제14장 MATPLOTLIB, PANDAS, REQUEST, SMTPLIB, SQLITE 사용해보기

파이썬 라이브러리

- 파이썬은 방대한 외부 라이브러리를 자랑한다. 현재 137,000개가 넘는 파이썬 라이브러리가 있다.

파이썬 라이브러리

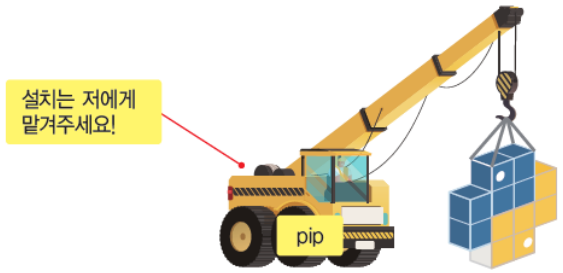
- 맷플롯립(Matplotlib): 몇 줄의 코드로 차트, 그래프, 파이 차트, 산점도, 히스토그램, 오류 차트 등을 그릴 수 있다.
- Seaborn: 히트 맵과 같은 통계 모델의 시각화와 관련하여 Seaborn은 신뢰할 수 있는 소스 중 하나이다.
- 판다스(Pandas): 데이터 분석 및 모델링과 같은 작업에 사용
- Requests: 웹에서 데이터를 가져올 때 사용하는 라이브러리이다.
- BeautifileSoup: HTML과 XML의 파서를 제공한다.

파이썬 라이브러리

- 넘파이(NumPy): 대규모 다차원 배열 및 행렬에 대한 지원을 제공하는 파이썬의 기본 패키지 중 하나이다.
- 필로우(Pillow): PIL(Python Imaging Library)와 호환성을 유지하면서 쉽게 사용할 수 있도록 한 라이브러리이다. 파이썬의 Tkinter 모듈과의 호환성도 가지고 있다.
- Scikit-learn: 머신러닝(기계 학습) 라이브러리이며 분류, 회귀, 클러스터링, 나이브 베이즈, K-Means clustering 등의 다양한 응용 프로그램에 효과적으로 사용할 수 있다.
- Tensorflow: 가장 인기 있는 딥러닝 프레임워크인 TensorFlow는 고성능 수치 계산을 위한 오픈 소스 소프트웨어 라이브러리이다.
- Keras: 심층 신경망을 빠르게 구현할 수 있도록 설계된 오픈 소스 신경망 라이브러리이다. Tensowflow를 백엔드로 사용할 수 있다.

외부 라이브러리 설치 방법

- 파이썬 패키지를 설치할 때 가장 많이 사용하는 도구가 바로 pip이다.




```
d:\>pip install Pillow
Collecting Pillow
  Downloading Pillow-3.3.0-cp35-cp35m-win32.whl (1.3MB)
    100% |################################| 1.3MB 867kB/s
Installing collected packages: Pillow
Successfully installed Pillow-3.3.0
You are using pip version 8.1.1, however version 8.1.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' comm
```

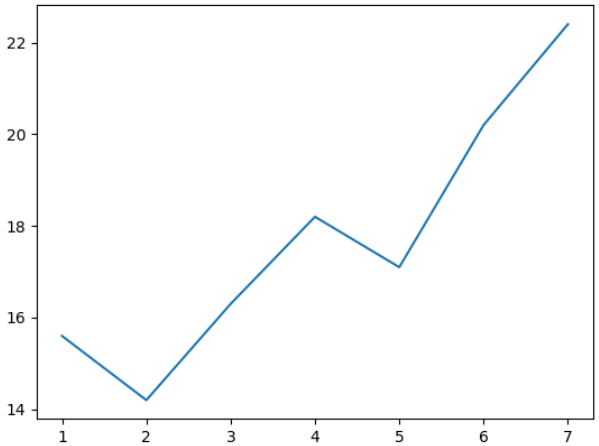
Matplotlib로 그래프를 그려보자.

- Matplotlib은 GNUplot처럼 그래프를 그리는 라이브러리이다. 최근에 파이썬의 인기가 아주 높기 때문에 Matplotlib도 많이 사용된다.

선 그래프

- 우리는 matplotlib의 하위 모듈인 pyplot을 사용한다. pyplot은 객체 지향적인 인터페이스를 제공한다.
- x값과 y값을 리스트 형태로 plot() 함수로 전달하면, plot() 함수는 이것으로 선 그래프를 그린다.

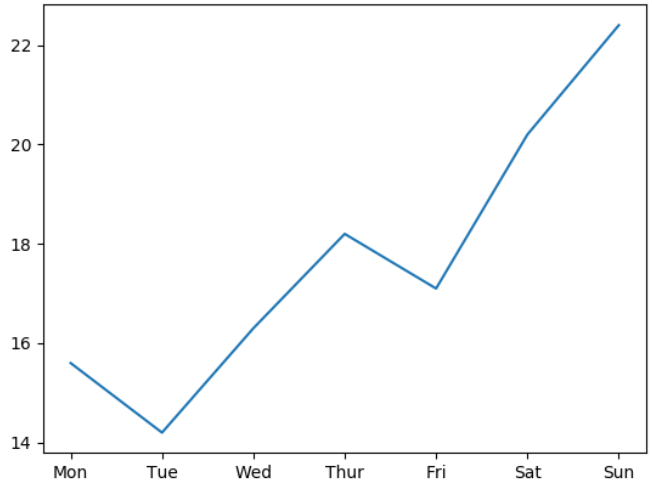
```
import matplotlib.pyplot as plt

X = [ 1, 2, 3, 4, 5, 6, 7]
Y = [15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4]

plt.plot(X, Y)                    # 선그래프를 그린다.
plt.show()
```



선 그래프

- 이번에는 좀 더 의미 있는 값을 x축에 표시해보자.

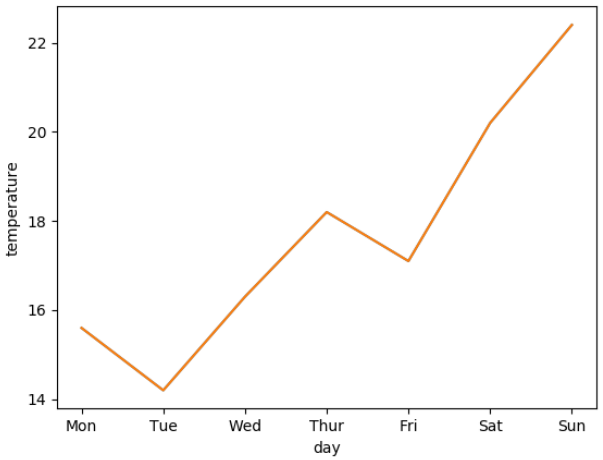
```
X = [ "Mon", "Tue", "Wed", "Thur", "Fri",  "Sat", "Sun" ]
Y = [15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4]

plt.plot(X, Y)
plt.show()
```



선 그래프

- 이번에는 x축과 y축에 레이블을 붙여보자.

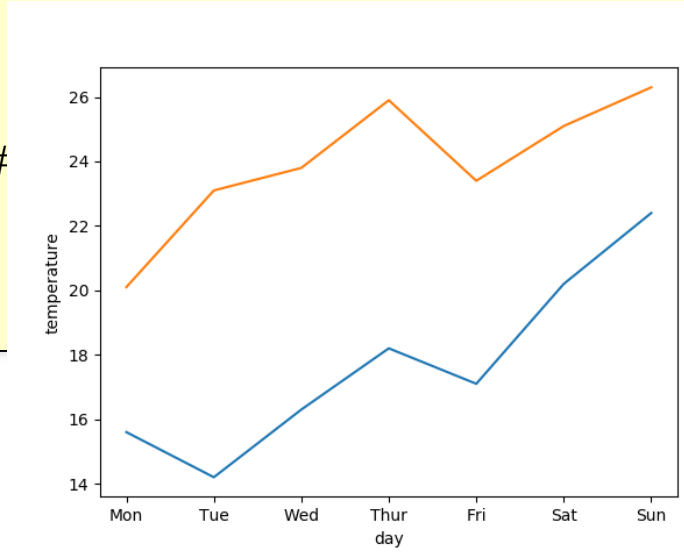
```
X = [ "Mon", "Tue", "Wed", "Thur", "Fri",  "Sat", "Sun" ]
Y = [15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4]

plt.plot(X, Y)
plt.xlabel("day")                          # x축 레이블
plt.ylabel("temperature")          # y축 레이블
plt.show()
```



선 그래프

- 이번에는 하나의 그래프에 2개의 값을 겹쳐서 표시해보자. y축에 해당하는 값을 추가로 입력하면 된다.

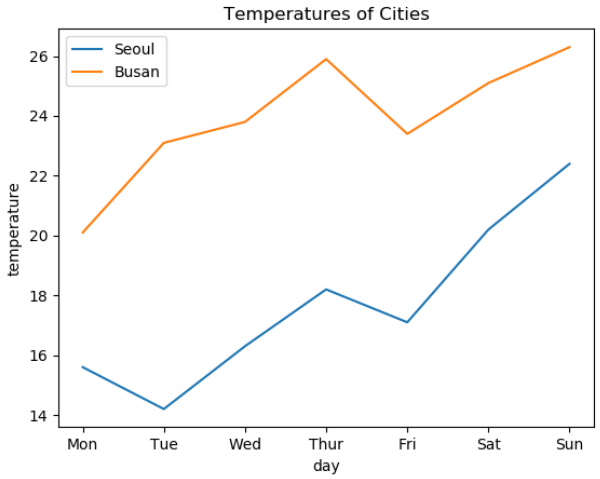
```
X = [ "Mon", "Tue", "Wed", "Thur", "Fri",  "Sat", "Sun" ]
Y1 = [15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4]
Y2 = [20.1, 23.1, 23.8, 25.9, 23.4, 25.1, 26.3]

plt.plot(X, Y1, X, Y2)                                  #
plt.xlabel("day")
plt.ylabel("temperature")
plt.show()
```



선 그래프

- 이번에는 레전드(legend)와 제목을 붙여보자. 레전드는 각 y축값이 무엇을 나타내는지를 설명한다.

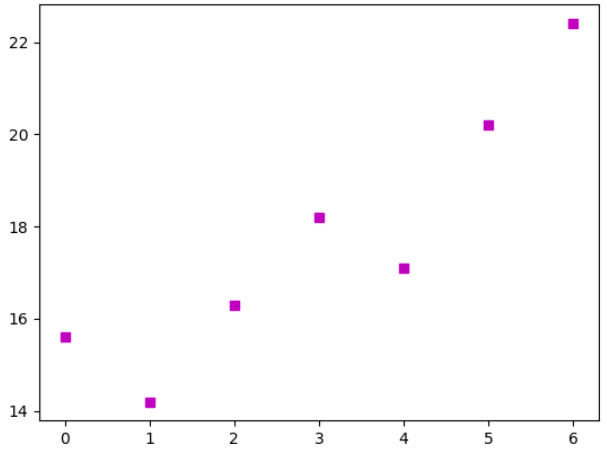
```
X = [ "Mon", "Tue", "Wed", "Thur", "Fri",  "Sat", "Sun" ]
Y1 = [15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4]
Y2 = [20.1, 23.1, 23.8, 25.9, 23.4, 25.1, 26.3]

plt.plot(X, Y1, label="Seoul")          # 분리시켜서 그려도 됨
plt.plot(X, Y2, label="Busan")          # 분리시켜서 그려도 됨
plt.xlabel("day")
plt.ylabel("temperature")
plt.legend(loc="upper left")            # 레전드
plt.title("Temperatures of Cities")     # 그래프의 제목
plt.show()
```



산점도

- 데이터 값만을 기호로 표시하고자 한다면 별도의 형식문자열을 전달하면 된다.

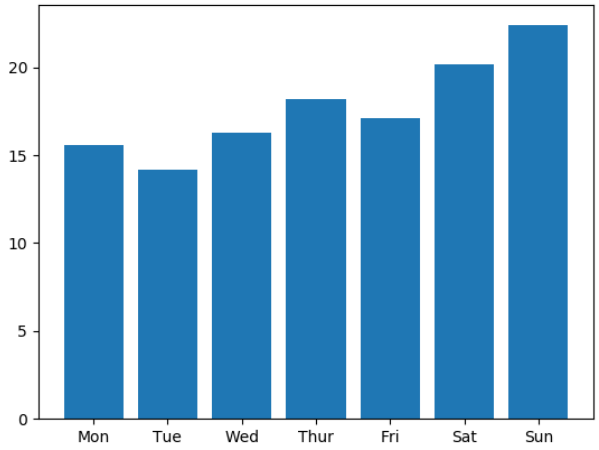
```
plt.plot([15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4], "sm")
plt.show()
```



막대 그래프

- 막대 그래프는 bar()를 호출하여 그린다.

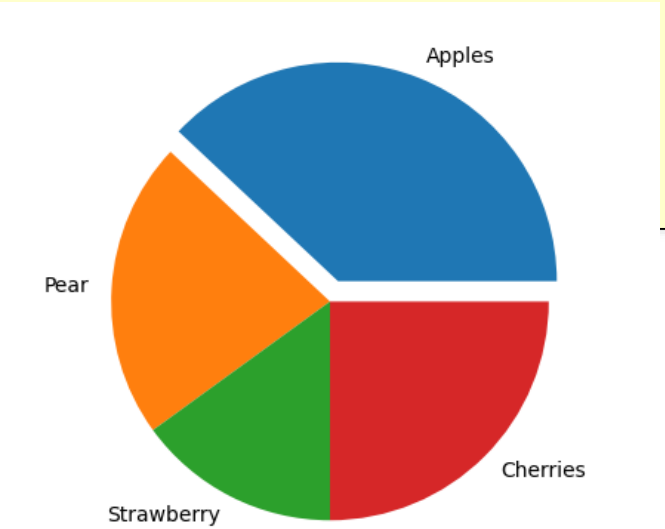
```
X = [ "Mon", "Tue", "Wed", "Thur", "Fri",  "Sat", "Sun" ]
Y = [15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4]
plt.bar(X, Y)                                    # 막대 그래프
plt.show()
```



파이챠트

- 파이 챠트는 pie()를 호출하여 그린다. 파이 챠트 중에서 하나의 파이가 눈에 띄기를 원한다면 explode 매개 변수(폭발처럼 떨어져 나가서 그려짐)를 이용한다.

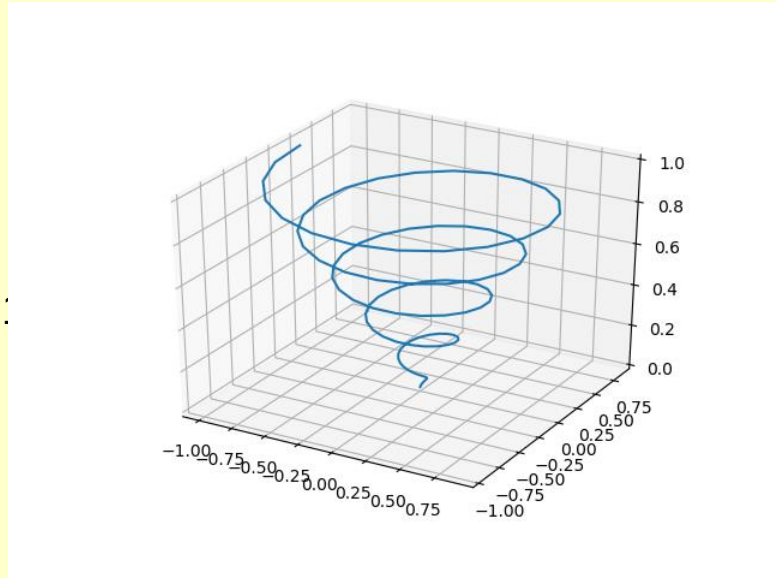
```
Y = [38, 22, 15, 25]
labels = ["Apples", "Pear", "Strawberry", "Cherries"]
explode = [0.1, 0, 0, 0]

plt.pie(Y, labels = labels, explode = explode)          # "
plt.show()
```



3차원 그래프

- 3차원 축은 projection = '3d' 키워드를 전달하여 다음 코드와 같이 생성할 수 있다. 배열 X에 x축 데이터가, 배열 Y에 y축 데이터가, 배열 Z에 z축 데이터가 저장된다.

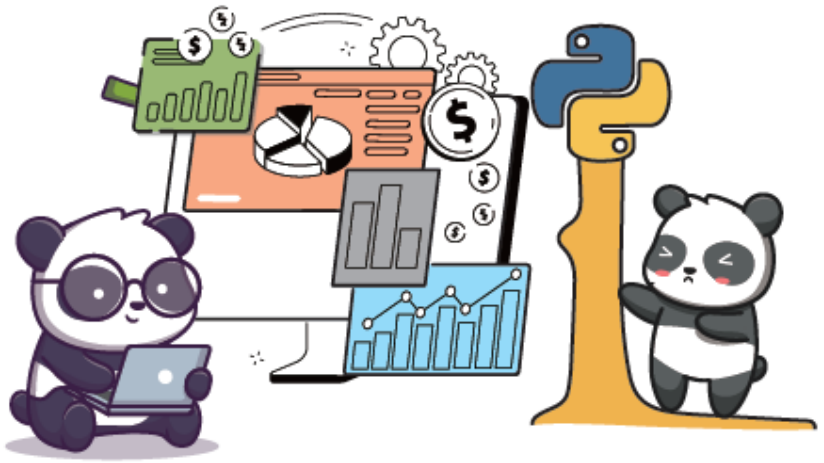
```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

# 3차원 축(axis)을 얻는다.
axis = plt.axes(projection='3d')
plt.show()

# 3차원 데이터를 넘파이 배열로 생성한다.
Z = np.linspace(0, 1, 100)                    # 0에서 1
X = Z * np.sin(30 * Z)
Y = Z * np.cos(30 * Z)

# 3차원 그래프를 그린다.
axis.plot3D(X, Y, Z)
```



판다스

- 데이터 과학자는 전통적으로 테이블 형태의 데이터를 선호한다. 2차원적인 행렬(matrix) 형태의 데이터는 개발자가 편리하게 각 요소나 행, 열에 접근할 수 있기 때문이다.

- 판다스는 데이터 프레임을 파이썬에 추가한다. 데이터 프레임은 행과 열로 구성되어 있는 테이블로서 다양한 함수를 지원한다. 행과 열에는 레이블(이름)이 붙어 있어서 우리가 원하는 행과 열에 쉽게 접근할 수 있다.



판다스로 데이터 읽기

- 데이터가 저장된 외부 파일을 찾고, 거기서 데이터를 불러온다. 국가에 대한 정보가 저장된 파일 countris.csv가 있다고 하자. 이 파일에는 몇 개 국가에 대한 정보가 저장되어 있다.

```
code,country,area,capital,population
KR,Korea,98480,Seoul,48422644
US,USA,9629091,Washington,310232863
JP,Japan,377835,Tokyo,127288000
CN,China,9596960,Beijing,1330044000
RU,Russia,17100000,Moscow,140702000
```

판다스로 데이터 읽기

```
>>> import pandas as pd
>>> df = pd.read_csv('countries.csv')

>>> df
  Unnamed: code country      area     capital  population
0         KR   Korea     98480       Seoul    48422644
1         US     USA   9629091  Washington   310232863
2         JP   Japan    377835       Tokyo   127288000
3         CN   China   9596960     Beijing  1330044000
4         RU  Russia  17100000      Moscow   140702000
```

인덱스와 컬럼 객체

- index는 행들의 레이블(label)이고 columns는 열들의 레이블이다.

index

columns

	code	country	area	eapital	polpulation
1	KR	Korea	98480	Seoul	48422644
2	US	USA	9629091	Washington	310232863
3	JP	Japan	377835	Tokyo	127288000
4	CN	China	9596960	Beijing	1330044000
5	RU	Russia	17100000	Moscow	140702000

인덱스 변경

```
>>> import pandas as pd

컬럼 0를 인덱스로 하겠다는 의미이다.
>>> df = pd.read_csv('countries.csv', index_col=0)
>>> df

   country      area     capital  population
KR   Korea     98480       Seoul    48422644
US     USA   9629091  Washington   310232863
JP   Japan    377835       Tokyo   127288000
CN   China   9596960     Beijing  1330044000
RU  Russia  17100000      Moscow   140702000
```

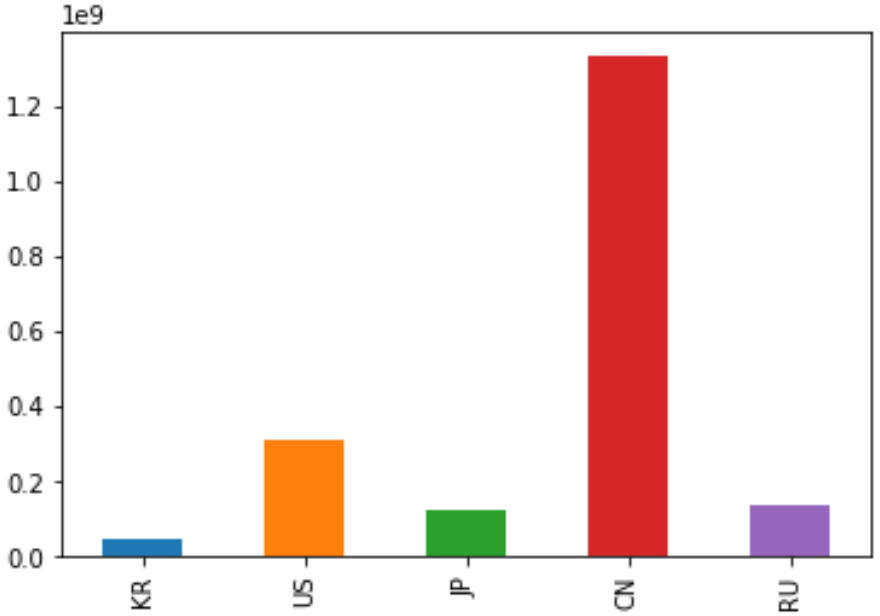
열 선택하기

```
>>> df['population']
KR      48422644
US     310232863
JP     127288000
CN    1330044000
RU     140702000
```

```
>>> df[  ['area', 'population']  ]
        area  population
KR     98480    48422644
US   9629091   310232863
JP    377835   127288000
CN   9596960  1330044000
RU  17100000   140702000
```

열 그래프로 그리기

```
>>> df['population'].plot(kind='bar')
```



행 선택하기

- 슬라이싱 표기법 사용

```
>>> df[:3]
  country     area     capital  population
KR   Korea    98480       Seoul    48422644
US     USA  9629091  Washington   310232863
JP   Japan   377835       Tokyo   127288000
```

```
>>> df.loc['KR']
country          Korea
area             98480
capital          Seoul
population     48422644
```

요소 선택하기

- 열 선택과 행 선택을 결합할 수도 있다.

```
>>> df['population'][:3]
KR     48422644
US    310232863
JP    127288000
```

```
>>> df.loc['US', 'capital']
'Washington'
```

간단히 데이터 분석하기

```
>>> df.describe()
              area    population
count  5.000000e+00  5.000000e+00
mean   7.360473e+06  3.913379e+08
std    7.185065e+06  5.333570e+08
min    9.848000e+04  4.842264e+07
25%    3.778350e+05  1.272880e+08
50%    9.596960e+06  1.407020e+08
75%    9.629091e+06  3.102329e+08
max    1.710000e+07  1.330044e+09
```

열 추가

```
>>> countries = pd.read_csv("countries.csv")
>>> countries["density"] =
countries["population"]/countries["area"]
>>> countries
  code country      area     capital  population     density
0   KR   Korea     98480       Seoul    48422644  491.700284
1   US     USA   9629091  Washington   310232863   32.218292
2   JP   Japan    377835       Tokyo   127288000  336.887795
3   CN   China   9596960     Beijing  1330044000  138.590137
4   RU  Russia  17100000      Moscow   140702000    8.228187
```

필터링

- 어떤 조건을 주어서 데이터 프레임을 필터링할 수 있다.

```
>>> df[df['density']>100]

  country     area  capital  population     density
KR   Korea    98480    Seoul    48422644  491.700284
JP   Japan   377835    Tokyo   127288000  336.887795
CN   China  9596960  Beijing  1330044000  138.590137
```

결손값 삭제하기

```
code,country,area,capital,population
KR,Korea,98480,Seoul,48422644
US,USA,9629091,Washington,310232863
JP,Japan,377835,Tokyo,127288000
CN,China,9596960,Beijing,1330044000
RU,Russia,17100000,Moscow,140702000
RU,Russia,17100000,Moscow,140702000

면적 데이터가 누락 되어 있다!
IN,India,,New Delhi,1368737513
```

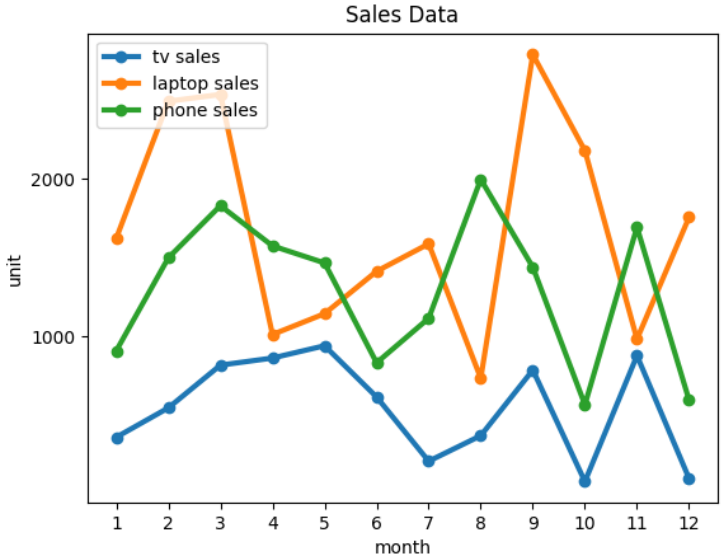
결손값 삭제하기

- 결손값을 다루는 가장 간단한 방법은 결손값을 가진 행을 삭제하는 것이다. 판다에서 dropna() 함수를 이용하여 삭제할 수 있다.

```
>>> df.dropna(how="all")
   country       area     capital  population

KR   Korea     98480.0       Seoul    48422644
US     USA   9629091.0  Washington   310232863
JP   Japan    377835.0       Tokyo   127288000
CN   China   9596960.0     Beijing  1330044000
RU  Russia  17100000.0      Moscow   140702000
```

Lab: 판매 데이터 시각화

month	tv	laptop	phone
1	363	1624	911
2	549	2493	1500
3	820	2536	1831
4	865	1014	1576
5	943	1146	1468
6	618	1415	835
7	211	1589	1117
8	373	737	1998
9	789	2789	1441
10	82	2180	569
11	880	985	1694
12	102	1757	599



Sol:

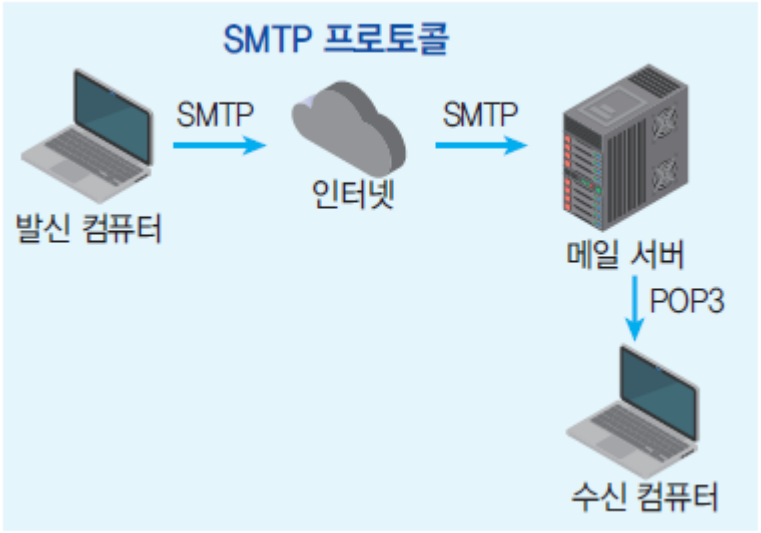
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("sales_data.csv")                    # CSV 파일을 읽는다.

monthList  = df ['month'].tolist()                    # ”month" 열을 추출하여서 리스트로 만든다.
tvData   = df ['tv'].tolist()                  # ”tv" 열을 추출하여서 리스트로 만든다.
laptopData   = df ['laptop'].tolist()
phoneData = df ['phone'].tolist()

# 각 리스트를 하나의 그래프에 중첩해서 그린다. 마커도 사용한다.
plt.plot(monthList, tvData,   label = 'tv sales', marker='o', linewidth=3)
plt.plot(monthList, laptopData, label = 'laptop sales',  marker='o', linewidth=3)
plt.plot(monthList, phoneData, label = 'phone sales', marker='o', linewidth=3)

# 그래프의 레이블, 레전드, 눈금을 설정한다.
plt.xlabel('month')
plt.ylabel('unit')
plt.legend(loc='upper left')
plt.xticks(monthList)
plt.yticks([1000, 2000])
plt.title('Sales Data')
plt.show()
```

엑셀 파일 읽어서 마케팅 이메일 보내기

- 이번 절에서는 엑셀 파일에 저장된 구매 정보를 읽어서 마케팅 이메일 보내보자. 이메일을 보낼 수 있어야 하고 엑셀 파일도 읽을 수 있어야 한다.

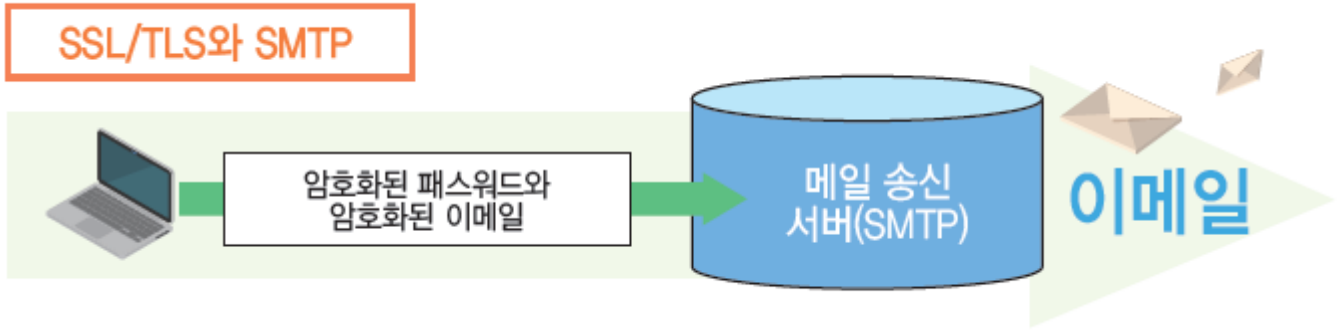
SMTP

- SMTP는 Simple Mail Transfer Protocol의 약자이다. SMTP는 메일 서버에서 발신자와 수신자 간에 메일을 보내고, 받고, 중계하는 데 사용하는 프로토콜이다.



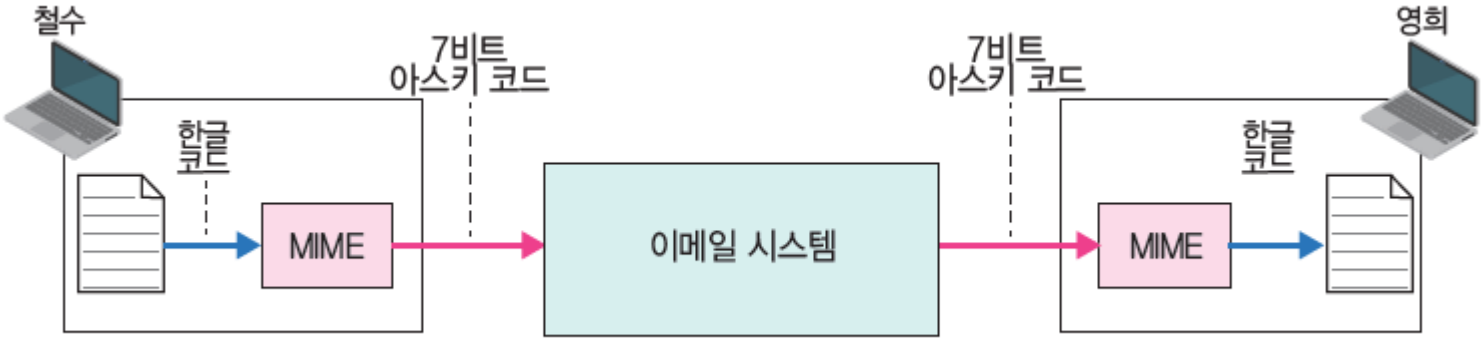
암호화 통신 프로토콜

- SMTP 서버의 암호화 방법에 따라 TLS(Transport Layer Security)와 SSL(Secure Sockets Layer)를 사용할 수 있다. TLS를 사용하면 smtplib.SMTP('서버이름', 포트번호)를 사용하고 SSL을 사용하는 경우에는 smtplib.SMTP_SSL('서버이름', 포트번호)을 사용한다.



MIME

- MIME (Multipurpose Internet Mail Extensions)은 ASCII 이외의 문자 집합의 텍스트와 오디오, 비디오, 이미지 및 응용 프로그램의 첨부 파일을 지원하도록 전자 메일 메시지 형식을 확장하는 인터넷 표준이다.

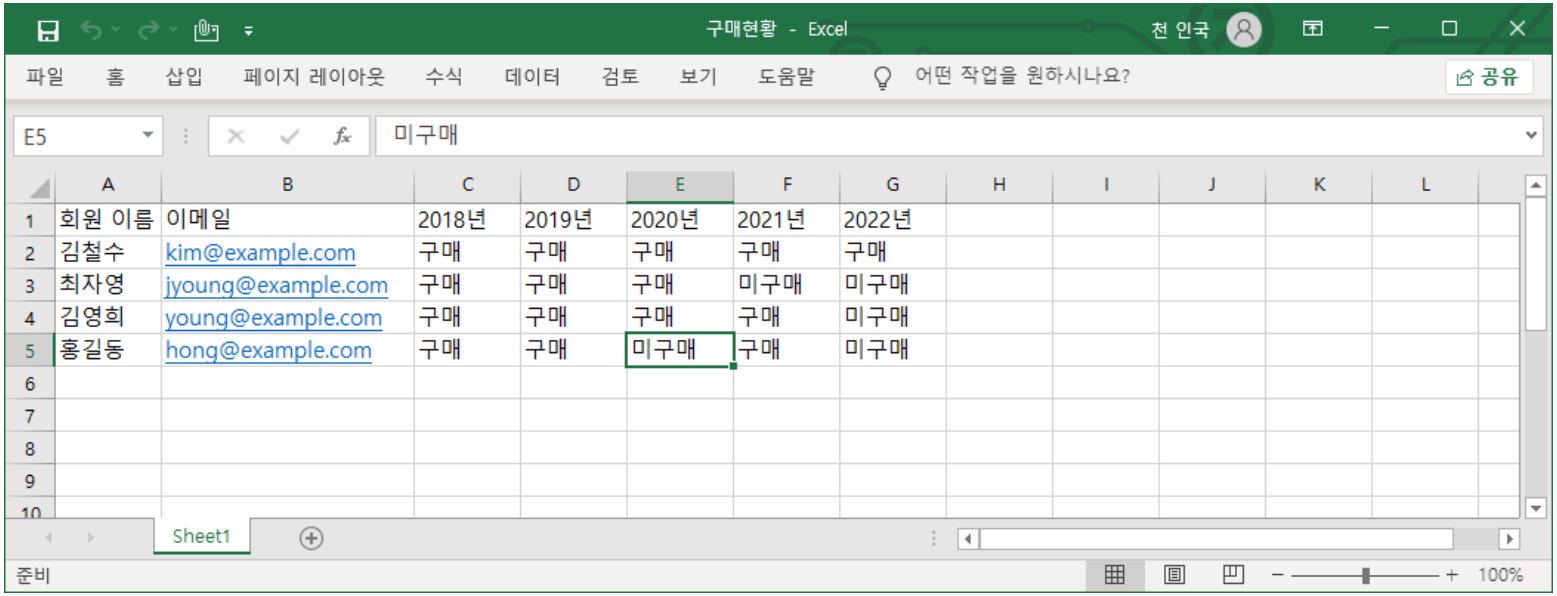


엑셀 파일 읽고 쓰기

```
>>> from openpyxl import Workbook
>>> workbook = Workbook()

통합 문서는 하나 이상의 워크시트로 생성된다. Workbook.active 속성을 사용하여 얻을 수 있다 .
>>> sheet = workbook.active

우리는 Workbook.create_sheet() 메소드를 사용하여 새 워크시트를 생성할 수 있다.
>>> sheet2 = workbook.create_sheet('Sheet1’)
>>> sheet['A1'] = 123

Worksheet.cell() 메소드를 사용하여도 된다. 이렇게 하면 행 및 열 표기법을 사용하여 셀에 액세스할 수 있다.
>>> sheet.cell(row=4, column=2, value=10)

통합 문서를 저장하는 가장 간단하고 안전한 방법은 Workbook개체 Workbook.save() 메서드를 사용하는 것이다.
>>> workbook.save("test.xlsx")
```

Lab: 구매 홍보 이메일 보내기

- 인터넷 쇼핑몰을 운영한다고 하자. 엑셀 파일에 회원들의 구매 여부가 저장되어 있다. 2022년도에 한 번도 구매하지 않은 회원을 찾아서 30% 할인 쿠폰을 보낸다고 하자.



	A	B	C	D	E	F	G
1	회원 이름	이메일	2018년	2019년	2020년	2021년	2022년
2	김철수	kim@example.com	구매	구매	구매	구매	구매
3	최자영	jyoung@example.com	구매	구매	구매	미구매	미구매
4	김영희	young@example.com	구매	구매	구매	구매	미구매
5	홍길동	hong@example.com	구매	구매	미구매	구매	미구매
6							
7							
8							
9							
10							

Sol.

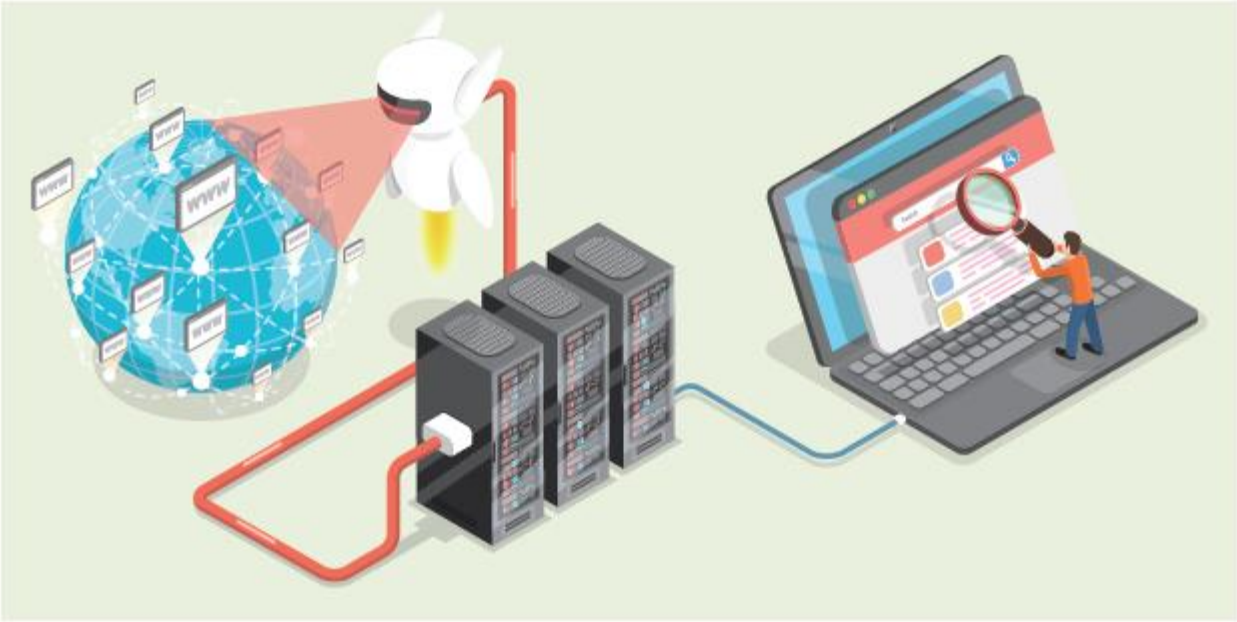
```
import openpyxl, smtplib, sys
from email.mime.text import MIMEText

workbook = openpyxl.load_workbook('구매현황.xlsx')
sheet = workbook.get_sheet_by_name('Sheet1')

members = {}
for r in range(2, sheet.max_row + 1):
    pay = sheet.cell(row=r, column=sheet.max_column).value

    if pay != '구매':
        name = sheet.cell(row=r, column=1).value
        email = sheet.cell(row=r, column=2).value
        members[name] = email
```

Sol.

```
네이버 메일 서버에 로그인한다. 이부분은 자신의 아이디로 변경하여야 한다.
session = smtplib.SMTP_SSL('smtp.naver.com', 465)
session.ehlo()
session.login('abc@naver.com', 'password')


딕셔너리에서 이름과 이메일을 꺼내서 메일을 작성하고 발송한다.
for name, email in members.items():
    msg = MIMEText(f'{name} 회원님, 30% 할인쿠폰이 발행되었습니다. \n 기간 내에 방문해주세요.\n 감사합니다.')
    msg['Subject']='할인쿠폰 증정 행사'
    msg['From']='abc@naver.com'
    msg['To']=email
    print(f'{email}에게 이메일을 보내는 중입니다.')
    status = session.sendmail('abc@naver.com', email, msg.as_string())

    if status != {}:
        print(f'이메일 {email} 전송에서 문제 {status}가 발생하였습니다.' )
session.quit()
```

Request와 BeautifulSoup

- 파이썬을 이용하면 웹페이지에서 데이터를 수집할 수 있다. 이것을 웹 크롤링(web crawling) 또는 웹 스크레이핑(Web Scraping)이라고 한다.



request 모듈

```
import requests

response = requests.get('https://www.naver.com/')
print(response.status_code)
print(response.text)
```

```
200

<!doctype html>                    <html lang="ko" data-dark="false"> <head> <meta
charset="utf-8"> <title>NAVER</title> <meta http-equiv="X-UA-Compatible"
content="IE=edge"> <meta name="viewport" content="width=1190"> <meta
name="apple-mobile-web-app-title" content="NAVER"/> ...
```

BeautifulSoup 모듈

- BeautifulSoup 모듈는 HTML과 XML 파일로부터 데이터를 뽑아내기 위한 파이썬 라이브러리이다. HTML 코드를 파싱하여서 트리 형태로 만들어준다. 우리는 이 트리에서 우리가 원하는 HTML 요소를 쉽게 추출할 수 있다.

BeautifulSoup 모듈

```
from bs4 import BeautifulSoup
import requests

response  = requests.get("http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=109")
print(response.text)
```

```
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
<channel>
<title>기상청 육상 중기예보</title>
...
```

BeautifulSoup 모듈

```
from bs4 import BeautifulSoup
import requests

response  = requests.get("http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=109")
soup = BeautifulSoup(response.content, 'html.parser')

for data in soup.select("location"):
   print(data.select_one("city").get_text())
   print(data.select_one("wf").get_text())
   print(data.select_one("tmn").get_text())
   print(data.select_one("tmx").get_text())
```

```
서울
맑음
4
15
인천
맑음
7
14
...
```

Lab: 우산 준비 이메일 보내기

- 이 실습에서는 웹 크롤링과 이메일 자동화를 합쳐보자. 어떤 지역의 날씨가 비가 오는 것으로 예보되면, 사용자한테 "우산 준비" 이메일을 보낸다. 우리는 기상청에서 서울 동작구 신대방동의 날씨 정보를 가져오자.

```
▼<description>
  ▼<header>
      <tm>202110231100</tm>
      <ts>3</ts>
      <x>59</x>
      <y>125</y>
    </header>
  ▼<body>
    ▼<data seq="0">
        <hour>15</hour>
        <day>0</day>
        <temp>17.0</temp>
        <tmx>17.0</tmx>
        <tmn>-999.0</tmn>
        <sky>1</sky>
        <pty>0</pty>
        <wfKor>맑음</wfKor>
        <wfEn>Clear</wfEn>
```

Sol.

```
from bs4 import BeautifulSoup
import requests
import openpyxl, smtplib, sys
from email.mime.text import MIMEText

response  =
requests.get("http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=1159068000")
soup = BeautifulSoup(response.content, 'html.parser')

weather = (soup.select('wfEn')[0]).text
temperature = (soup.select('temp')[0]).text

print(weather)
print(temperature)
```

Sol.

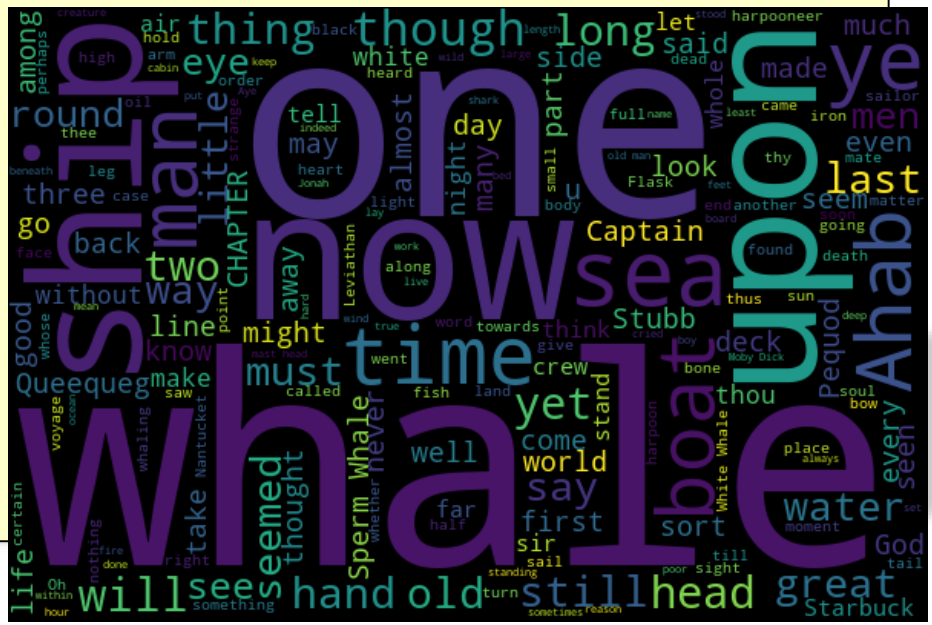
```
if weather == "Rain" or  weather == "Snow":
    session = smtplib.SMTP_SSL('smtp.naver.com', 465)
    session.ehlo()
    session.login('abc@naver.com', 'password')

    msg = MIMEText('비가 온다고 합니다.\n 우산을 준비하세요!\n 감사합니다.')
    msg['Subject']='우산 준비!'
    msg['From']='abc@naver.com'                          # 변경하여야 함!
    msg['To']='abc@naver.com'
    print('abc@naver.com에게 이메일을 보내는 중입니다.')
    status = session.sendmail('abc@naver.com', 'abc@naver.com', msg.as_string())

    if status != {}:
        print(f'abc@naver.com 이메일 전송에서 문제 {status}가 발생하였습니다.' )
    session.quit()
```

워드 클라우드 만들어보기

- 워드 클라우드(Word Cloud)는 패턴과 진화하는 추세를 이해하고 결정하기 위한 시각화 기술로 등장했다.

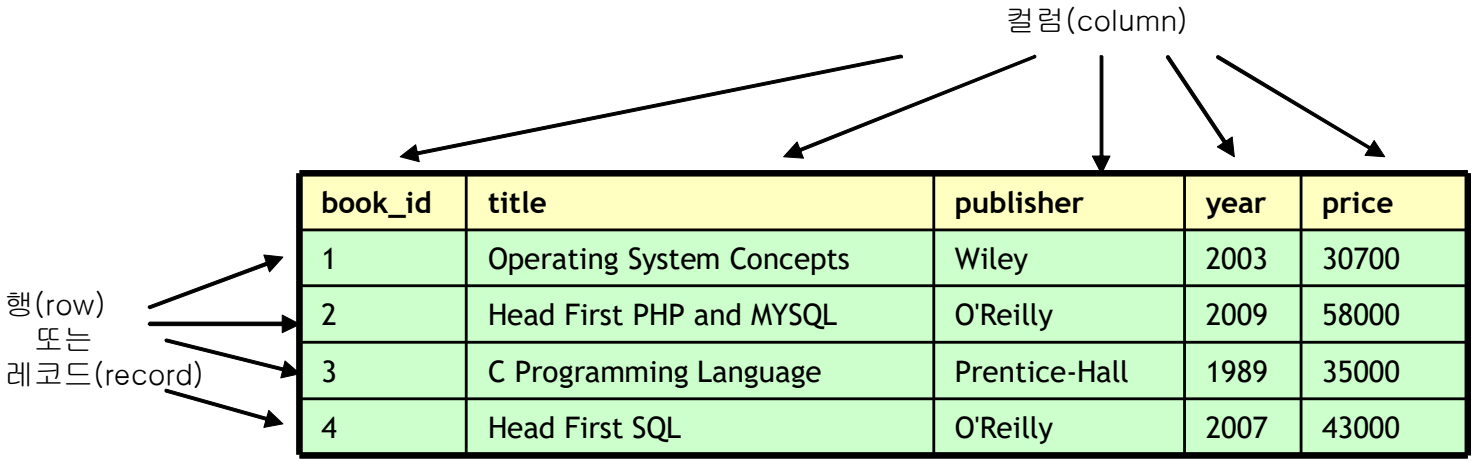
파이썬에서 워드 클라우드를 만드는 방법

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

text=""
with open("mobydick.txt", "r", encoding="utf-8") as f:
    lines = f.readlines()
    for line in lines:
            text += line

wc = WordCloud(width=600, height=400)

wc.generate(text)

wc.to_file("wc.png")
plt.figure(figsize=(30, 10))
plt.imshow(wc)
plt.show()
```



SQLite 데이터베이스

- SQLite는 이름에서도 알 수 있듯이 초경량급의 데이터베이스이다. 자세한 정보는 www.sqlite.org에서 얻을 수 있다. SQLite는 C언어로 작성된 효율적인 SQL 데이터베이스 엔진을 가지고 있다. SQLite는 별도의 서버 컴퓨터가 필요 없고 SQL 언어를 사용하여서 데이터베이스에 액세스할 수 있는 경량 데이터베이스이다.



book_id	title	publisher	year	price
1	Operating System Concepts	Wiley	2003	30700
2	Head First PHP and MYSQL	O'Reilly	2009	58000
3	C Programming Language	Prentice-Hall	1989	35000
4	Head First SQL	O'Reilly	2007	43000

SQL

구분	명령어	설명
데이터 정의 명령어 (Data Definition Language)	CREATE	사용자가 제공하는 컬럼 이름을 가지고 테이블을 생성한다. 사용자는 컬럼의 데이터 타입도 지정해야 한다. 데이터 타입은 데이터베이스에 따라 달라진다. 이미 테이블이 만들어져 있는 경우가 많기 때문에 CREATE TABLE은 통상적으로 DML보다 적게 사용된다.
	ALTER	테이블에서 컬럼을 추가하거나 삭제한다.
	DROP	테이블의 모든 레코드를 제거하고 테이블의 정의 자체를 데이터베이스로부터 삭제하는 명령어이다.
	USE	어떤 데이터베이스를 사용하는지 지정한다.
데이터 조작 명령어 (Data Manipulation Language)	SELECT	데이터베이스로부터 데이터를 쿼리하고 출력한다. SELECT 명령어들은 결과 집합에 포함시킬 컬럼을 지정한다. SQL 명령어 중에서 가장 자주 사용된다.
	INSERT	새로운 레코드를 테이블에 추가한다. INSERT는 새롭게 생성된 테이블을 채우거나 새로운 레코들를 이미 존재하는 테이블에 추가할 때 사용된다.
	DELETE	지정된 레코드를 테이블로부터 삭제한다.
	UPDATE	테이블에서 레코드에 존재하는 값을 변경한다.

SQL 사용하여 데이터 저장하기

```
import sqlite3
con = sqlite3.connect('test.db')
cur = con.cursor()

cur.execute('''CREATE TABLE product
            (id INTEGER, name TEXT, price INTEGER, qty INTEGER)''')

cur.execute("INSERT INTO product VALUES (1, '노트북', 1200000, 9) ")
cur.execute("INSERT INTO product VALUES (2, '데스크탑', 1600000, 6) ")
cur.execute("INSERT INTO product VALUES (3, '마우스', 20000, 100) ")
cur.execute("INSERT INTO product VALUES (4, '키보드', 50000, 65) ")
cur.execute("INSERT INTO product VALUES (5, 'CPU', 600000, 12) ")

con.commit()
con.close()
```

상품 아이디(id)	상품 이름(name)	가격(price)	재고수량(qty)
1	노트북	1200000	9
2	데스크탑 컴퓨터	1600000	6
3	마우스	20000	100
4	키보드	50000	65
5	CPU	600000	12

검색하기

```
import sqlite3

con = sqlite3.connect('test.db')
cur = con.cursor()
for row in cur.execute('SELECT * FROM product ORDER BY price'):
        print(row)
```

```
(3, '마우스', 20000, 100)
(4, '키보드', 50000, 65)
(5, 'CPU', 600000, 12)
(1, '노트북', 1200000, 9)
(2, '데스크탑', 1600000, 6)
```

업데이트 하기

```
import sqlite3
con = sqlite3.connect('test.db')
cur = con.cursor()
cur.execute("UPDATE product set price = 2000000 where id = 1")
con.commit()
for row in cur.execute('SELECT * FROM product'):
    print(row)

con.close()
```

```
(1, '노트북', 2000000, 9)
(2, '데스크탑', 1600000, 6)
(3, '마우스', 20000, 100)
(4, '키보드', 50000, 65)
(5, 'CPU', 600000, 12)
```

삭제 하기

```
import sqlite3
con = sqlite3.connect('test.db')
cur = con.cursor()
cur.execute("DELETE from product where id = 2")
con.commit()
for row in cur.execute('SELECT * FROM product'):
    print(row)

con.close()
```

```
(1, '노트북', 2000000, 9)
(3, '마우스', 20000, 100)
(4, '키보드', 50000, 65)
(5, 'CPU', 600000, 12)
```

Lab: 데이터베이스–> 웹페이지

- 데이터베이스에 저장된 데이터를 꺼내서 웹페이지로 만들어보자. 편의점의 현재의 재고를 데이터베이스로 저장하고 이것을 꺼내서 웹페이지로 보여준다.

Sol:

```
import sqlite3

con = sqlite3.connect('inventory.db')
cur = con.cursor()
cur.execute("CREATE TABLE stock (item char(100), number INTEGER, id INTEGER
PRIMARY KEY )")
cur.execute("INSERT INTO stock (item, number) VALUES ('eggs', 100)")
cur.execute("INSERT INTO stock (item, number) VALUES ('milk', 30)")
cur.execute("INSERT INTO stock (item, number) VALUES ('bread', 70)")
con.commit()
con.close()
print("데이터베이스가 생성되었습니다.")
```

Sol:

```
import sqlite3
from bottle import route, run

@route('/')
def itemlist():                                    # 웹 메인 페이지를 반환한다.
    con = sqlite3.connect('inventory.db')
    cur = con.cursor()

    html = "<h1> 재고 리스트</h1>"
    cur.execute("SELECT * FROM stock")
    result = cur.fetchall()

    for row in result:
        html += "<li>"+ row[0] + ": " + str(row[1]) + "개 </li>\n"

    con.close()
    return html

run()                                              # 웹서버 실행
```

Sol: