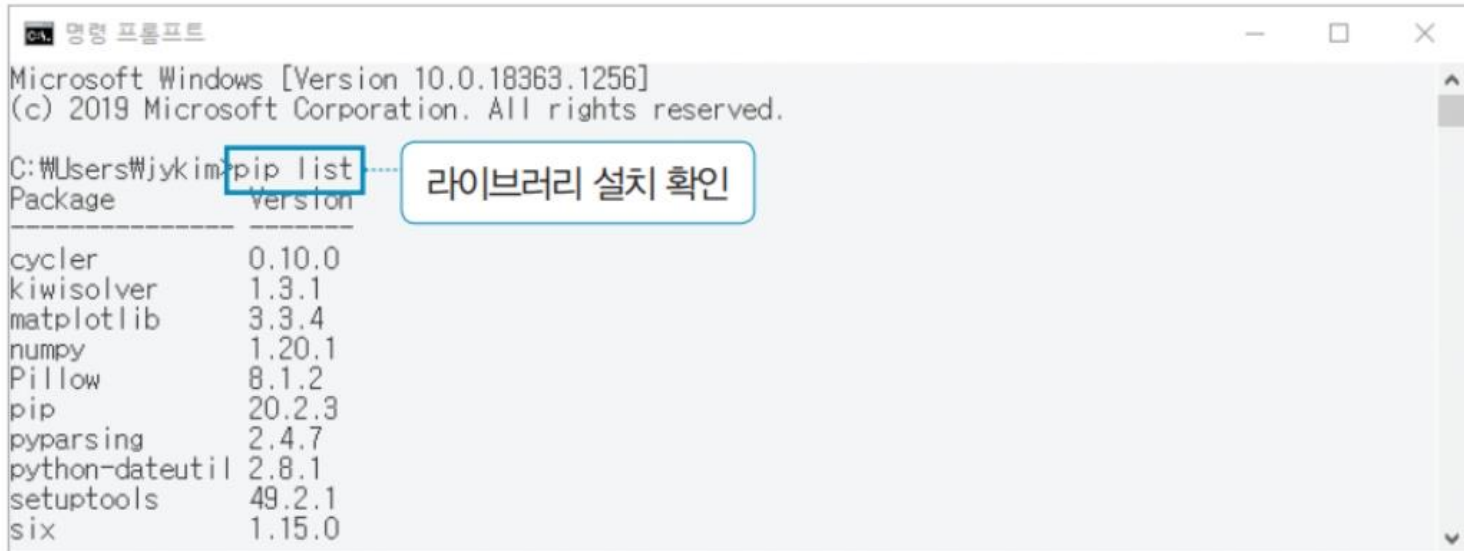
02

## 데이터 시각화

## 02. 데이터 시각화

### I. matplotlib 사용법

- 그래프와 같은 시각적 도구를 이용해 정보를 표현하는 과정이 데이터 시각화
- 데이터 시각화의 목적은 전달하려는 정보를 더 명확하고 효과적으로 표현
- matplotlib을 이용하면 파이썬에서 다루는 수치 데이터를 막대형, 선형, 분산형, 원형 등 다양한 그래프로 표시



```
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Wjykim>pip list
Package            Version
-----
cycler              0.10.0
kiwisolver          1.3.1
matplotlib          3.3.4
numpy               1.20.1
Pillow              8.1.2
pip                 20.2.3
pyparsing           2.4.7
python-dateutil     2.8.1
setuptools          49.2.1
six                 1.15.0
```

그림 13-9 matplotlib 설치 확인

## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-2

막대형 그래프 그리기

code13-02.py

- ① 그래프를 그리는 함수들이 정의되어 있는 pyplot 모듈을 사용하기 위해 import 문을 작성

```
import matplotlib.pyplot as plt
```

- ② 그래프로 표현할 데이터를 리스트에 저장

```
x = ['a', 'b', 'c', 'd', 'e']  
y = [10, 30, 50, 70, 90]
```

- ③ 리스트 x와 y의 값을 각각 x축(가로 방향)과 y축(세로 방향)으로 지정하여 막대형 그래프 생성

```
plt.bar(x, y)  
plt.show()
```

생성된 그래프의 모양을 그림 창에 표시  
그래프 설정이 끝난 후, 마지막으로 호출

## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-2

막대형 그래프 그리기

code13-02.py

- ④ 파일로 저장하고 실행하면 그래프와 확대, 축소 등 조작 메뉴를 보여주는 창이 생성

```
01 import matplotlib.pyplot as plt
02
03 x = ['a', 'b', 'c', 'd', 'e']
04 y = [10, 30, 50, 70, 90]
05
06 plt.bar(x, y)
07 plt.show()
```

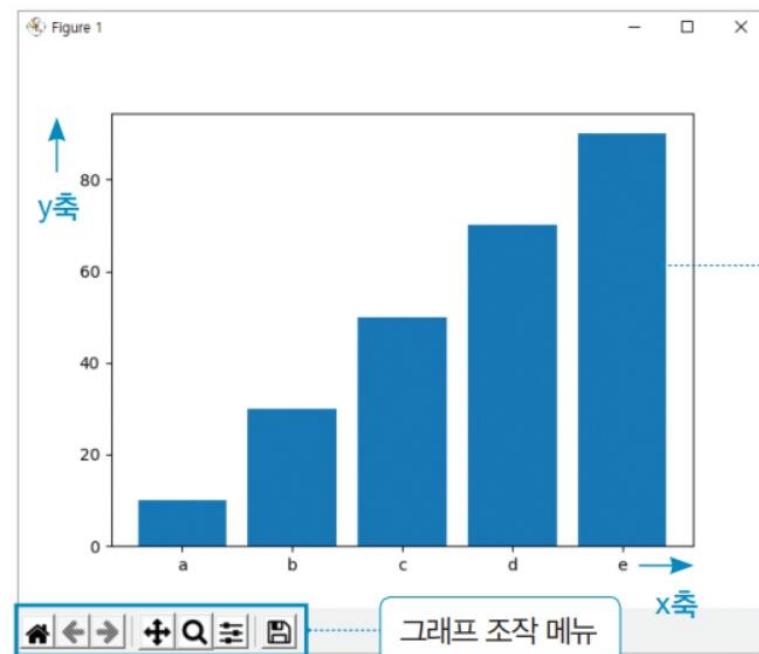


그림 13-11 실행 결과

## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-3

분산형 그래프 그리기

code13-03.py

- ① pyplot 모듈을 import하고, 데이터를 리스트에 저장

```
import matplotlib.pyplot as plt  
  
x = ['choi', 'han', 'jung', 'kim', 'lee']  
y = [93, 67, 90, 78, 80]
```

- ② 리스트의 값을 x축과 y축으로 지정하여 분산형 그래프를 그림

```
plt.scatter(x, y)  
plt.show()
```

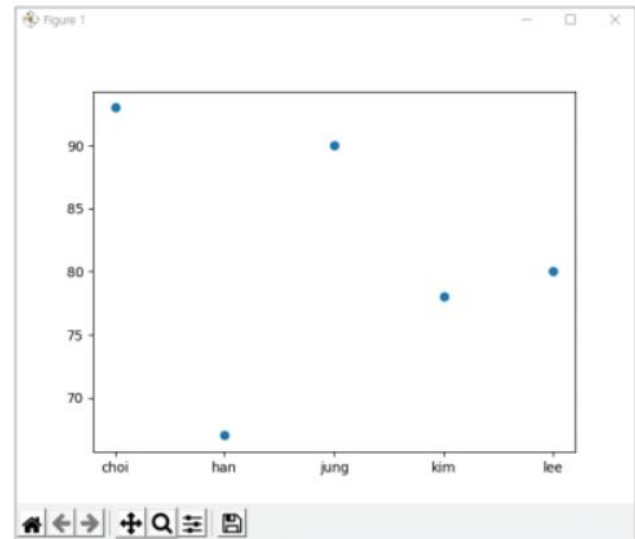


그림 13-13 그래프 생성

## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-3

분산형 그래프 그리기

code13-03.py

- ③ 그래프의 제목과 레이블, 눈금선이 표시되도록 코드를 추가

```
plt.title('Scores of Students')  
plt.xlabel('Student')  
plt.ylabel('Score')  
plt.grid()
```

단계 2의 코드 앞에 추가

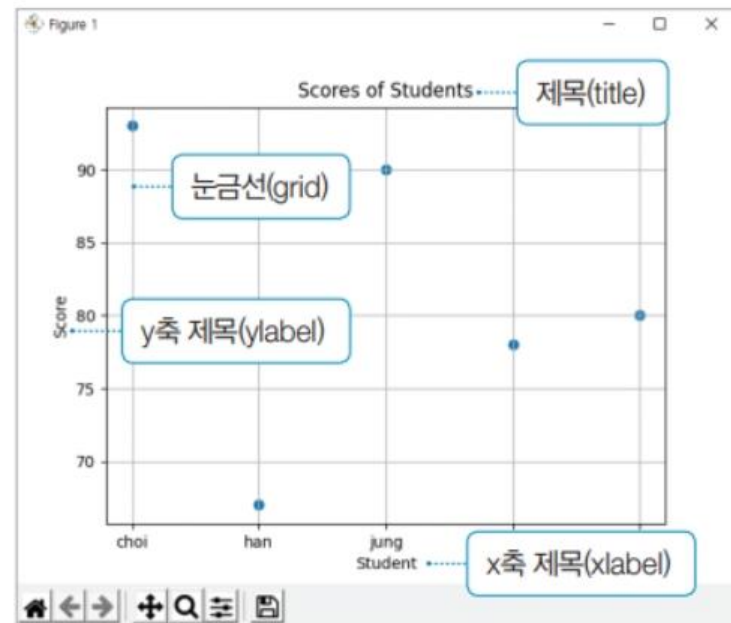


그림 13-14 그래프 요소 설정



## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-3

분산형 그래프 그리기

code13-03.py

- ④ 그래프의 색상, 마커의 모양과 크기를 설정한 전체 프로그램

```
01 import matplotlib.pyplot as plt
02
03 x = ['choi', 'han', 'jung', 'kim', 'lee']
04 y = [93, 67, 90, 78, 80]
05
06 plt.title('Scores of Students')
07 plt.xlabel('Student')
08 plt.ylabel('Score')
09 plt.grid()
10 plt.scatter(x, y, c='red', s=100, marker='>')
11 plt.show()
```

marker 인수를 지정하지 않으면 기본 마커 모양인 도트 형태('o')로 표현

# 눈금선

# 그래프 모양 수정

## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-4

선형 그래프 그리기

code13-04.py

- ① pyplot과 random 모듈을 import하고 데이터를 리스트에 저장

```
import matplotlib.pyplot as plt
import random

x = ['a', 'b', 'c', 'd', 'e']      # x축에 표시할 값
y = [10, 30, 50, 70, 90]        # 첫 번째 그래프에 사용할 데이터
```

- ② random.sample( )과 range( ) 함수를 이용해서, 0에서 100 사이의 정수 5개를 랜덤 선택하고 리스트에 저장

```
z = random.sample(range(0, 100), 5)  # 두 번째 그래프에 사용할 랜덤 데이터
```

## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-4

선형 그래프 그리기

code13-04.py

- ③ pyplot 모듈의 plot( )를 사용하여 x축과 y축에 표시할 리스트를 각각 지정하여 2개의 선형 그래프를 그림

```
plt.plot(x, y, x, z)  
plt.show()
```

혹은

```
plt.plot(x, y)  
plt.plot(x, z)  
plt.show()
```

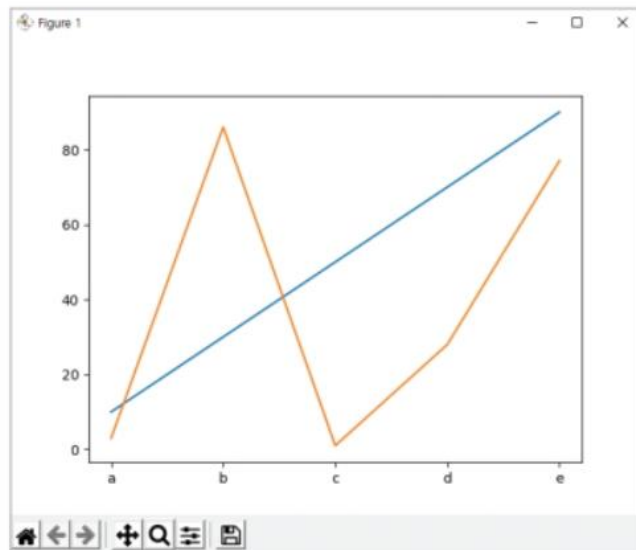


그림 13-16 그래프 생성

## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-4

선형 그래프 그리기

code13-04.py

- ④ 두 종류의 데이터를 구분하기 위해 범례를 표시하고, 그래프 색상이나 선 스타일, 마커의 색상과 크기, 모양을 다르게 설정

```
plt.plot(x, y, label='diagonal', marker='o')
plt.plot(x, z, label='random', c='red', linestyle='dashed', marker='s')
plt.legend()
plt.show()
```

# 범례 표시

# 단계 3의 코드를 수정

범례에 표시할 레이블 설정

## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-4

선형 그래프 그리기

code13-04.py

#### ⑤ 파일로 저장하고 실행

```
01 import matplotlib.pyplot as plt
02 import random
03
04 x = ['a', 'b', 'c', 'd', 'e']
05 y = [10, 30, 50, 70, 90]
06 z = random.sample(range(0, 100), 5)
07
08 plt.plot(x, y, label='diagonal', marker='o')
09 plt.plot(x, z, label='random', c='red', linestyle='dashed', marker='s')
10 plt.legend()                # 그래프의 모양에 따라 자동으로 범례 위치 설정
11 plt.show()
```

## 02. 데이터 시각화

### I. matplotlib 사용법

실습 13-4

선형 그래프 그리기

code13-04.py

#### ⑤ 파일로 저장하고 실행

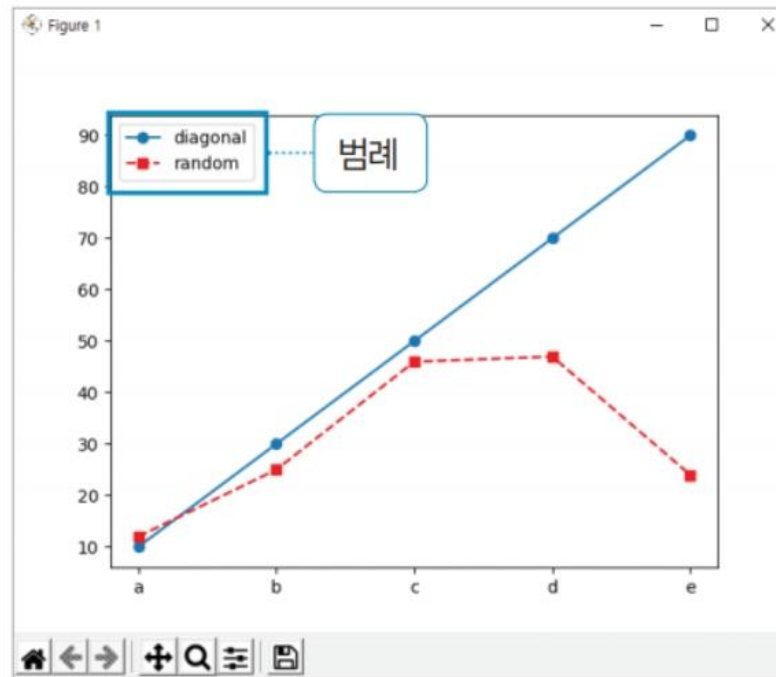


그림 13-17 범례를 추가한 선형 그래프

## 02. 데이터 시각화

### II. matplotlib의 활용

- 공공기관이 보유한 많은 데이터를 국민에게 개방하기 위해 웹사이트에 등록해둔 것이 공공데이터 포털
- 공공데이터 포털을 이용해 필요한 데이터를 찾고, 이를 시각화하는 데 파이썬의 matplotlib을 활용할 수 있음



그림 13-21 통합 공공데이터 포털(<https://www.data.go.kr/tcs/opd/ndm/view.do>)

## 02. 데이터 시각화

### II. matplotlib의 활용

실습 13-5

연도별 교통사고통계 그래프 그리기

code13-05.py

- ① 웹 브라우저를 실행하고  
공공데이터 포털에 접속  
하여 '연도별 교통사고'  
를 검색어로 입력  
  
검색 결과 목록에서 '한  
국도로공사\_교통사고통  
계'의 [다운로드] 버튼  
클릭

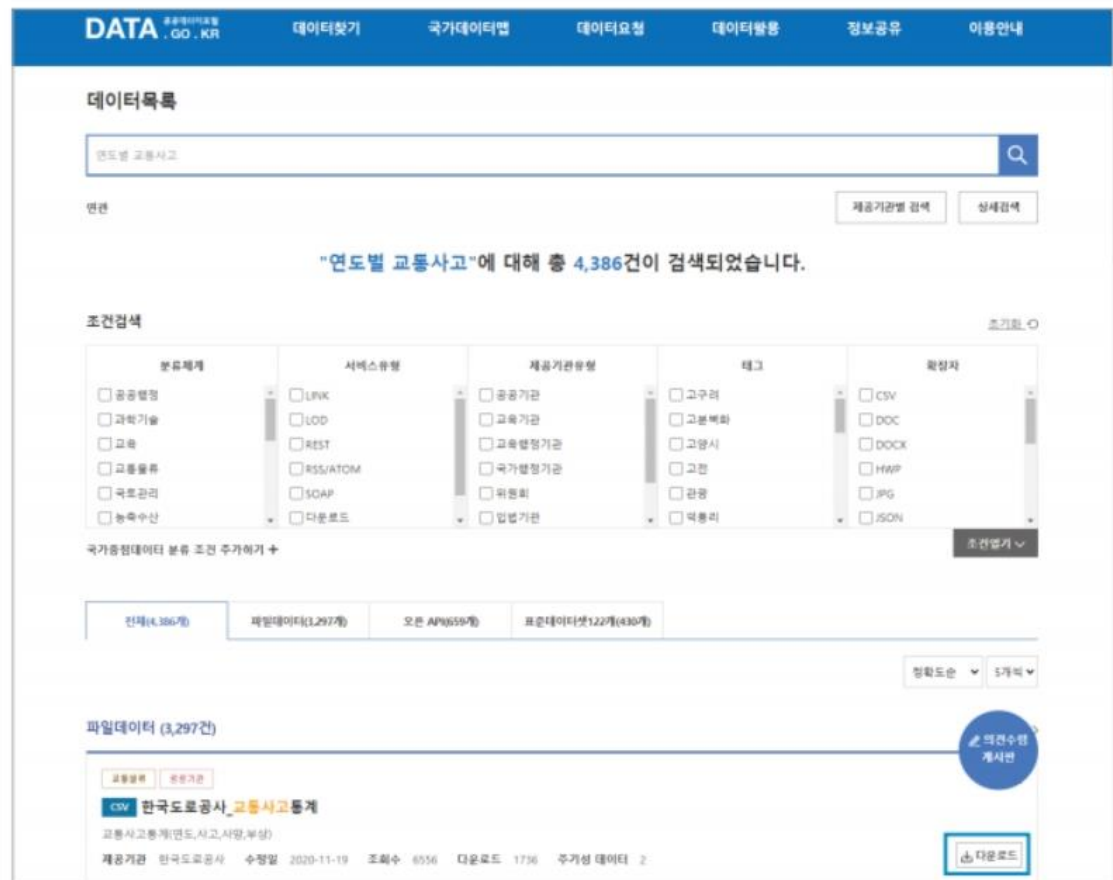


그림 13-23 교통사고통계 다운로드



## 02. 데이터 시각화

### II. matplotlib의 활용

실습 13-5

연도별 교통사고통계 그래프 그리기

code13-05.py

- ② 파일의 저장 경로는 파이썬 코드가 저장되는 디렉토리(C:\python으로 가정)  
[파일 이름]을 '교통사고통계'로 수정하고 [저장] 클릭

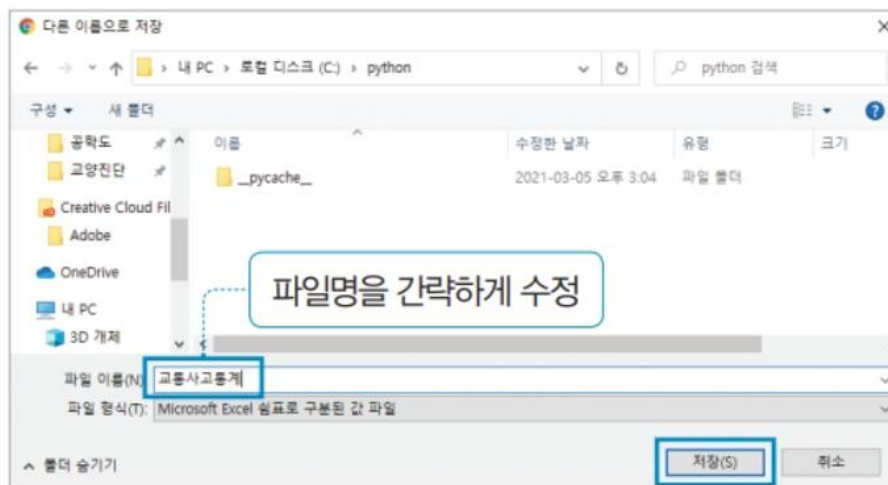


그림 13-24 파일 저장

✓ **TIP** 단계 1~2를 생략하고 제공되는 '교통사고통계.csv' 파일을 사용해도 됨

## 02. 데이터 시각화

### II. matplotlib의 활용

실습 13-5

연도별 교통사고통계 그래프 그리기

code13-05.py

- ③ 파이썬 코드 편집기를 열고 다음과 같이 입력하고 실행

csv 모듈의 reader( )를 사용하면 행 단위로 텍스트를 읽어 리스트에 저장

```
import csv

f = open("교통사고통계.csv", "r")
data = csv.reader(f) # 데이터를 읽어 리스트에 저장
for row in data :
    print(row)
f.close()
```

['연도',	'사고',	'사망',	'부상']
['2000',	'3910',	'569',	'2845']
['2001',	'3638',	'456',	'2331']
['2002',	'3957',	'421',	'2115']
['2003',	'3585',	'348',	'1843']
...			
['2018',	'2030',	'227',	'858']
['2019',	'1931',	'176',	'830']

## 02. 데이터 시각화

### II. matplotlib의 활용

실습 13-5

연도별 교통사고통계 그래프 그리기

code13-05.py

- ③ 단계3 의 코드를 수정해서 2개의 리스트에 '연도'와 '사고'의 값을 분리하여 저장

```
import csv

f = open("교통사고통계.csv", "r")
data = csv.reader(f)
next(data) # 제목 행 건너뛰기

x = []
y = []
for row in data:
    x.append(row[0])
    y.append(int(row[1])) # 정수형으로 변환
print(x)
print(y)

f.close()
```

필요한 데이터만  
추출해서 출력

```
['2000', '2001', '2002', '2003',  
'2004', '2005', '2006', '2007',  
'2008', '2009', '2010', '2011',  
'2012', '2013', '2014', '2015',  
'2016', '2017', '2018', '2019']
```

```
[3910, 3638, 3957, 3585, 3242,  
2880, 2583, 2550, 2449, 2374,  
2368, 2640, 2600, 2496, 2395,  
2251, 2195, 2145, 2030, 1931]
```

## 02. 데이터 시각화

### II. matplotlib의 활용

실습 13-5

연도별 교통사고통계 그래프 그리기

code13-05.py

- ⑤ 리스트 x와 y에 저장된 데이터를 파이썬 쉘에 출력하는 대신, 막대형 그래프로 표시하도록 수정

```
01 import matplotlib.pyplot as plt
02 import csv
03
04 f = open("교통사고통계.csv", "r")
05 data = csv.reader(f)
06 next(data)                # 제목 행 건너뛰기
07 x = []
08 y = []
09 for row in data:
10     x.append(row[0])
11     y.append(int(row[1]))
12 f.close()
13
14 plt.rcParams['font.family']='Malgun Gothic'           # 한글 폰트 설정
15 plt.figure(figsize=(12, 5), facecolor='lightyellow')  # 그림 크기(가로, 세로)와 배경색
```

## 02. 데이터 시각화

### II. matplotlib의 활용

실습 13-5

연도별 교통사고통계 그래프 그리기

code13-05.py

- ⑤ 리스트 x와 y에 저장된 데이터를 파이썬 쉘에 출력하는 대신, 막대형 그래프로 표시하도록 수정

```
16 plt.bar(x, y, color="darkgreen")
17 plt.xlabel('Year')
18 plt.ylabel('Number of accidents')
19 plt.title('연도별 교통사고통계')
20 plt.show()
```

03

## 웹 데이터 수집

## 03. 웹 데이터 수집

### I. BeautifulSoup 설치와 사용

- 파이썬 언어에서도 웹스크래핑이라고 부르는 웹 데이터 수집을 위한 여러 가지 라이브러리를 사용할 수 있음
- BeautifulSoup은 오픈소스 라이브러리로, 간편한 사용법이 장점이라 파이썬에서 웹 데이터를 수집할 때 많이 사용
- 웹 문서를 구성하는 HTML이나 XML 파일에 있는 내용을 검색해서 필요한 부분을 추출할 수 있음





## 03. 웹 데이터 수집

### I. BeautifulSoup 설치와 사용

실습 13-6

BeautifulSoup 설치와 기본 사용법 알기

code13-06.py

- ② 설치가 끝나면 'pip list'로 현재 설치되어 있는 라이브러리 목록에 beautifulsoup4, bs4 등 이 표시되는지 확인



```
C:\Users\Wjykim>pip list
Package            version
-----
beautifulsoup4     4.9.3
bs4                 0.0.1
cyclcr              0.10.0
kiwisolver          1.3.1
matplotlib          3.3.4
numpy               1.20.1
Pillow              8.1.2
pip                 20.2.3
pyparsing           2.4.7
python-dateutil     2.8.1
setuptools          49.2.1
six                 1.15.0
soupsieve           2.2
WARNING: You are using pip version 20.2.3; however, version 21.0.1 is available.
You should consider upgrading via the 'c:\Users\Wjykim\appdata\local\programs\python\python39\python
.exe -m pip install --upgrade pip' command.
C:\Users\Wjykim>
```

그림 13-27 beautifulsoup4와 bs4 확인

## 03. 웹 데이터 수집

### I. BeautifulSoup 설치와 사용

실습 13-6

BeautifulSoup 설치와 기본 사용법 알기

code13-06.py

- ③ 파이썬 쉘에서 설치한 라이브러리를 import

```
>>> from bs4 import BeautifulSoup
```

- ④ 제공하는 'example.html' 파일을 읽어 변수 html에 텍스트형으로 저장

```
>>> f = open("example.html", "r", encoding="utf-8")
>>> html = f.read()
```

- ⑤ 읽어온 텍스트는 HTML 형식으로 구문을 분석(html.parser)하고 BeautifulSoup 객체로 생성

```
>>> soup = BeautifulSoup(html, "html.parser")
```

## 03. 웹 데이터 수집

### I. BeautifulSoup 설치와 사용

여기서 잠깐

#### HTML 구문의 형식

- HTML은 웹 문서를 만들기 위한 표준 언어
- '<'와 '>'로 둘러싸인 태그와 속성, 속성값에 따라 구조와 표시 형식이 달라짐
- 텍스트를 웹 브라우저에서 어떻게 보여줄 것인지를 태그나 속성 정보로 결정



그림 13-29 HTML의 구조

## 03. 웹 데이터 수집

### I. BeautifulSoup 설치와 사용

실습 13-6

BeautifulSoup 설치와 기본 사용법 알기

code13-06.py

- ⑥ 구문 분석이 끝나면 `prettify()`로 읽어온 HTML 문서를 모두 볼 수 있고, `get_text()`로 태그를 제외한 텍스트만 추출할 수도 있음

```
>>> print(soup.prettify())
<html>
<head>
  <title>
    노래 가사 보기
  ...
</body>
</html>
```

```
>>> print(soup.get_text())
노래 가사 보기

I'm Not The Only One
Sam Smith
: In The Lonely Hour
You and me we made a vow
For better or for worse ... 중략...
```

- ⑦ 문서의 제목(title)을 추출

```
>>> soup.title
<title>노래 가사 보기</title>
```

```
>>> soup.title.string    # 텍스트만 가져오기
'노래 가사 보기'
```

## 03. 웹 데이터 수집

### I. BeautifulSoup 설치와 사용

실습 13-6

BeautifulSoup 설치와 기본 사용법 알기

code13-06.py

- ⑧ find( )에 특정 태그('a')를 지정해서 검색

```
>>> soup.find('a')
<a href="temp.html" id="singer">Sam Smith</a>
>>> soup.find('a').string          # 태그를 찾아서 텍스트만 가져오기
'Sam Smith'
```

```
>>> soup.find_all('a')
[<a href="temp.html" id="singer">Sam Smith</a>, <a href="temp2.html" id="album"><i> : In The
Lonely Hour</i></a>]
```

- ⑨ 2회 사용되고 있는 태그 중에서 id가 'album'인 항목을 가져오려면 다음과 같이  
입력

```
>>> soup.find('a', {'id': 'album'})
<a href="temp2.html" id="album"><i> : In The Lonely Hour</i></a>
>>> soup.find('a', {'id': 'album'}).string          # 텍스트(앨범 이름)만 가져오기
' : In The Lonely Hour'
```

## 03. 웹 데이터 수집

### I. BeautifulSoup 설치와 사용

여기서 잠깐

id와 class 속성값 지정하기

```
soup.find('a', {'id':'album'})
```

=

```
soup.find(id='album')
```

id 속성값

```
soup.find('p', {'class':'song'})
```

=

```
soup.find('p', 'song')
```

class 속성값

태그 이름

그림 13-30 속성값을 지정하는 다른 방법

## 03. 웹 데이터 수집

### I. BeautifulSoup 설치와 사용

실습 13-6

BeautifulSoup 설치와 기본 사용법 알기

code13-06.py

- ⑩ find( )나 find\_all( ) 대신에 select\_one( )이나 select( )를 사용

```
>>> soup.select_one('i')
<i> : In The Lonely Hour</i>
>>> soup.select('i')
[<i> : In The Lonely Hour</i>, <i>... 중략...</i>]
```

- ⑪ select\_one( )이나 select( )에도 태그 이름과 속성값을 이용

```
>>> soup.select_one('p.song')           # soup.find('p', 'song')과 같다.
<p class="song"><b>I'm Not The Only One</b></p>
>>> soup.select_one('p.song').string    # 텍스트(노래 제목)만 가져오기
"I'm Not The Only One"
```

## 03. 웹 데이터 수집

### I. BeautifulSoup 설치와 사용

실습 13-6

BeautifulSoup 설치와 기본 사용법 알기

code13-06.py

- ⑫ select( )나 select\_one( ) 함수는 중첩된 태그 구조를 이용해 필요한 값을 추출할 수 있어서 편리

```
<p class="music"><a href="temp.html" id="singer">Sam Smith</a>  
  <a href="temp2.html" id="album">✕✕ : In The Lonely Hour</i></a></p>  
<p class="lyrics">You and me we made a vow<br>  
  For better or for worse <i>... 중략...</i></p>
```

('p>a>i') 구조

상위 태그 이름

태그 이름

가져올 부분 ('p>i')

```
>>> soup.select_one('p > i').string  
'... 중략...'
```



## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-7

오늘의 뉴스 가져오기

code13-07.py

- ① <https://news.daum.net/ranking/popular> 링크로 이동  
파이썬의 표준 모듈인 urllib과 BeautifulSoup을 import

```
from urllib import request
from bs4 import BeautifulSoup
```

- ② urllib 모듈의 request.urlopen( ) 함수에 웹 주소를 넣어주면 해당 페이지의 내용을 가져오게 되고, HTML 구문 분석 후 객체가 생성

```
target = request.urlopen("https://news.daum.net/ranking/popular")
soup = BeautifulSoup(target, "html.parser")
```

- ③ 가져온 정보 중 웹 문서의 제목을 출력

```
print("*** {0} ***\n".format(soup.title.string))
```

```
*** 뉴스 랭킹 | 다음뉴스 ***
```

## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-7

오늘의 뉴스 가져오기

code13-07.py

- ④ 뉴스 타이틀을 모두 추출하기 위해 select( ) 함수를 사용

```
news = soup.select('div > strong > a')
```

여기서 잠깐

웹브라우저의 개발자 도구로 태그 이름 찾기

- 웹브라우저에서 제공하는 개발자 도구의 셀렉터를 이용하면 태그를 쉽게 찾을 수 있음
- ① 필요한 웹사이트에 연결하고 개발자 도구 메뉴를 선택(크롬이나 인터넷 익스플로러에서는 F12를 누름)

## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-7

오늘의 뉴스 가져오기

code13-07.py

여기서 잠깐

웹브라우저의 개발자 도구로 태그 이름 찾기

- ② 개발자 도구 창의 선택터 버튼( )을 클릭하고 태그 정보가 필요한 영역을 누르면, 개발자 도구 창에 해당 코드가 표시



그림 13-34 태그 찾기

## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-7

오늘의 뉴스 가져오기

code13-07.py

- ⑤ 태그를 제거하고 텍스트만 추출하기 위해 `get_text()`를 사용

```
rank = 1
for title in news :
    print("[{:2d}] {}".format(rank, title.get_text()))    # title.text와 동일
    rank += 1
```

- ⑥ 작성한 파이썬 프로그램을 실행

```
01 from urllib import request
02 from bs4 import BeautifulSoup
03
04 target = request.urlopen("https://news.daum.net/ranking/popular")
05 soup = BeautifulSoup(target, "html.parser")
06
07 print("*** {} ***\n".format(soup.title.string))
08 news = soup.select('div > strong > a')
09
10 rank = 1
```

## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-7

오늘의 뉴스 가져오기

code13-07.py

- ⑥ 작성한 파이썬 프로그램을 실행

```
11 for title in news :  
12     print("[{0:2d}] {1}".format(rank, title.get_text()))  
13     rank += 1
```

실습 13-8

무비챗 가져오기

code13-08.py

- ① 라이브러리를 import하고, 영화 사이트에 접속 → 연결한 웹페이지를 구문 분석한 후, 제목(title)을 출력 → 링크는 <http://www.cgv.co.kr/movies>를 사용

```
from urllib import request  
from bs4 import BeautifulSoup  
  
target = request.urlopen("http://www.cgv.co.kr/movies/")  
soup = BeautifulSoup(target, "html.parser")  
  
print('*** ' + soup.title.string + ' ***\n')
```

## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-8

무비챗 가져오기

code13-08.py

- ② 웹브라우저의 개발자 도구(F12)로 영화 제목의 태그 정보를 확인하고 select( )의 인수로 사용→ 영화 제목을 모두 가져와 리스트로 저장

```
movies = soup.select('strong.title')      # <strong> 태그의 "title" 속성값으로 추출
```

## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-8

무비차트 가져오기

code13-08.py

- ② 웹브라우저의 개발자 도구(F12)로 영화 제목의 태그 정보를 확인하고 select( )의 인수로 사용→ 영화 제목을 모두 가져와 리스트로 저장

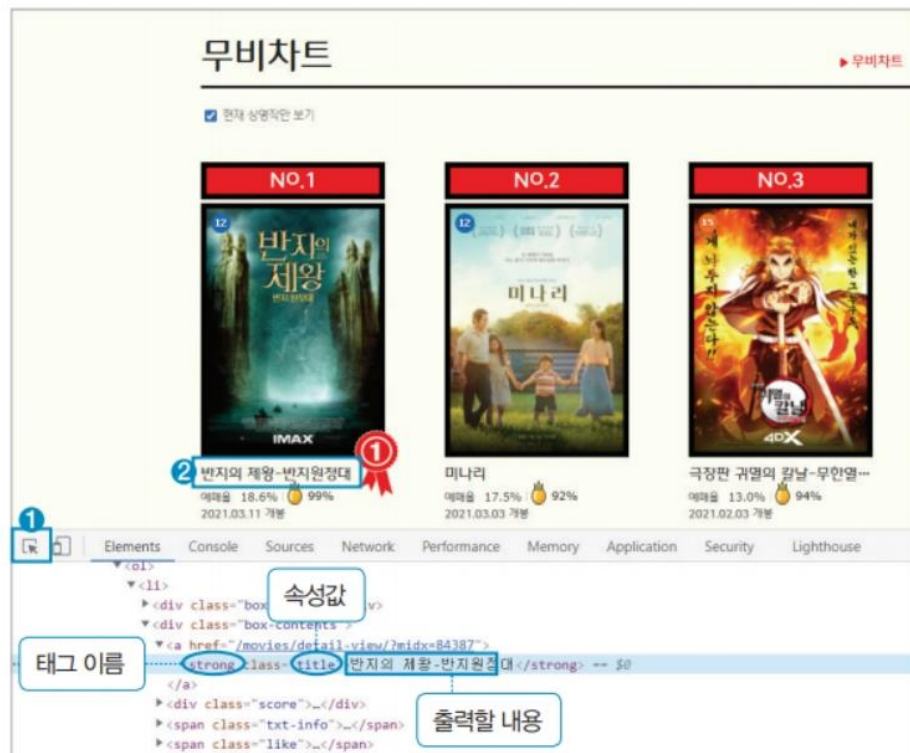


그림 13-36 영화 제목 추출

## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-8

무비챗 가져오기

code13-08.py

- ③ 웹 브라우저에서 예매율에 대한 태그도 확인하고 데이터를 추출

<strong>의 'percent' 하위에 있는 태그의 내용('18.6%')이 필요하기 때문에, select()의 인수를 다음과 같이 설정

```
rates = soup.select('strong.percent > span') # <strong>의 "percent" 하위에 있는 <span>
```



그림 13-37 예매율 추출



## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-8

무비챠트 가져오기

code13-08.py

- ④ 리스트 movies와 rates에 저장해 놓은 값들을 zip( ) 함수로 묶어서 출력

```
num = 1
for m in zip(movies, rates):
    title = m[0].text
    rate = m[1].text
    print("[{0}] {1} (예매율 {2})".format(num, title, rate))
    num = num + 1
```

- ⑤ 파일로 저장하고 실행

```
01 from urllib import request
02 from bs4 import BeautifulSoup
03
04 target = request.urlopen("http://www.cgv.co.kr/movies/")
05 soup = BeautifulSoup(target, "html.parser")
06
07 print('*** ' + soup.title.string + ' ***\n')
08
09 movies = soup.select('strong.title')
```

## 03. 웹 데이터 수집

### II. BeautifulSoup의 활용

실습 13-8

무비챗 가져오기

code13-08.py

#### ⑤ 파일로 저장하고 실행

```
10 rates = soup.select('strong.percent > span')
11
12 num = 1
13 for m in zip(movies, rates):
14     title = m[0].text
15     rate = m[1].text
16     print("[{0}] {1} (예매율 {2})".format(num, title, rate))
17     num = num + 1
```

# Thank You !

[Python]