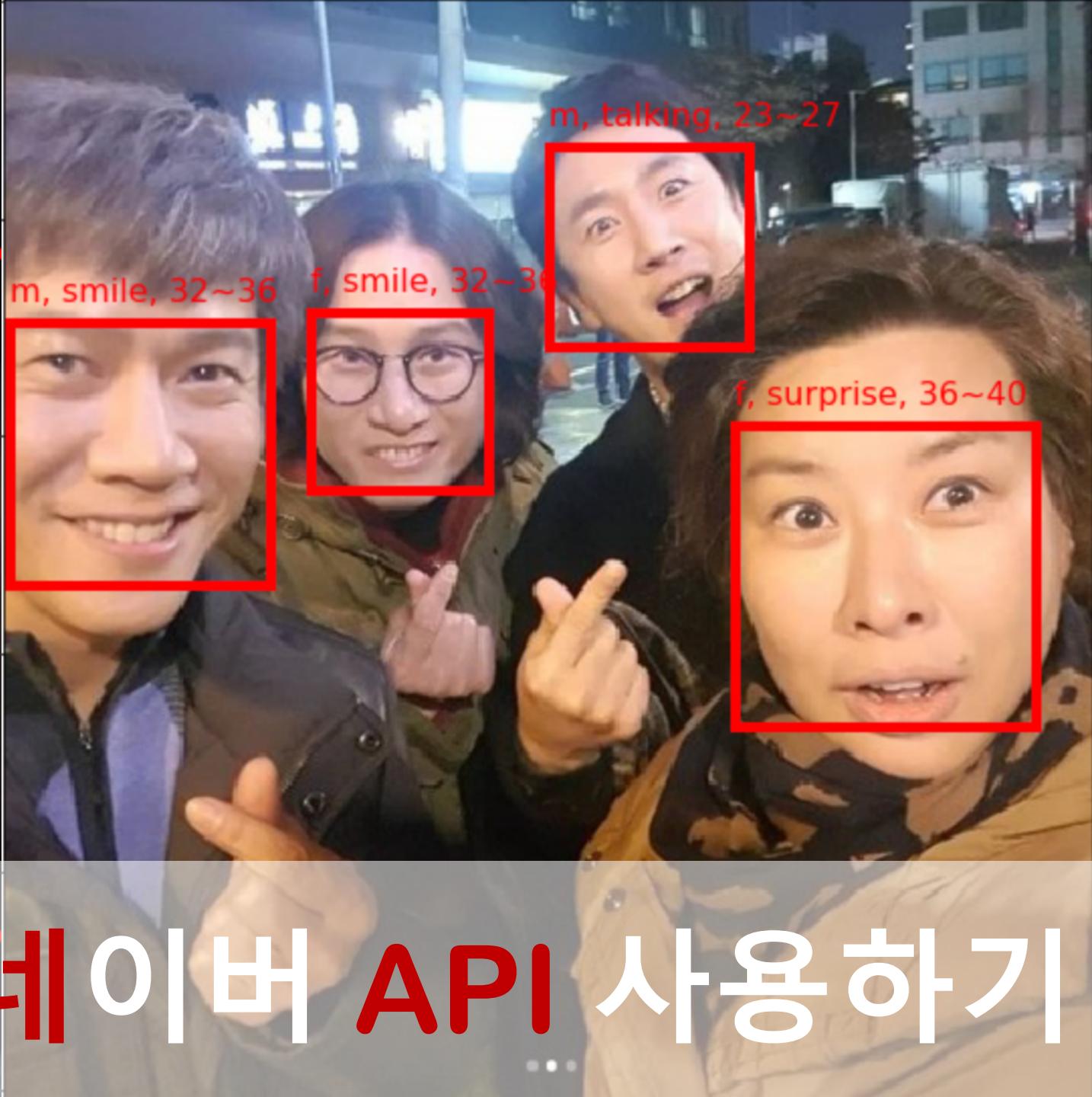


female : 0.996178
neutral : 0.999103
17~21 : 0.469397



네이버 API 사용하기

We use cookies on kaggle to deliver our services, analyze web traffic, and improve your experience on the site. By using kaggle, you agree to our use of cookies. Got it Learn more

kaggle Search kaggle Competitions Datasets Kernels Discussion Learn ... Sign In

The Home of Data Science & Machine Learning

Kaggle helps you learn, work, and play

jobs board >

What's Kaggle?

- 수업전에 유용한 사이트 하나 소개
- kaggle.com**
- 학습자에게 필요한 데이터 제공
- 더 나아가 고수들의 코드를 들여다 볼 수 있는 절호의 찬스

Create an account or Host a competition

Public

Sort by

Hotness

7,166 Datasets

Sizes

File types

Licenses

Tags

Search datasets



325



Data Science For Good: DonorsChoose.org

Help DonorsChoose.org connect donors with projects they care about

DonorsChoose.org updated 10 days ago

education
crowdfunding
marketing
+ 2 more...

CSV
1 GB
CC0

97
26
45k

176



Gun Violence Data

Comprehensive record of over 260k US gun violence incidents from 2013-2018

James Ko updated a month ago

crime
law
violence
social sciences

CSV
34 MB
Other

25
3
25k

61



Honey Production In The USA (1998-2012)

Honey Production Figures and Prices by National Agricultural Statistics Service

Jessica Li updated a month ago

animals
natural resources
agriculture

CSV
80 KB
CC0

18
2
11k

28



Grammar and Online Product Reviews

A list of 71,045 online reviews from 1,000 different products.

Datafiniti updated 3 months ago

databases
grammar
product
+ 2 more...

CSV
9 MB
CC4

4
0
3k

741



Data Science for Good: Kiva Crowdfunding

Use Kernels to assess welfare of Kiva borrowers for \$30k in prizes

Kiva updated 3 months ago

geography
finance
lending
+ 2 more...

CSV
42 MB
CC0

225
46
121k



I,Coder

Terrorism Around The World

243
voters



last run 3 months ago · IPython Notebook HTML · 15,928 views
using data from [Global Terrorism Database](#) · Public

[Notebook](#)

[Code](#)

[Data \(1\)](#)

[Output](#)

[Comments \(53\)](#)

[Log](#)

[Versions \(44\)](#)

[Forks \(296\)](#)

[Fork Notebook](#)

Tags

[data visualization](#)

[geography](#)

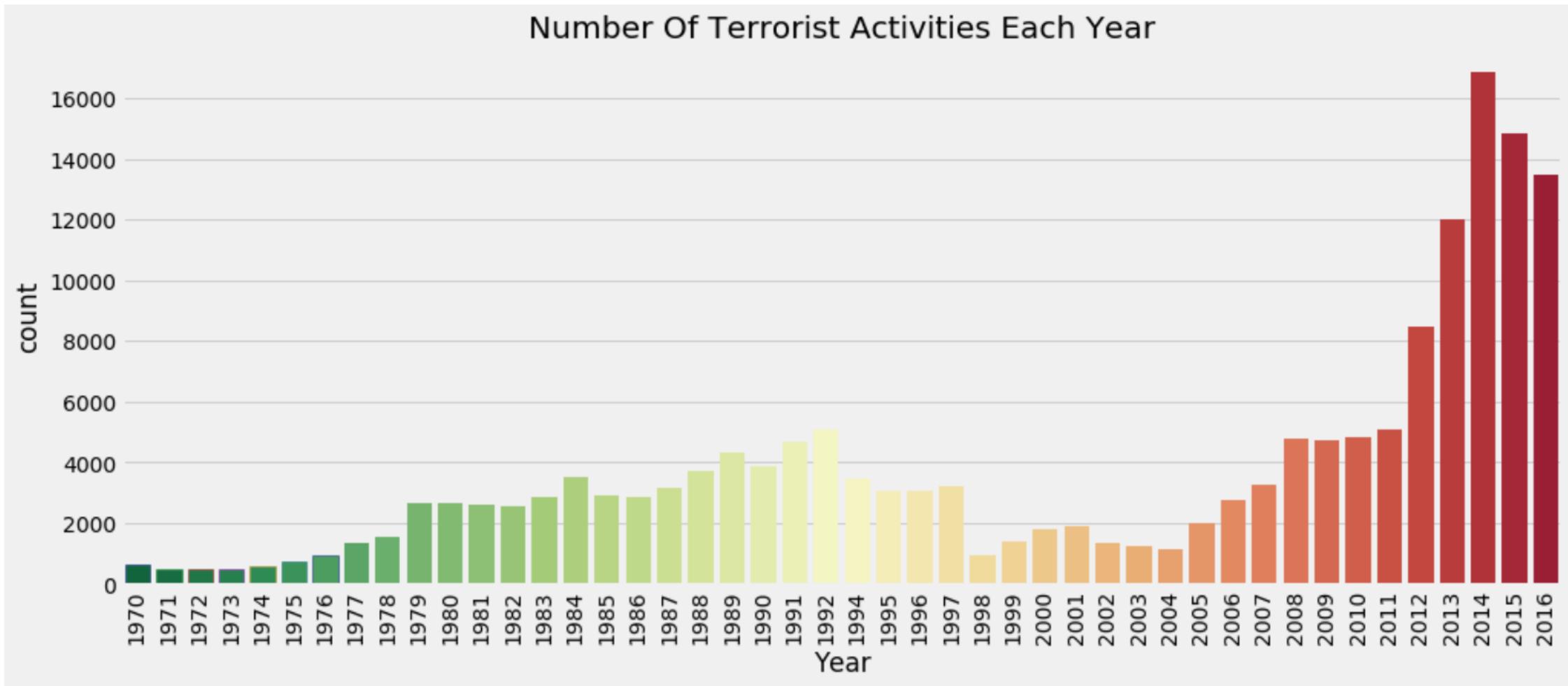
[terrorism](#)

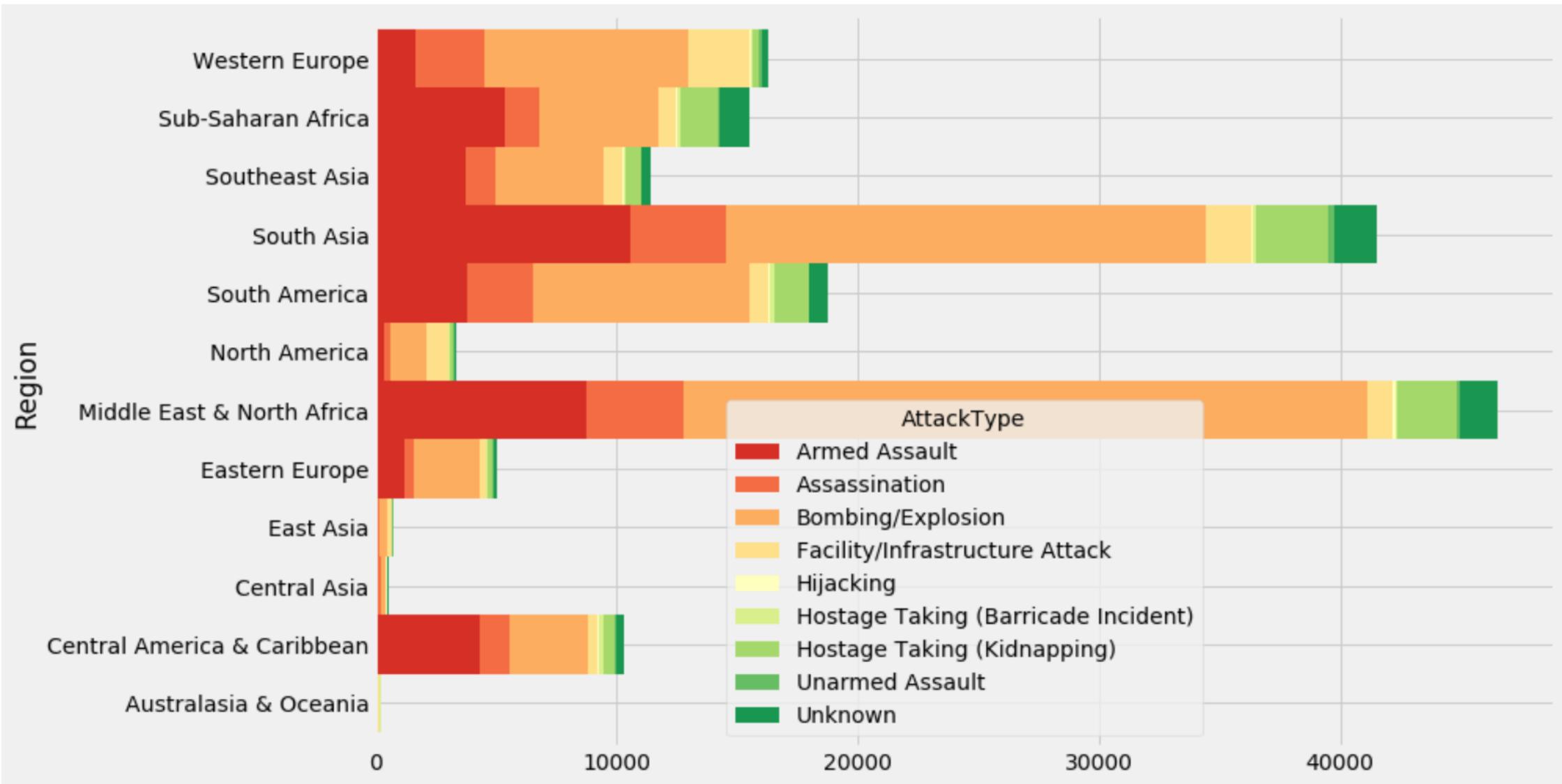
[animation](#)

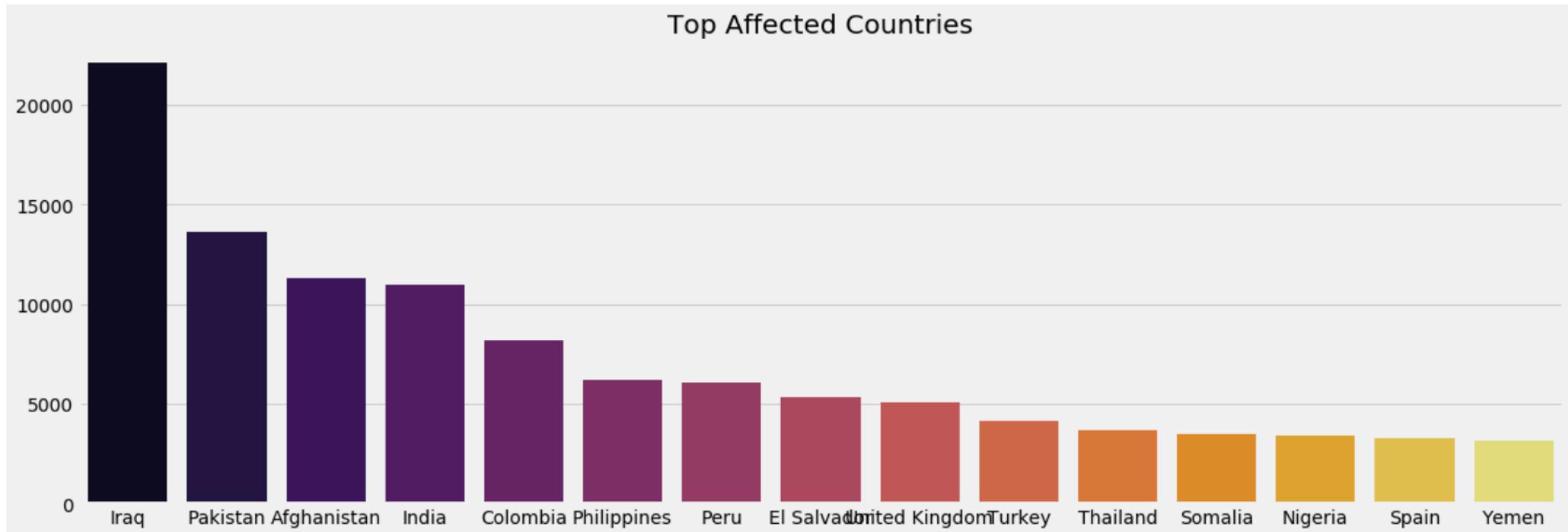
Notebook

<https://www.kaggle.com/ash316/terrorism-around-the-world/notebook>

	Year	Month	Day	Country	Region	city	latitude	longitude	AttackType	Killed	Wounded	Target
0	1970	7	2	Dominican Republic	Central America & Caribbean	Santo Domingo	18.456792	-69.951164	Assassination	1.0	0.0	Julio Guzman
1	1970	0	0	Mexico	North America	Mexico city	19.432608	-99.133207	Hostage Taking (Kidnapping)	0.0	0.0	Nadine Chaval, daughter
2	1970	1	0	Philippines	Southeast Asia	Unknown	15.478598	120.599741	Assassination	1.0	0.0	Employee







Animation Of Terrorist Activities Year:1970





어떤 상황에서도 무조건적인 평화를 지지합니다.

또 하나 ... 요즘 같은 시대 많은 글을 읽자 ...

2017년 12월 파이썬 세미나 - 나는 올해 얼마나 큰 꿈을 꾸었나

Dec 27, 2017

2017년 12월 파이썬 세미나 “나는 올해 얼마나 큰 꿈을 꾸었나”의 발표 자료를 공유합니다.

세션

이진석 - Azure function 활용한 파이썬 크롤링 스케줄링 슬라이드 [발표영상](#)

김은향 - 스타트업 인턴 개발자 3달간의 고군분투기 [슬라이드 발표영상](#)

이성민 - 작은 스타트업에서 머신러닝 맛보기 [슬라이드](#)

라이트닝 토크

김석훈 - which is better between FBV and CBV [슬라이드 발표영상](#)

이새로찬 - 개알못 개발자 생존기 [슬라이드 발표영상](#)

강명서 - “DjangoCon korea 라는 꿈을 꾸었다” [슬라이드 발표영상](#)

박민우 - “다시 개발자가 되는 꿈을 꾸었다” [슬라이드 발표영상](#)

강남 출근길에 정자/판교역 내릴 사람 예측하기

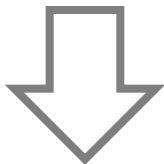
를 하기위한 탐색적 데이터 분석



<https://www.slideshare.net/ssuser2fe594/2107-80754131>

- 많은 경험이 필요한 **Data Sciece**
- 가장 빠르게 간접 경험을 얻을 수 있는 방법을 스스로 찾을 필요가 있음

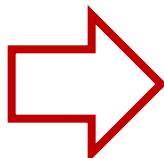
Data Scientist가 되자



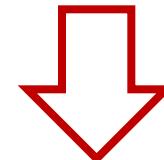
Python을 공부하자

???

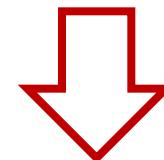
서울 주유소 가격 현황을 수집해서
실제 셀프 주유소가 저렴한지 확인하자



인터넷에서 정보를 어떻게 얻지?



얻은 정보를 어떻게 정리하지?



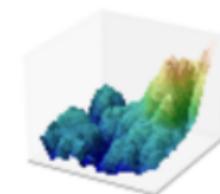
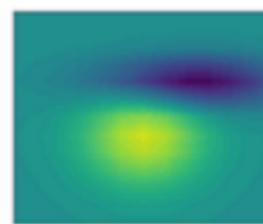
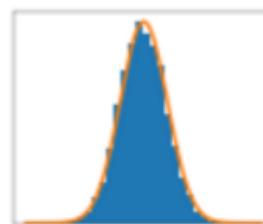
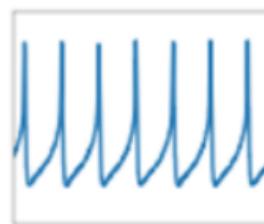
MATPLOTLIB의 기초



[home](#) | [examples](#) | [gallery](#) | [pyplot](#) | [docs](#) »

Introduction

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](#) shell, the [jupyter](#) notebook, web application servers, and four graphical user interface toolkits.



Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For a sampling, see the [screenshots](#), [thumbnail](#) gallery, and [examples](#) directory

For simple plotting the [pyplot](#) module provides a MATLAB-like interface, particularly when combined with [IPython](#). For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

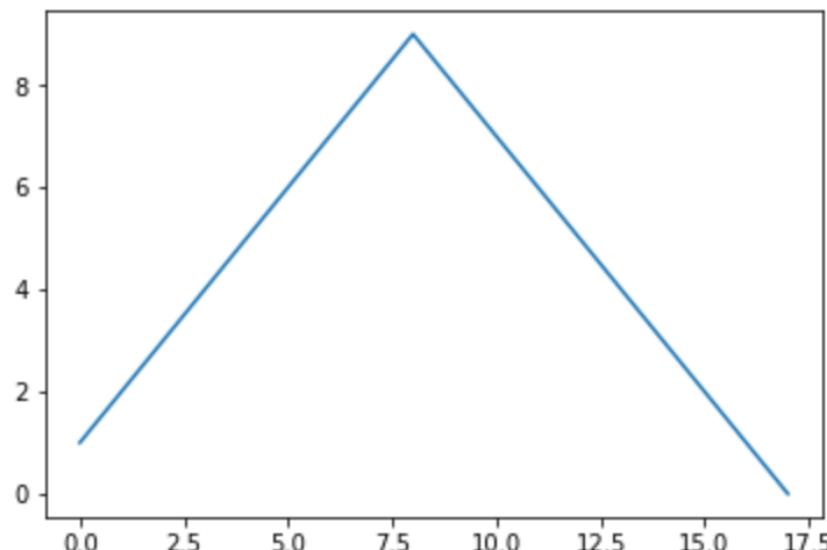
In [8]:

```
import matplotlib.pyplot as plt  
%matplotlib inline
```

- **matplotlib**에서 2D 그래프를 그리는 일반적 명령들은 **pyplot**에 있다.
- 대부분 **matplotlib.pyplot**을 **plt**로 이름짓는다
- 아웃 섹션에 그래프의 결과가 나오게 하기 위해
 - **%matplotlib inline**이라고 설정해둔다

In [41]:

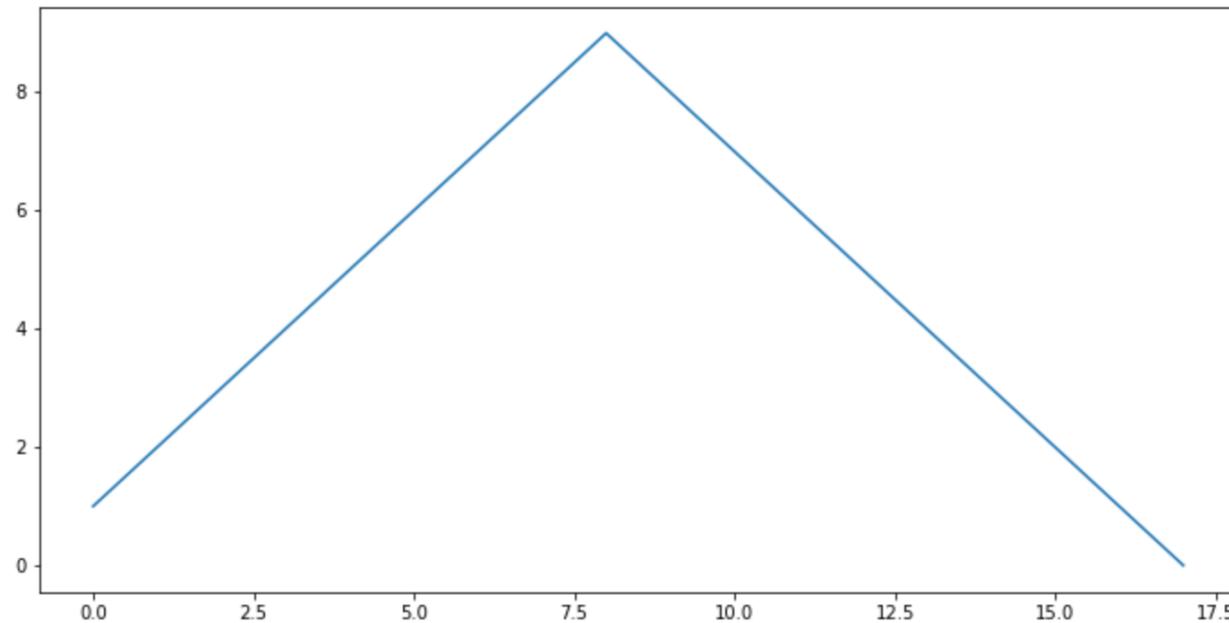
```
plt.figure()
plt.plot([1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1,0])
plt.show()
```



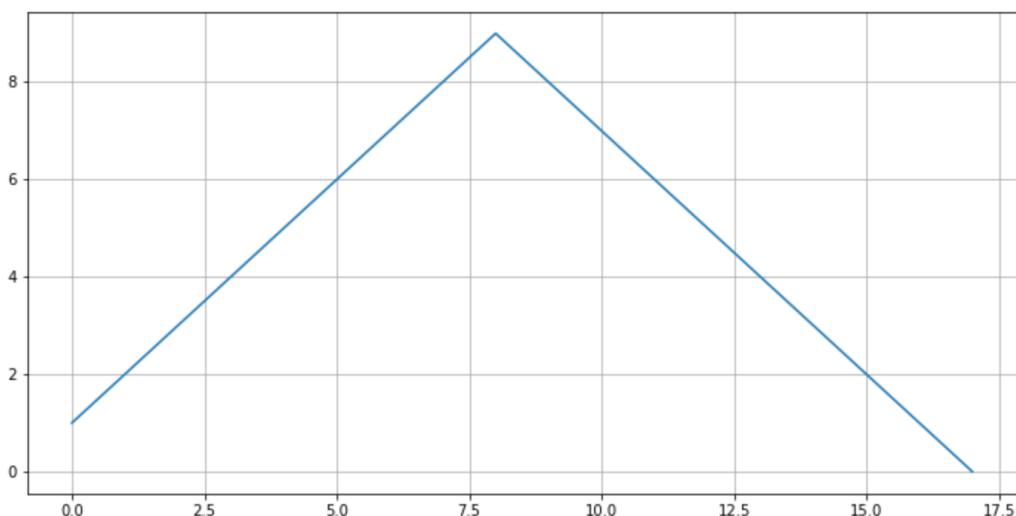
- 그냥 정말 그려보자.
- **figure()**로 시작해서 **show**로 닫는다
- **plot**은 값을 이어서 선으로 표현
- **plot**의 대상은 일반적으로 **list**형

Matplotlib 기초

```
In [11]: y = [1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1,0]  
  
plt.figure(figsize=(12,6))  
plt.plot(y)  
plt.show()
```



```
In [18]:  
y = [1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1,0]  
x = range(0,len(y))  
  
plt.figure(figsize=(12,6))  
plt.plot(x, y)  
plt.grid()  
plt.show()
```



- 보통 그래프는 x, y값을 인가
- `grid()` 명령은 격자 설정

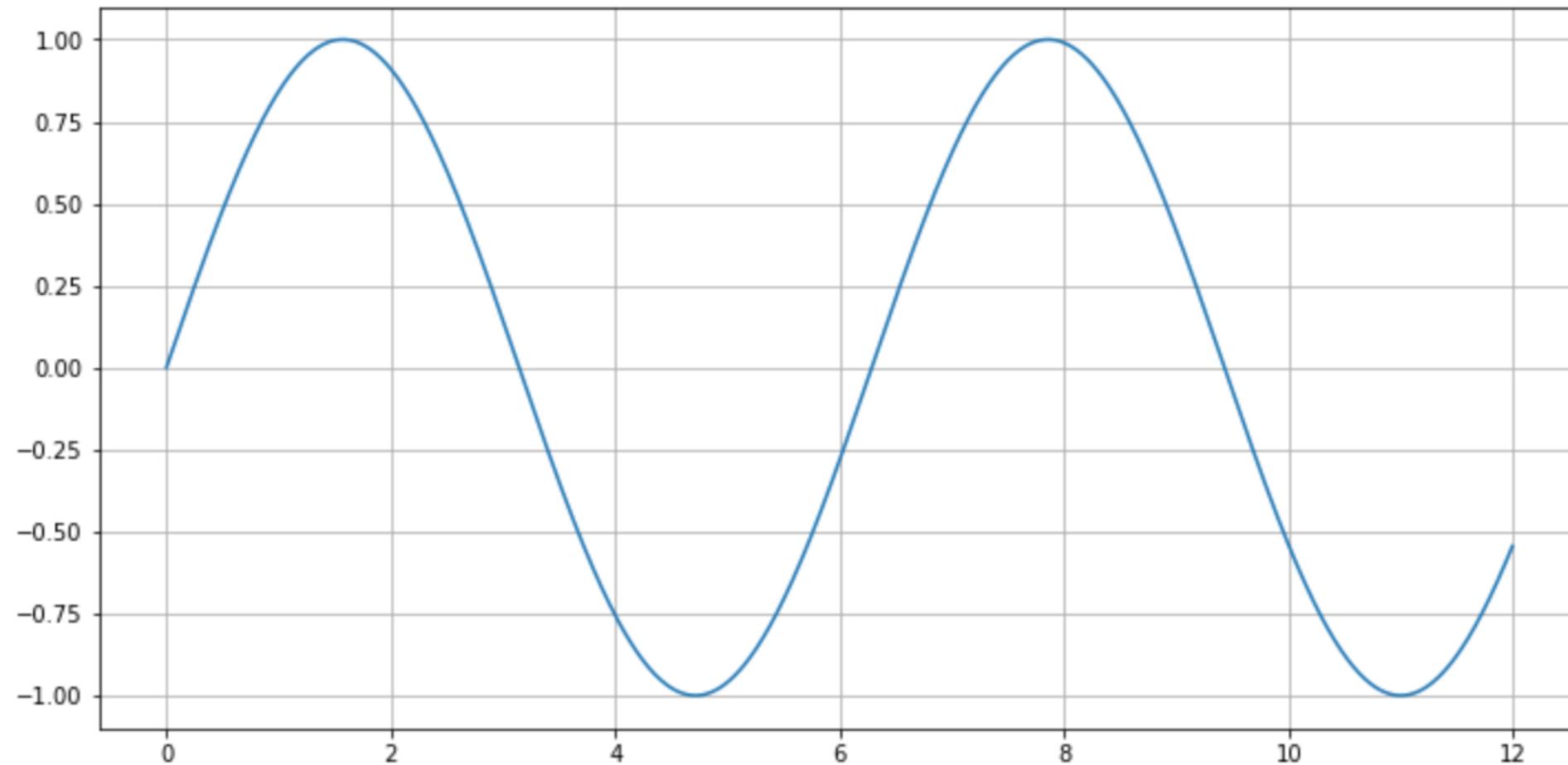
In [21]:

```
import numpy as np

t = np.arange(0,12,0.01)
y = np.sin(t)

plt.figure(figsize=(12,6))
plt.plot(t, y)
plt.grid()
plt.show()
```

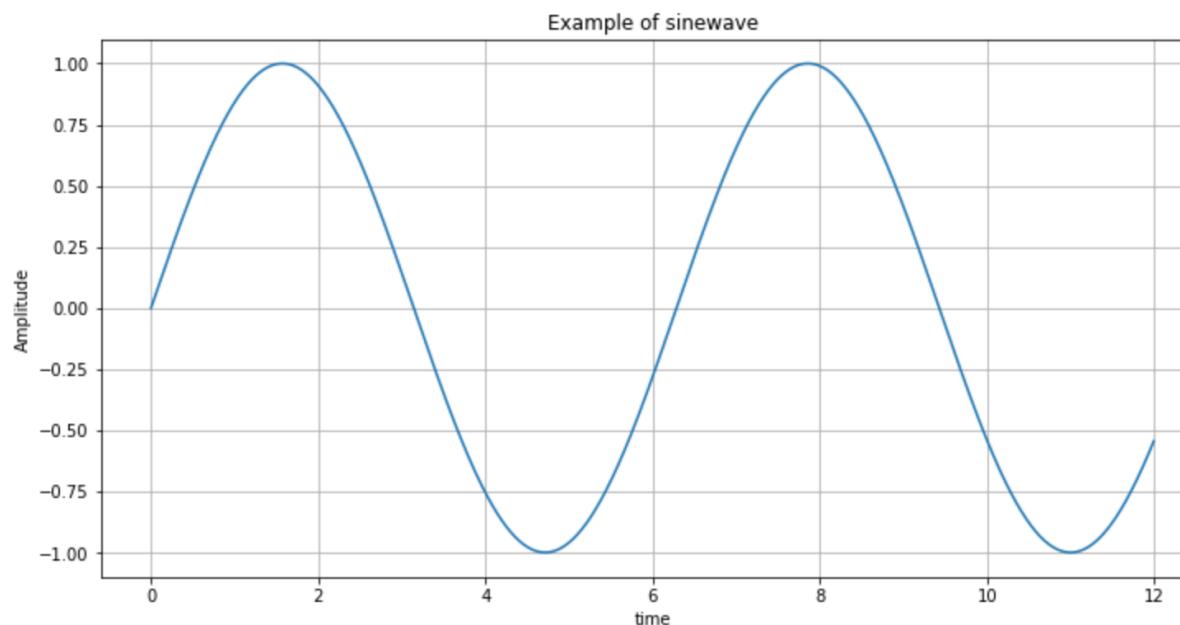
- **numpy**의 **arange** 명령 : 0부터 12까지 0.01 간격으로 t를 설정
- **np.sin(t)**로 한번에 y 값 생성
- **plot(t, y)**



Matplotlib 기초

In [23]:

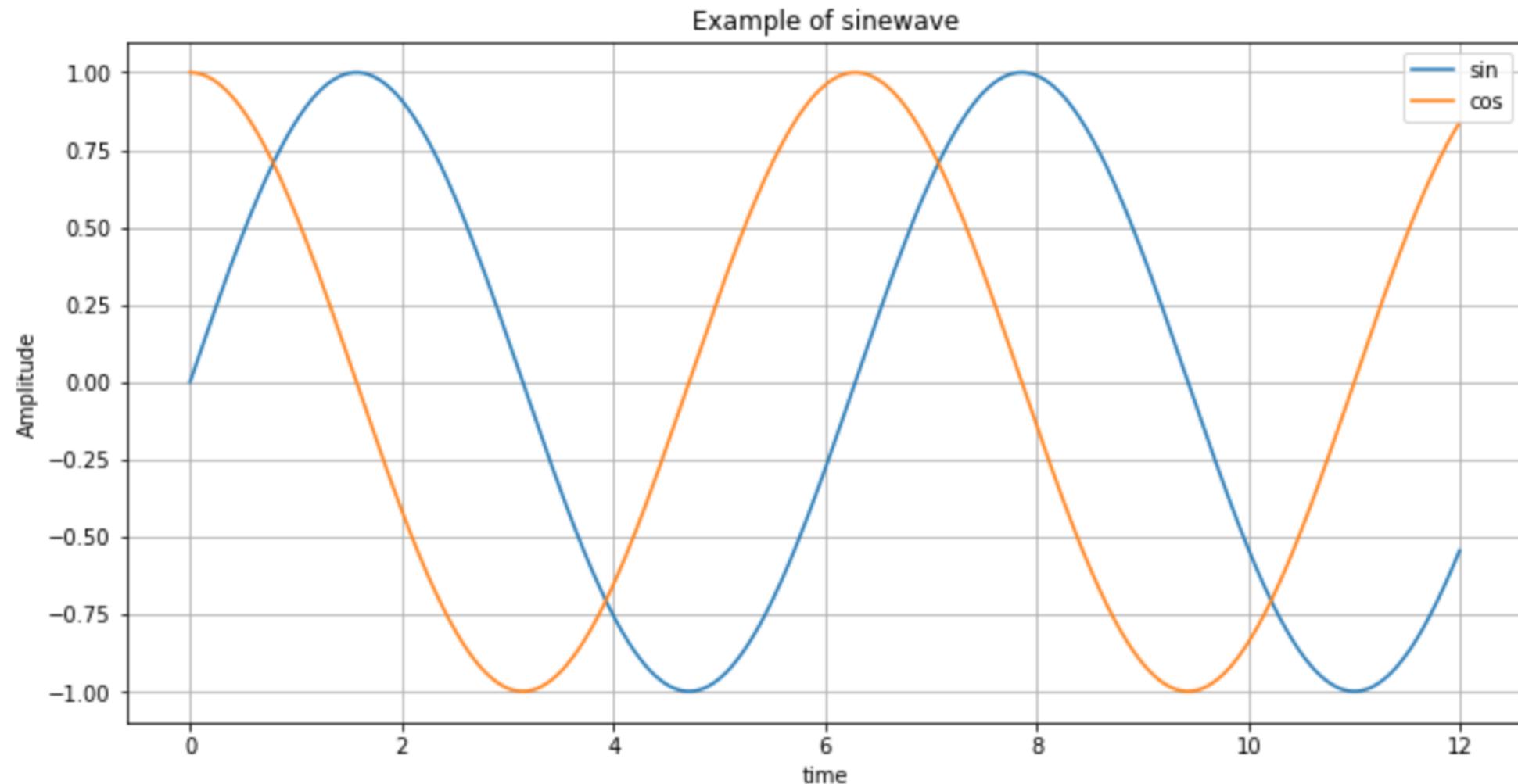
```
plt.figure(figsize=(12,6))  
plt.plot(t, y)  
plt.grid()  
plt.xlabel('time')          # x축 라벨 적용하기  
plt.ylabel('Amplitude')    # y축 라벨 적용하기  
plt.title('Example of sinewave') # 그래프의 타이틀 적용하기  
plt.show()
```



In [25]:

```
plt.figure(figsize=(12,6))
plt.plot(t, np.sin(t), label='sin')
plt.plot(t, np.cos(t), label='cos')
plt.grid()
plt.legend()
plt.xlabel('time')
plt.ylabel('Amplitude')
plt.title('Example of sinewave')
plt.show()
```

- **show()**로 닫기 전에, **plot**을 여러번 사용할 수 있다
- 한 그래프에 여러개의 선을 표현
- 구분을 위해 **label** 옵션을 사용하면
- **legend()** 명령으로 범례를 만들어 줄 수 있다.



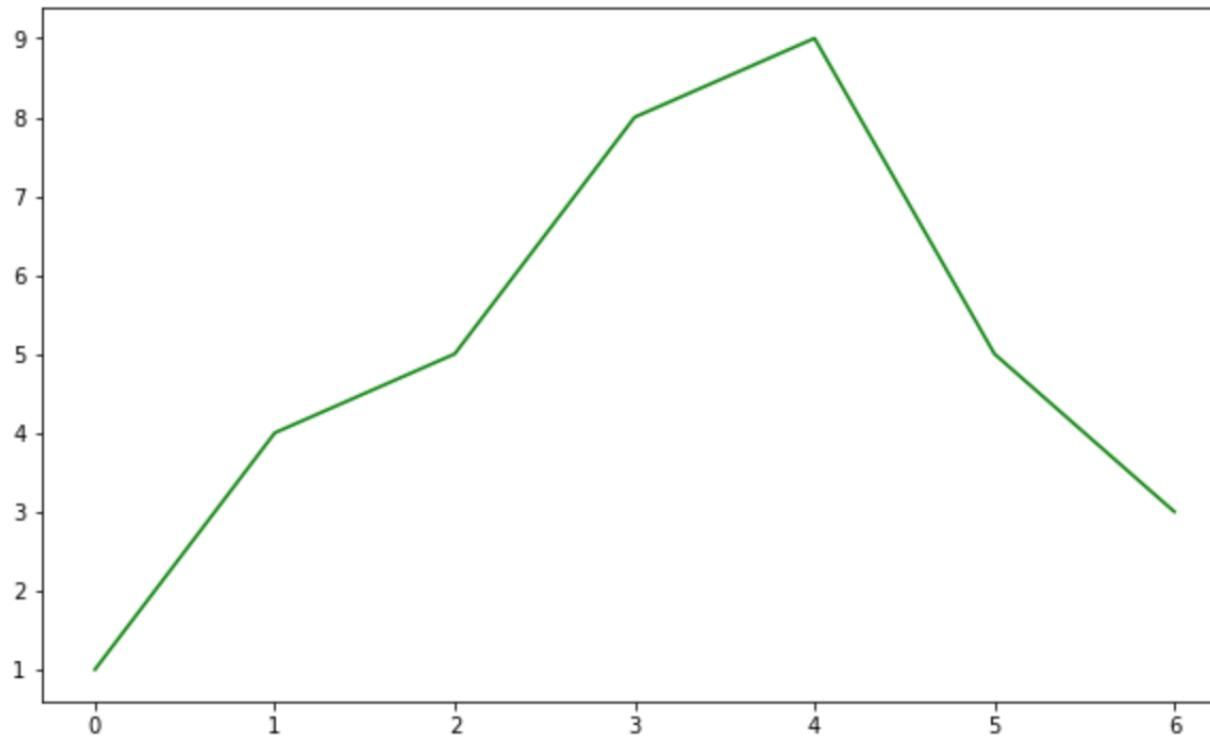
In [26]:

```
plt.figure(figsize=(12,6))
plt.plot(t, np.sin(t), lw=3, label='sin')
plt.plot(t, np.cos(t), 'r', label='cos')
plt.grid()
plt.legend()
plt.xlabel('time')
plt.ylabel('Amplitude')
plt.title('Example of sinewave')
plt.show()
```

- 그 외에도 **lw (line width)** 선 두께를 지정할 수 있다.
- 또, 선 색을 지정할 수 있다.
 - one of {'b', 'g', 'r', 'c', 'm', 'y', 'k', 'w'}

Matplotlib 기초

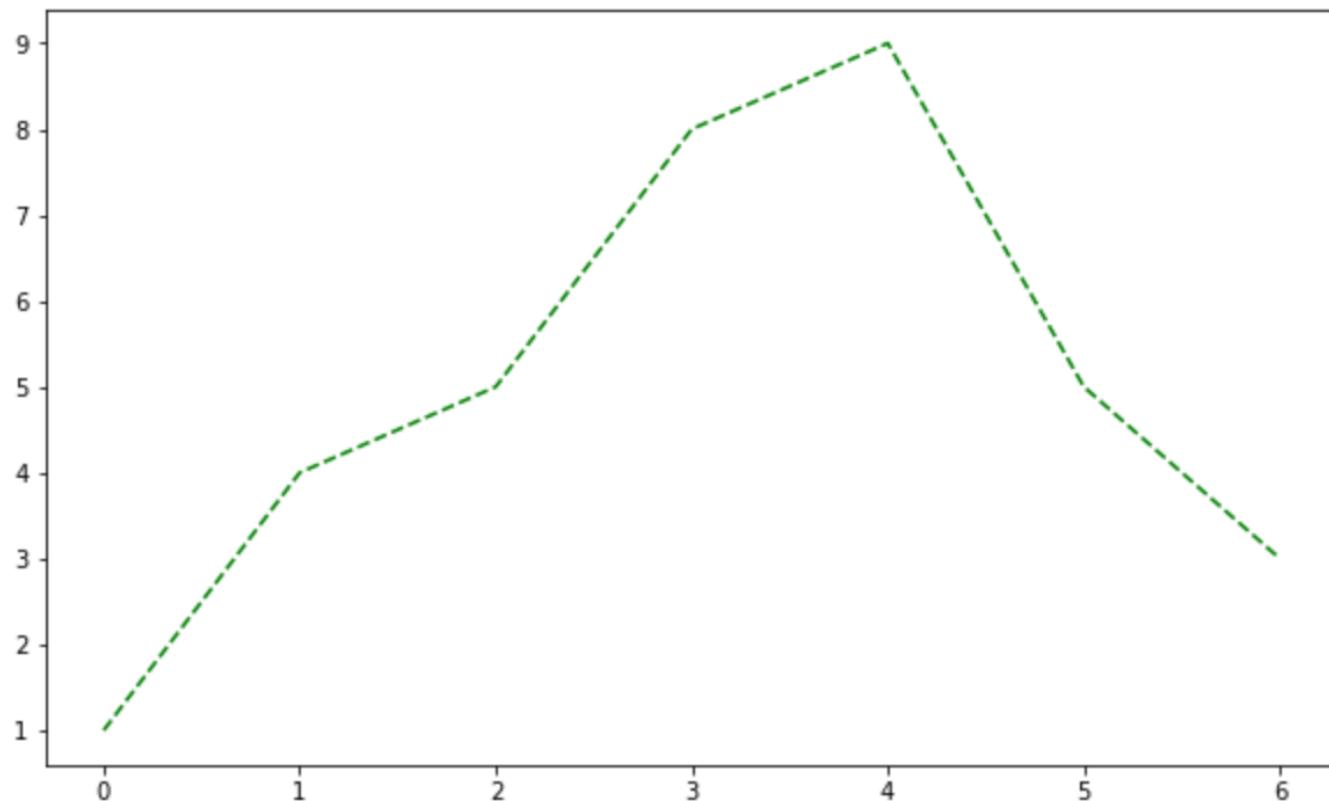
```
In [27]:  
t = [0, 1, 2, 3, 4, 5, 6]  
y = [1, 4, 5, 8, 9, 5, 3]  
  
plt.figure(figsize=(10,6))  
plt.plot(t, y, color='green')  
plt.show( )
```



Matplotlib 기초

In [28]:

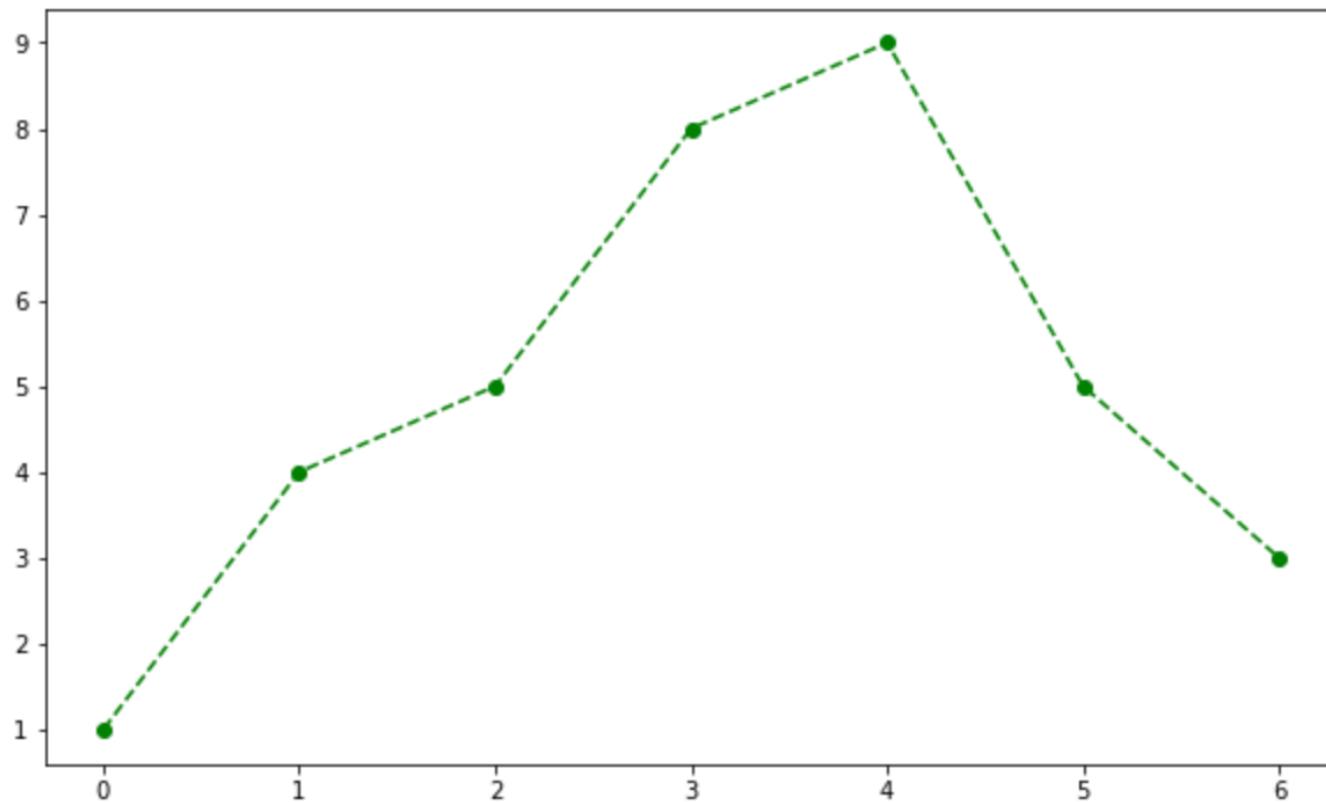
```
plt.figure(figsize=(10,6))  
plt.plot(t, y, color='green', linestyle='dashed')  
plt.show()
```



Matplotlib 기초

In [29]:

```
plt.figure(figsize=(10,6))  
plt.plot(t, y, color='green', linestyle='dashed', marker='o')  
plt.show()
```

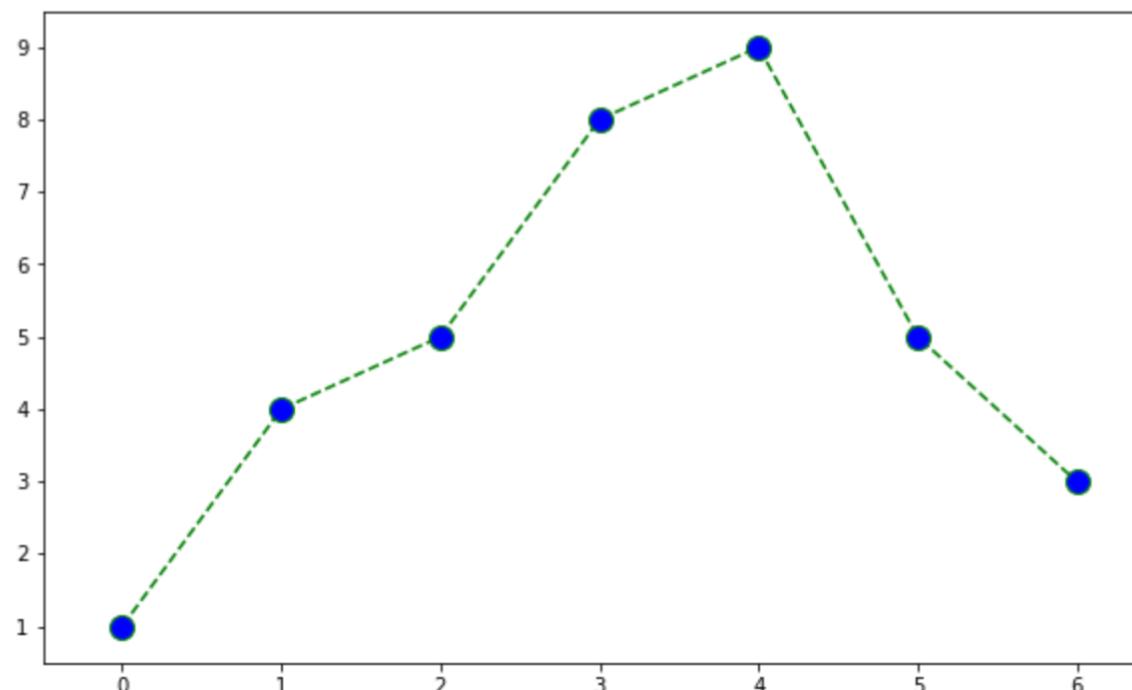


Matplotlib 기초

In [30]:

```
plt.figure(figsize=(10,6))
plt.plot(t, y, color='green', linestyle='dashed', marker='o',
          markerfacecolor = 'blue', markersize=12)

plt.xlim([-0.5, 6.5])
plt.ylim([0.5, 9.5])
plt.show()
```

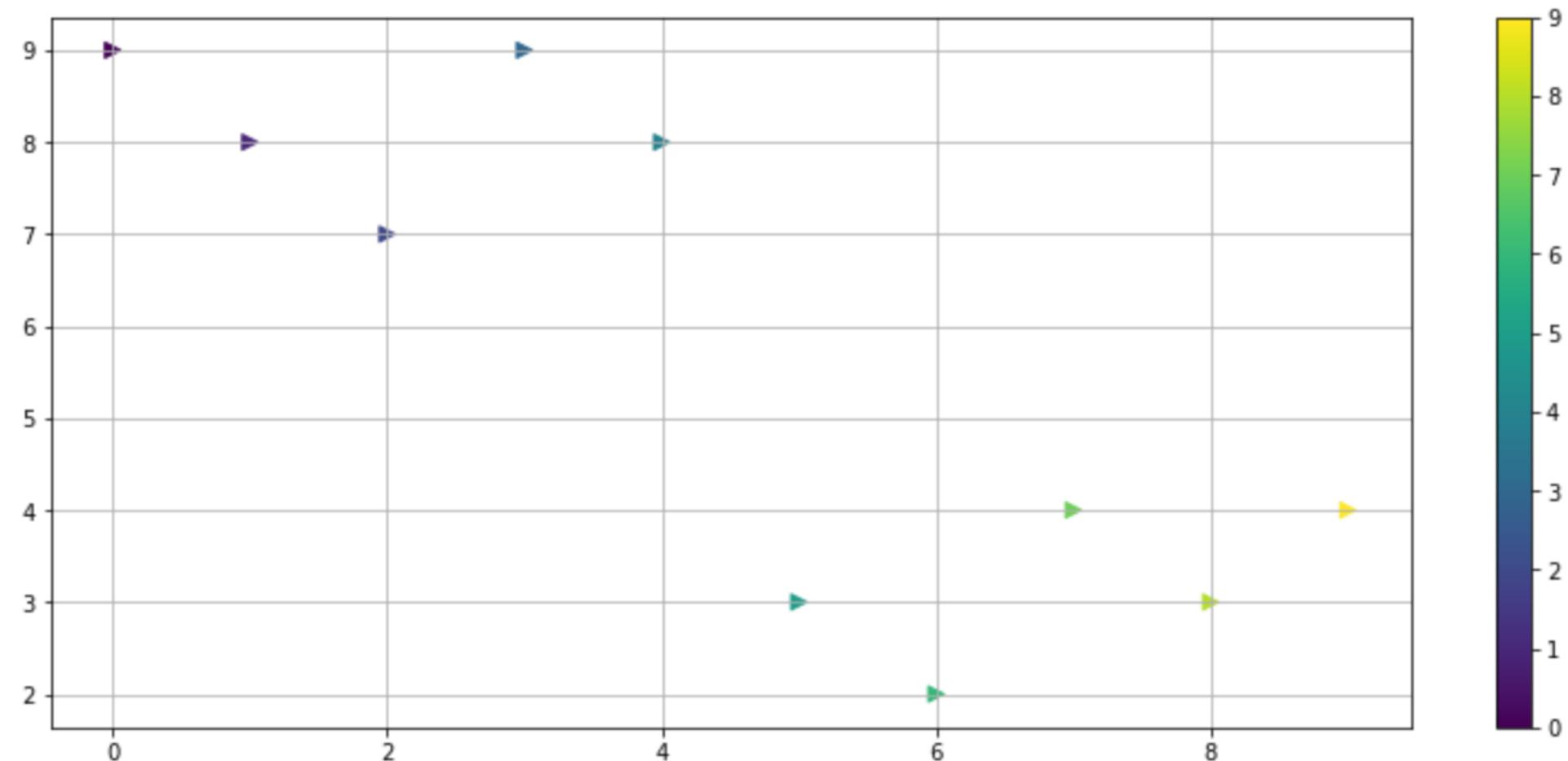


In [53]:

```
t = np.array([0,1,2,3,4,5,6,7,8,9])
y = np.array([9,8,7,9,8,3,2,4,3,4])

plt.figure(figsize=(14,6))
plt.scatter(t,y, s = 50, c = t, marker='>')
plt.colorbar()
plt.grid()
plt.show()
```

- **scatter** : 산점도
- **c** : colormap, colorbar()와 함께 사용
- **marker** : ('o', 'v', '^', '<', '>', 's', 'p', '*', 'h', 'H', 'D', 'd', 'P', 'X')

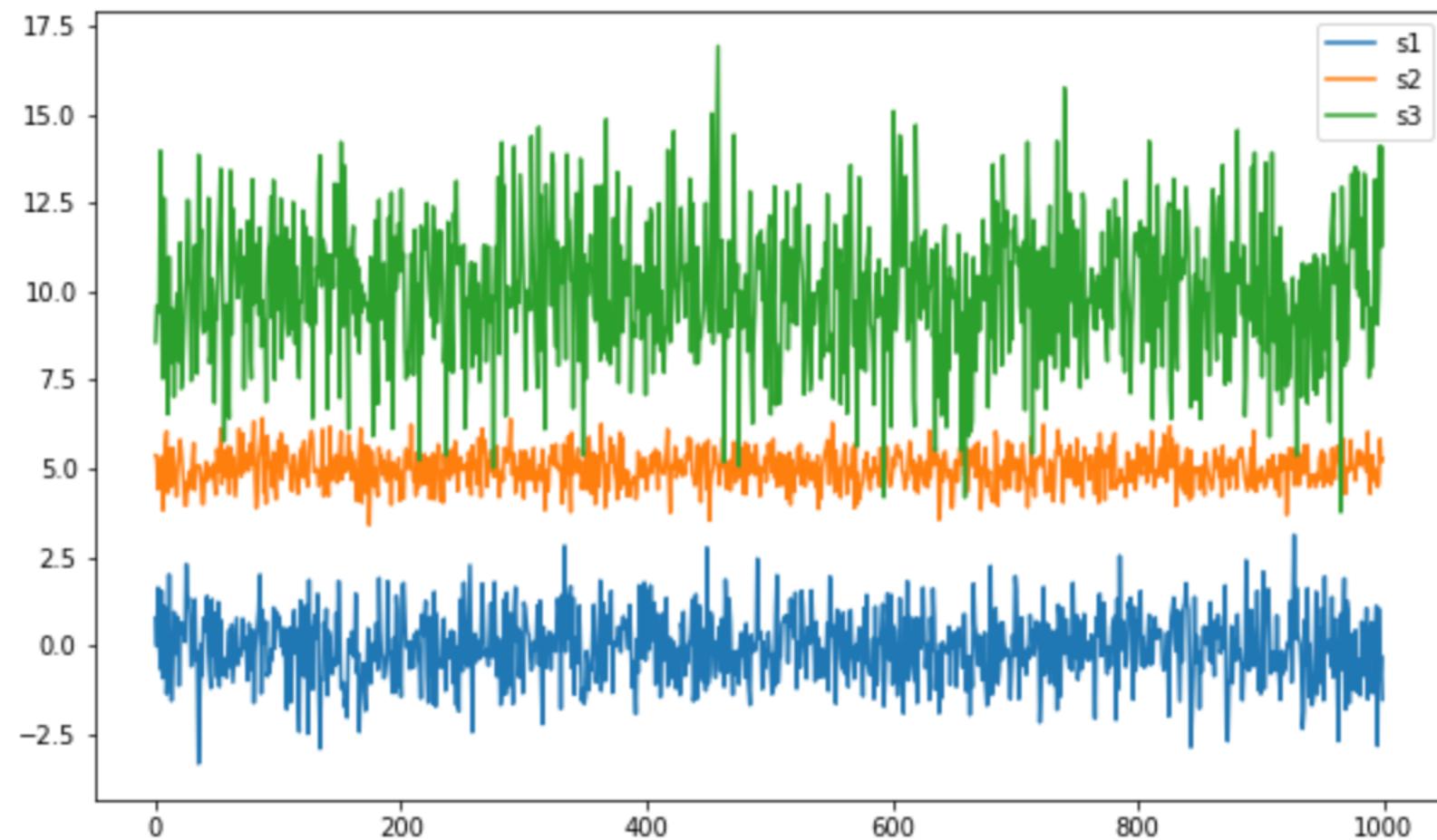


In [38]:

```
s1 = np.random.normal(loc=0, scale=1, size=1000)
s2 = np.random.normal(loc=5, scale=0.5, size=1000)
s3 = np.random.normal(loc=10, scale=2, size=1000)

plt.figure(figsize=(10,6))
plt.plot(s1, label='s1')
plt.plot(s2, label='s2')
plt.plot(s3, label='s3')
plt.legend(loc=1)
plt.show()
```

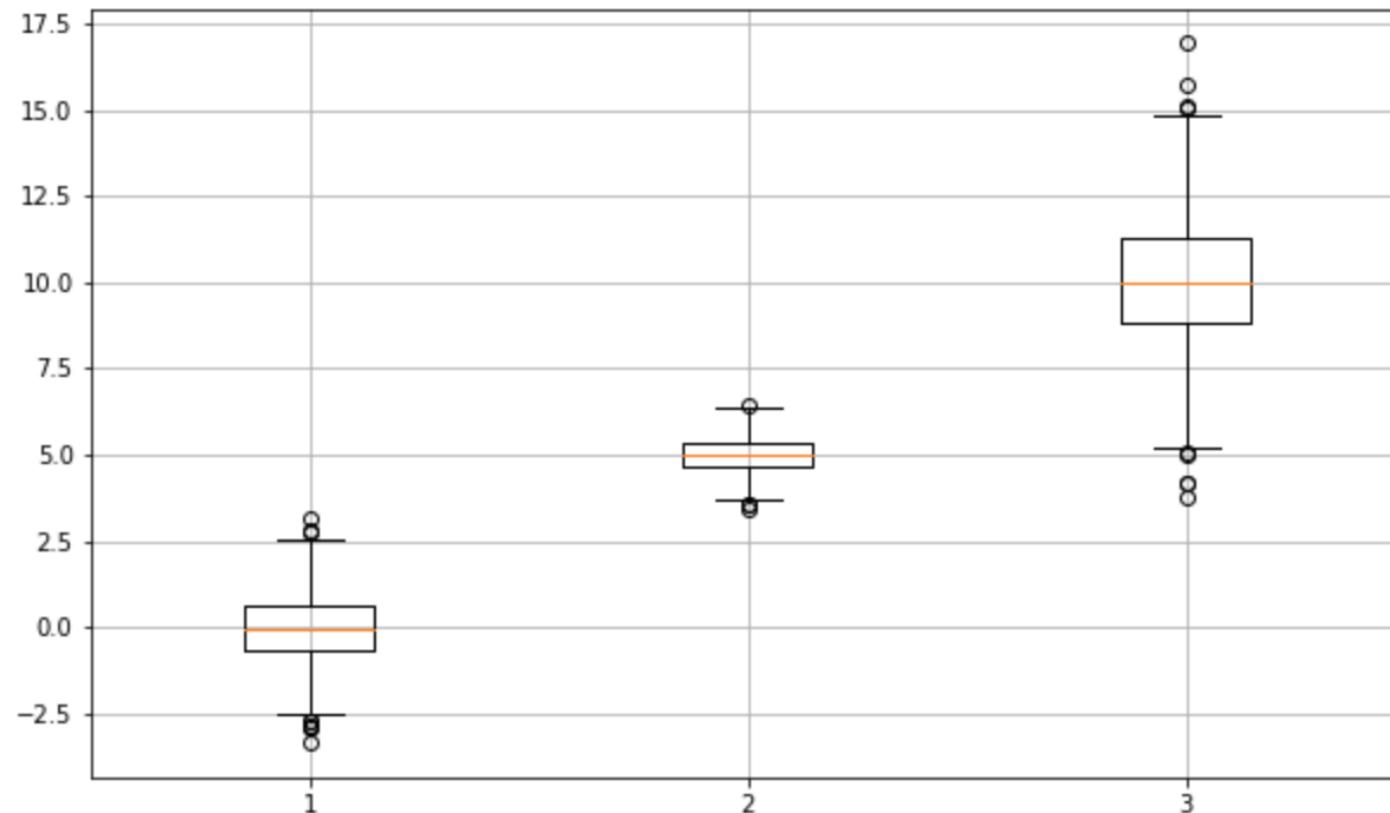
- 평균이 0, 5, 10이고 표준편차가 각각 1, 0.5, 2인 랜덤 변수 생성



Matplotlib 기초

In [39]:

```
plt.figure(figsize=(10,6))  
plt.boxplot((s1, s2, s3))  
plt.grid()  
plt.show()
```

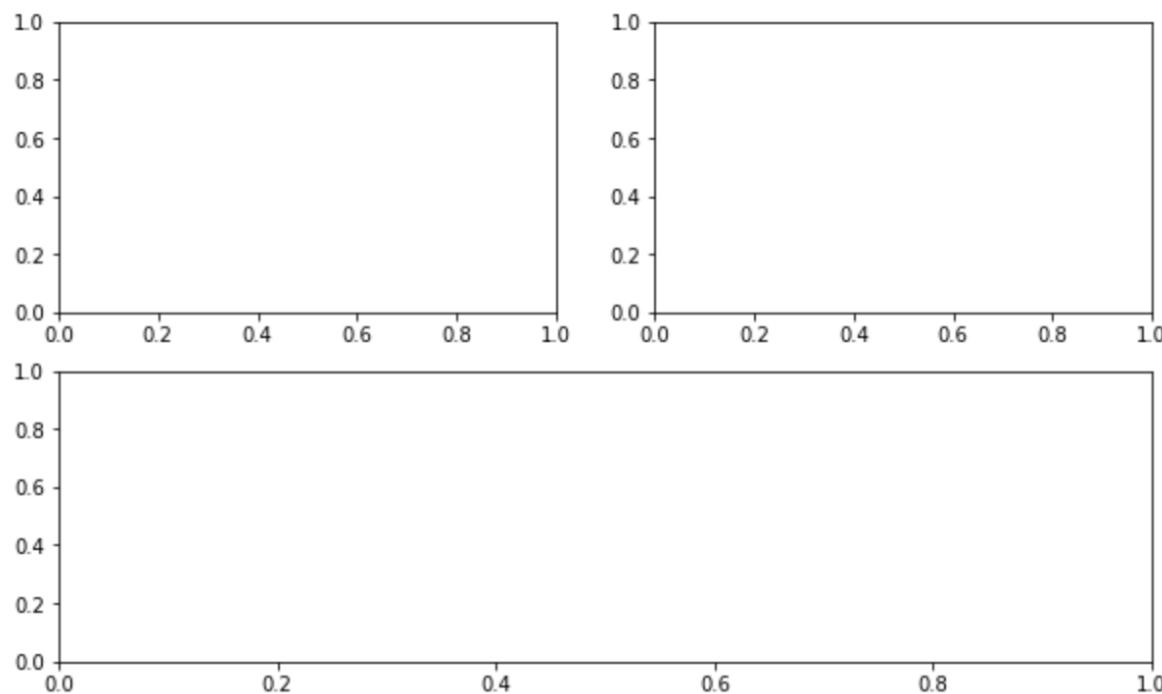


Matplotlib 기초

```
In [40]: plt.figure(figsize=(10,6))

plt.subplot(221)
plt.subplot(222)
plt.subplot(212)

plt.show()
```



- 네이버, 페이스북, 구글 등 IT의 거대 기업들로부터 많은 S/W 관련 회사들이
- 자사 제품을 사용하는 것에 대해 API를 제공한다
- 특히 요즘처럼 웹 크롤링에 대한 방어가 점점 심해지는 이때
 해당 회사의 API를 사용하는 것이 오히려 더 바람직 할 수 있다
- 점점 API의 사용에 익숙해 질 필요 또한 분명하다

함수의 기초

```
In [1]: def test_def(a, b):  
    return a + b
```

```
In [2]: test_def(10,5)
```

15

- 어떤 언어든 함수의 개념은 정말 간단하다
- 입력 인자를 받고
- 내부 연산을 하고
- 결과값을 **return** 한다 (혹은 리턴하지 않을 수도 있다)

```
In [18]: def test_args(a, *arg):  
    print("첫 번째 인자:", a)  
    for each in arg:  
        print ("*arg 다른 인자", each)
```

```
In [19]: test_args('a','b')
```

첫 번째 인자: a
*arg 다른 인자 b

- 혹은 입력 인자의 수를 제한하지 않고 받을 수 있다
- **arg**라는 이름이 아니라 *표가 하나 있다는 것이 중요^^

```
In [21]: def test_kwargs(**kwargs):  
    print(kwargs)
```

```
In [23]: test_kwargs(A='aa', B='bb')  
  
{'A': 'aa', 'B': 'bb'}
```

- 별표 두개의 의미는 **key-worded argument**라는 뜻으로
- 흔히 **A=1, B=2** 와 같이 입력하는 것을 의미한다
- 내부적으로는 **dict** 형으로 처리한다

```
In [24]: def test_kwargs(**kwargs):
    if kwargs is not None:
        for key, value in kwargs.items():
            print(key, ' : ', value)
```

```
In [25]: test_kwargs(A='aa', B='bb')
```

```
A : aa
B : bb
```

- dic형이니 이렇게 내부적으로 처리해 볼 수 있다

```
In [30]: def test_kwargs(**kwargs):
    A = kwargs.get('A', 1)
    print(A)
```

```
In [33]: test_kwargs()
```

```
1
```

```
In [34]: test_kwargs(A=10)
```

```
10
```

- ****kwargs**는 **get** 함수를 사용해서 – 당연히 **dict** – 입력되지 않은 변수의 값을 정의할 수 있다

In [3]:

```
import matplotlib.pyplot as plt  
import numpy as np  
%matplotlib inline
```

- 그럼 재미있는거 하나 시작할까

$$y = a \sin(2\pi ft + t_0) + b$$

- 이 함수의 그래프를 그려주는 함수를 만들자...

In [9]:

```
def plotSinWave(amp, freq, endTime, sampleTime, bias):
    """
    plot sine wave
    y = a sin(2 pi f t) + b
    """
    time = np.arange(0, endTime, sampleTime)
    result = amp * np.sin(2*np.pi*freq*time) + bias

    plt.figure(figsize=(12,6))
    plt.plot(time, result); plt.grid(True)
    plt.xlabel('time'); plt.ylabel('sin')
    plt.title(str(amp) + '*sin(2*pi*' + str(freq) + '*t )+' + str(bias))
    plt.show()
```

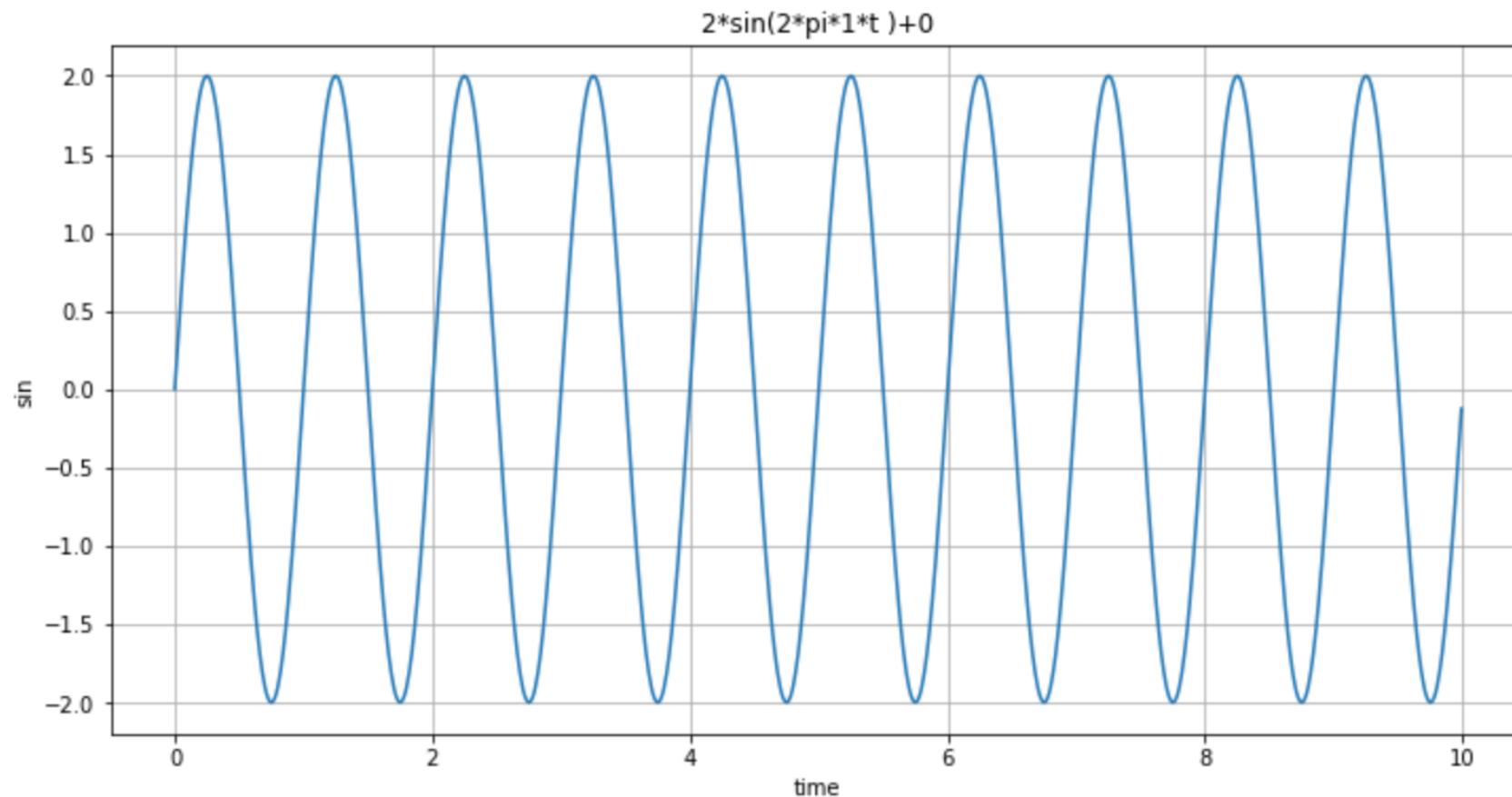
In [9]:

```
def plotSinWave(amp, freq, endTime, sampleTime, bias):
    """
    plot sine wave
    y = a sin(2 pi f t) + b
    """
    time = np.arange(0, endTime, sampleTime)
    result = amp * np.sin(2*np.pi*freq*time) + bias

    plt.figure(figsize=(12,6))
    plt.plot(time, result); plt.grid(True)
    plt.xlabel('time'); plt.ylabel('sin')
    plt.title(str(amp) + '*sin(2*pi*' + str(freq) + '*t )+' + str(bias))
    plt.show()
```

In [10]:

```
plotSinWave(2, 1, 10, 0.01, 0)
```



In [38]:

```
def plotSinWave(**kwargs):
    """
    plot sine wave
    y = a sin(2 pi f t) + b
    """

    endTime, sampleTime = kwargs.get('endTime', 1), kwargs.get('sampleTime', 0.01)
    amp, freq, bias = kwargs.get('amp', 1), kwargs.get('freq', 1), kwargs.get('bias', 0)
    figsize = kwargs.get('figsize', (12,6))

    time = np.arange(0, endTime, sampleTime)
    result = amp * np.sin(2*np.pi*freq*time) + bias

    plt.figure(figsize=(12,6))
    plt.plot(time, result); plt.grid(True)
    plt.xlabel('time'); plt.ylabel('sin')
    plt.title(str(amp)+'*sin(2*pi*'+str(freq)+'*t )+'+str(bias))
    plt.show()
```

In [40]:

plotSinWave?

```
Signature: plotSinWave(**kwargs)
Docstring:
plot sine wave
y = a sin(2 pi f t) + b
File:      ~/Documents/FastCampus 수업자료/FC 08/source_code/<ipython-input-38-91c4d4fce032>
Type:      function
```

- ?표를 달고 실행하면 화면 하단에 설명이 나온다

```
    plt.title(str(amp) + 'sin(' + str(freq) + 't) + ' + str(phase))  
plt.show()
```

In [40]: plotSinWave

In [39]: Signature: plotSinWave(**kwargs)

Docstring:

plot sine wave

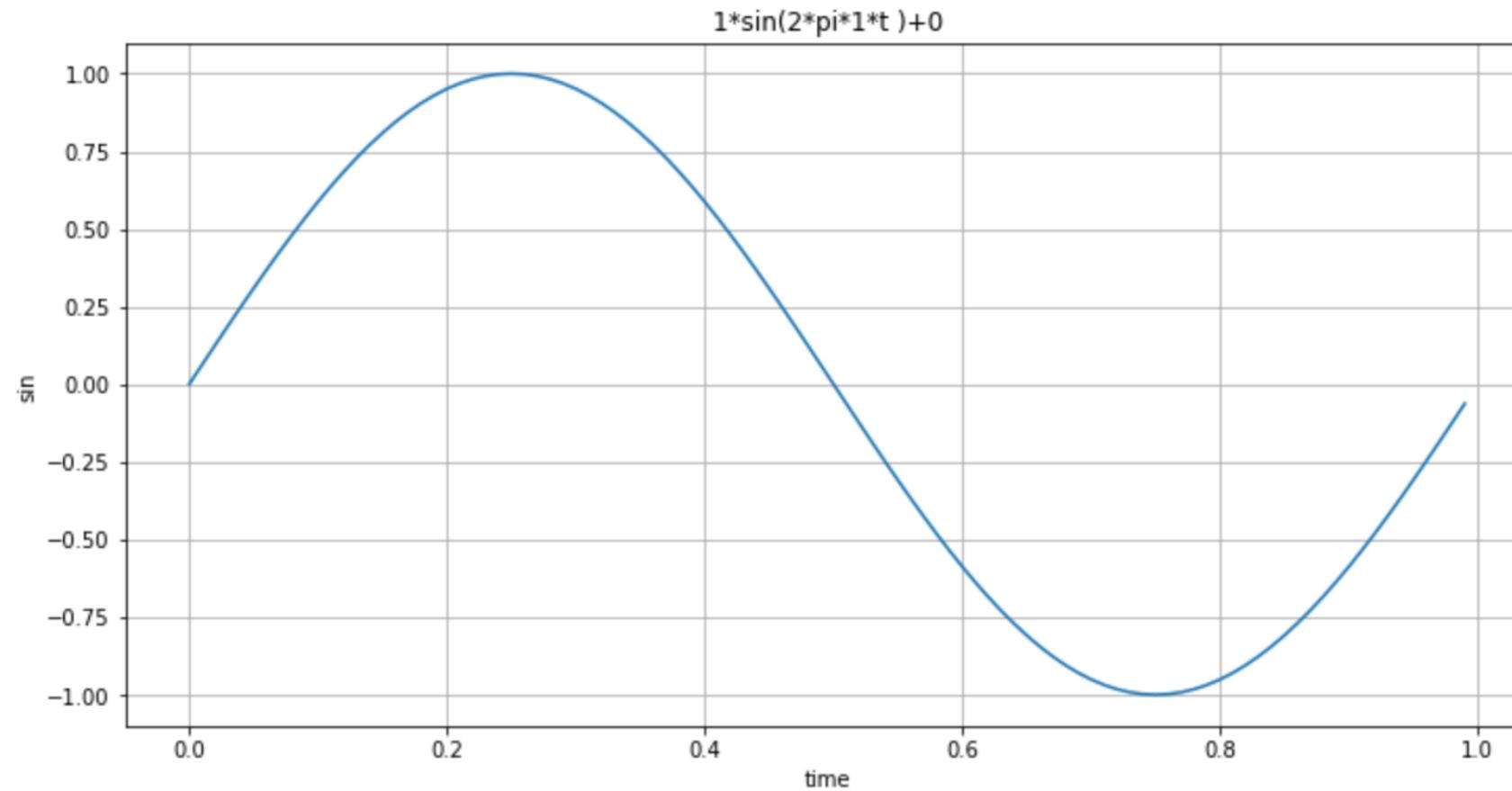
y = a sin(2 pi f t) + b

File: ~/Documents/FastCampus 수업자료/FC 08/source_code/<ipython-input-3
8-91c4d4fce032>

Type: function

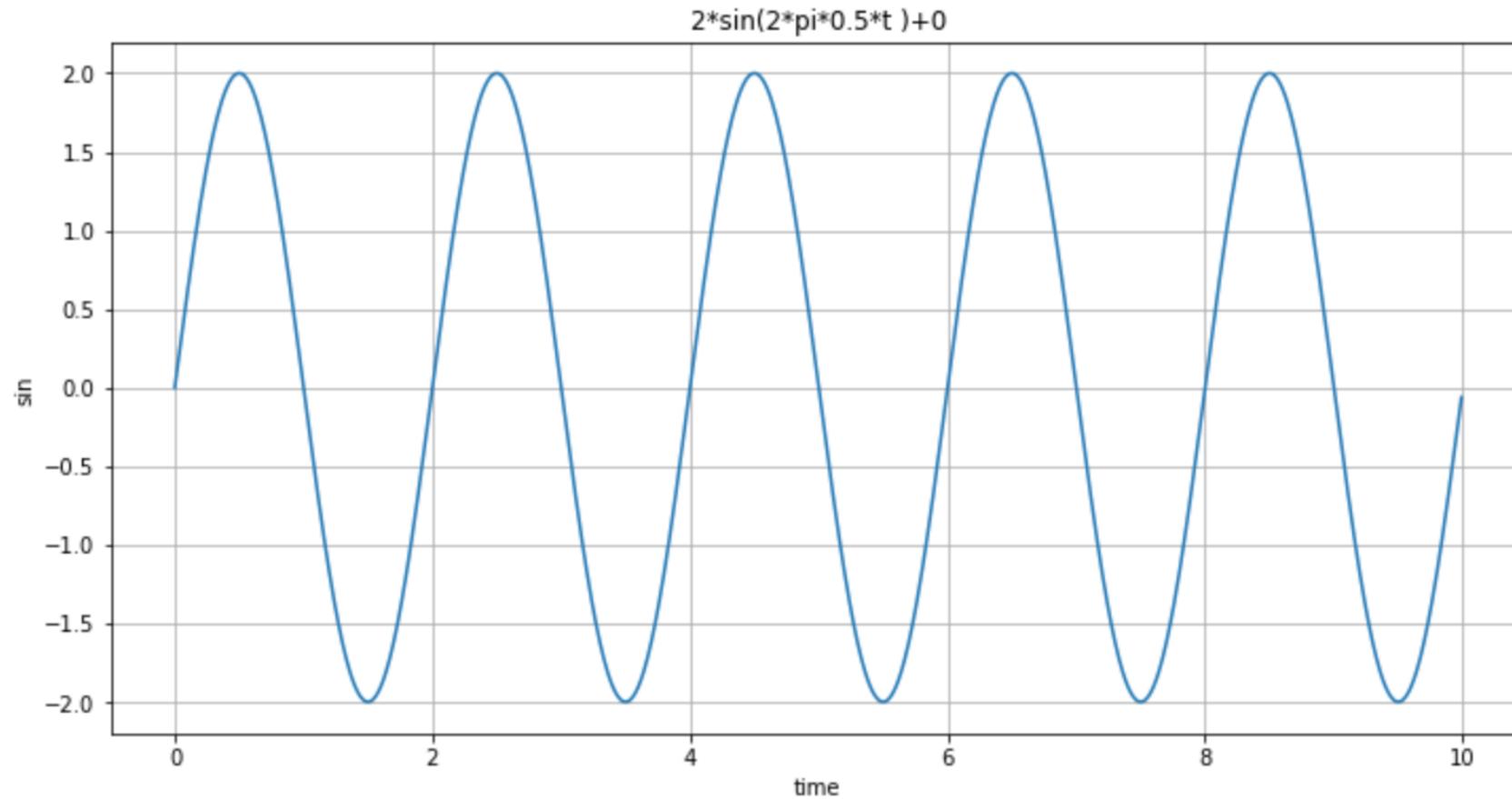
- Shift + Tab을 해도 된다

```
In [39]: plotSinWave( )
```

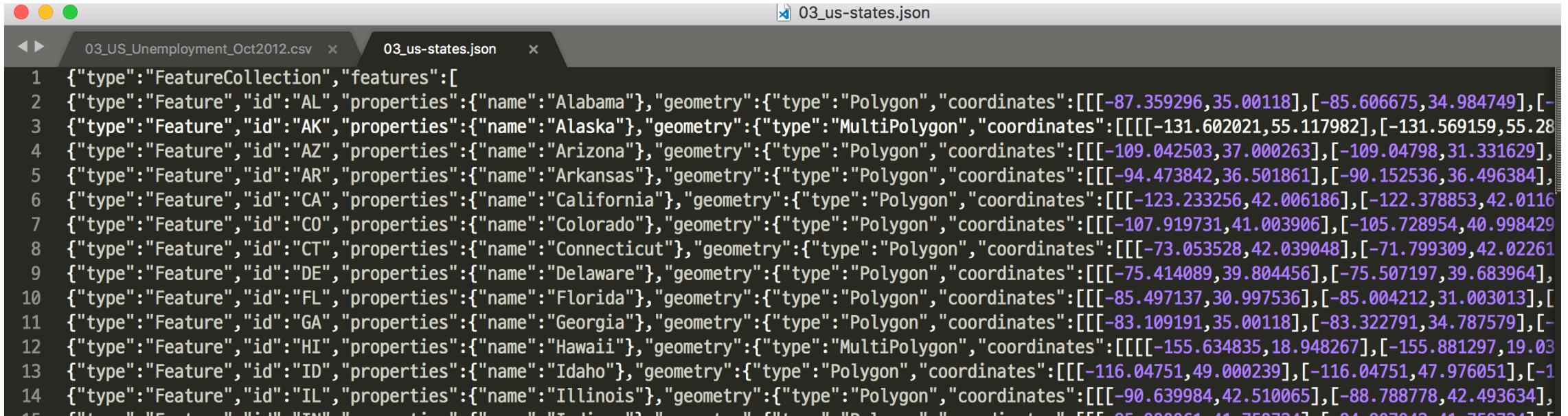


In [37]:

```
plotSinWave(amp=2, freq=0.5, endTime=10)
```



안하고 가진 좀 짐찜한 - JSON 기초



```
03_us-states.json
1 {"type": "FeatureCollection", "features": [
2 {"type": "Feature", "id": "AL", "properties": {"name": "Alabama"}, "geometry": {"type": "Polygon", "coordinates": [[[[-87.359296, 35.00118], [-85.606675, 34.984749], [-85.506675, 34.984749], [-85.506675, 35.00118], [-87.359296, 35.00118]]]}},
3 {"type": "Feature", "id": "AK", "properties": {"name": "Alaska"}, "geometry": {"type": "MultiPolygon", "coordinates": [[[[-131.602021, 55.117982], [-131.569159, 55.28117982], [-131.569159, 55.28117982], [-131.602021, 55.117982]]]}},
4 {"type": "Feature", "id": "AZ", "properties": {"name": "Arizona"}, "geometry": {"type": "Polygon", "coordinates": [[[[-109.042503, 37.000263], [-109.04798, 31.331629], [-109.04798, 31.331629], [-109.042503, 37.000263]]]}},
5 {"type": "Feature", "id": "AR", "properties": {"name": "Arkansas"}, "geometry": {"type": "Polygon", "coordinates": [[[[-94.473842, 36.501861], [-90.152536, 36.496384], [-90.152536, 36.496384], [-94.473842, 36.501861]]]}},
6 {"type": "Feature", "id": "CA", "properties": {"name": "California"}, "geometry": {"type": "Polygon", "coordinates": [[[[-123.233256, 42.006186], [-122.378853, 42.011618], [-122.378853, 42.011618], [-123.233256, 42.006186]]]}},
7 {"type": "Feature", "id": "CO", "properties": {"name": "Colorado"}, "geometry": {"type": "Polygon", "coordinates": [[[[-107.919731, 41.003906], [-105.728954, 40.998429], [-105.728954, 40.998429], [-107.919731, 41.003906]]]}},
8 {"type": "Feature", "id": "CT", "properties": {"name": "Connecticut"}, "geometry": {"type": "Polygon", "coordinates": [[[[-73.053528, 42.039048], [-71.799309, 42.022611], [-71.799309, 42.022611], [-73.053528, 42.039048]]]}},
9 {"type": "Feature", "id": "DE", "properties": {"name": "Delaware"}, "geometry": {"type": "Polygon", "coordinates": [[[[-75.414089, 39.804456], [-75.507197, 39.683964], [-75.507197, 39.683964], [-75.414089, 39.804456]]]}},
10 {"type": "Feature", "id": "FL", "properties": {"name": "Florida"}, "geometry": {"type": "Polygon", "coordinates": [[[[-85.497137, 30.997536], [-85.004212, 31.003013], [-85.004212, 31.003013], [-85.497137, 30.997536]]]}},
11 {"type": "Feature", "id": "GA", "properties": {"name": "Georgia"}, "geometry": {"type": "Polygon", "coordinates": [[[[-83.109191, 35.00118], [-83.322791, 34.787579], [-83.322791, 34.787579], [-83.109191, 35.00118]]]}},
12 {"type": "Feature", "id": "HI", "properties": {"name": "Hawaii"}, "geometry": {"type": "MultiPolygon", "coordinates": [[[[-155.634835, 18.948267], [-155.881297, 19.03118], [-155.881297, 19.03118], [-155.634835, 18.948267]]]}},
13 {"type": "Feature", "id": "ID", "properties": {"name": "Idaho"}, "geometry": {"type": "Polygon", "coordinates": [[[[-116.04751, 49.000239], [-116.04751, 47.976051], [-116.04751, 47.976051], [-116.04751, 49.000239]]]}},
14 {"type": "Feature", "id": "IL", "properties": {"name": "Illinois"}, "geometry": {"type": "Polygon", "coordinates": [[[[-90.639984, 42.510065], [-88.788778, 42.493634], [-88.788778, 42.493634], [-90.639984, 42.510065]]]}],
15 {"type": "Feature", "id": "IN", "properties": {"name": "Indiana"}, "geometry": {"type": "Polygon", "coordinates": [[[[-85.750734, 40.212261], [-85.750734, 40.212261], [-85.750734, 40.212261], [-85.750734, 40.212261]]]}]
```

- **JSON**은 **Java Script Object Notation**의 약자...
- 웹서버와 클라이언트간 데이터 교환에 유리한 것으로 알려져 있다
- 그 구조가 **Python**에서는 **Dict**형과 닮아 있다

In [93]:

```
import json

customer = {
    'id': '000001',
    'name': '홍길동',
    'history': [
        {'date': '2018-05-22', 'log': True},
        {'date': '2018-05-23', 'log': False},
    ]
}
```

- 연습용 **dict** 데이터로 **customer**를 만들자
- **json**을 할 거니까 **json**을 **import**하고

In [94]:

```
# JSON 인코딩  
json_test = json.dumps(customer, indent=4, ensure_ascii=False)  
  
# 문자열 출력  
print(json_test)
```

```
{  
    "id": "000001",  
    "name": "홍길동",  
    "history": [  
        {  
            "date": "2018-05-23",  
            "log": false  
        }  
    ]  
}
```

- **json.dumps**를 사용하면 **json** 데이터로 만들어진다.

In [92]:

```
# Write JSON
with open('../data/03_json_tutorial.json', 'w', encoding="utf-8") as make_file:
    json.dump(customer, make_file, ensure_ascii=False, indent=4)
```

- Python의 권장이 파일 입출력에는 **with** 구문을 사용하는 것
- **with** 구문은 다음 기회에 다시 배우고
- 파일로 쓸때는 **dumps**가 아니라 **dump**이다.

```
03_US_Unemployment_Oct2012.csv x 03_us-states.json x 03_json_tutorial.json x
1 [
2   "id": "000001",
3   "name": "홍길동",
4   "history": [
5     {
6       "date": "2018-05-22",
7       "log": true
8     },
9     {
10       "date": "2018-05-23",
11       "log": false
12     }
13   ]
14 }
```

```
In [99]: with open('../data/03_us-states.json') as f:  
    json_test = json.load(f)
```

- 이번에는 읽어보자
- **json.load()**

In [100]:

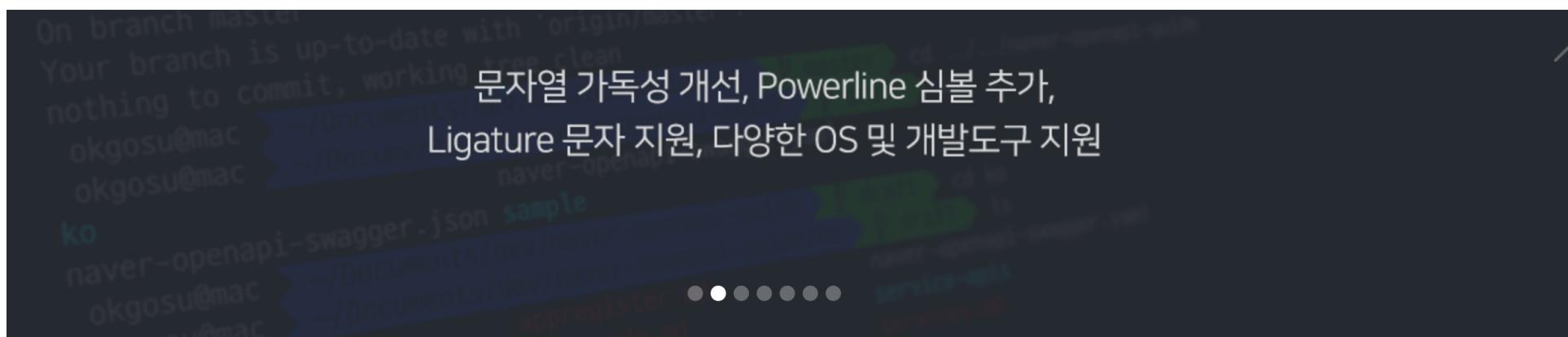
```
json_test
```

```
{'type': 'FeatureCollection',
 'features': [{ 'type': 'Feature',
    'id': 'AL',
    'properties': { 'name': 'Alabama' },
    'geometry': { 'type': 'Polygon',
      'coordinates': [[[[-87.359296, 35.00118],
                      [-85.606675, 34.984749],
                      [-85.431413, 34.124869],
                      [-85.184951, 32.859696],
                      [-85.069935, 32.580372],
                      [-84.960397, 32.421541],
                      [-85.004212, 32.322956],
                      [-84.889196, 32.262709],
                      [-85.058981, 32.13674],
```

네이버 API 사용



- 사실... 개인적으로 좋아하는 회사가 아니지만
- 그래도 구글과의 경쟁에서 살아남기 위해 나름 괜찮은 서비스가 많음



Clova



네이버 아이디로 로그인



Maps



papago



서비스 API



네이버 클라우드 플랫폼

Products

네이버에서 제공하는 다양한 서비스와 컨텐츠를 소개합니다.

Products > API 이용 안내 > API 소개

API 이용 안내



네이버 오픈 API 목록



API 소개

운영 정책

FAQ

BI 가이드

이용약관

상표 사용 가이드

Clova

네이버 아이디로 로그인

네이버 오픈API 목록 및 안내입니다.

API명	설명	호출제한
검색	네이버 블로그, 이미지, 웹, 뉴스, 백과사전, 책, 카페, 지식iN 등 검색	25,000회/일
지도(Web, Mobile)	네이버 지도 표시 및 주소 좌표 변환	20만/일
네이버 아이디로 로그인	외부 사이트에서 네이버 아이디로 로그인 기능 구현	없음

- 무료인 경우 제한이 좀 있다

서비스 API

Papago NMT 번역

인공신경망 기반 기계 번역 (영,중)

10,000글자/일

API 이용 안내

Clova

네이버 아이디로 로그인

지도

파파고

서비스 API



데이터랩

검색

단축URL

캡차

네이버 공유하기

모바일 앱 연동

네이버 오픈메인

검색



트윗



공유하기 1개

웹, 뉴스, 블로그 등 분야별 네이버 검색 결과를 웹 서비스 또는 모바일 앱에서 바로 보여 줄 수 있습니다. 또한 'OO역 맛집'과 같은 지역 검색을 할 수도 있으며, 부가적으로 성인검색어 판별과 한영키 오타 변환 기능을 이용하실 수 있습니다.

네이버 검색 결과 컨텐츠

웹 서비스 또는 모바일 앱에 네이버 웹문서/블로그/뉴스/책/영화/카페글/지식IN/쇼핑/이미지/백과사전/전문자료 분야에 대한 검색 결과를 보여 줄 수 있습니다.



지역 검색

'OO역 맛집', 'OO동 술집'과 같은 검색 결과를 보여주고 싶을 때 사용하며, 네이버 지역 서비스에 등록된 각 지역별 업체 및 상호 검색결과를 보여줍니다.



검색 부가 기능

검색 부가 기능으로 특정 검색어에 대해 성인검색어 여부를 알려주는 기능과 검색창에 입력된 오타를 바로 잡아주는 오타변환 기능을 제공합니다.



* 처리한도 : 25,000/일

네이버 검색 결과 컨텐츠

웹 서비스 뿐만 아니라 애드온이나 웹사이트 / 브로그 / 뉴스 / 채널 / 여행 / 카페 / 지식인 / 스피 / 이벤트 / 배고파서치 / 저널

- **오픈 API 이용 신청을 클릭하면 이제 시작된다.**

파파고

서비스 API

데이터랩

검색

단축URL

캡차

네이버 공유하기

모바일 앱 연동

네이버 오픈메인

지역 검색

'OO역 맛집', 'OO동 술집'과 같은 검색 결과를 보여주고 싶을 때 사용하며, 네이버 지역 서비스에 등록된 각 지역별 업체 및 상호 검색 결과를 보여줍니다.

검색 부가 기능

검색 부가 기능으로 특정 검색어에 대해 성인검색어 여부를 알려주는 기능과 검색창에 입력된 오타를 바로 잡아주는 오타변환 기능을 제공합니다.

* 처리한도 : 25,000/일

오픈 API 이용 신청

개발 가이드 보기

내 애플리케이션

애플리케이션 등록

Clova Platform Console β

API 제휴 신청

계정 설정

애플리케이션 등록 (API 이용신청)

애플리케이션의 기본 정보를 등록하면, 좌측 [내 애플리케이션](#) 메뉴의 서브 메뉴에 등록하신 애플리케이션 이름으로 서브 메뉴가 만들어집니다.

The screenshot shows the 'API Registration' section of the Clova Platform Console. It includes fields for 'Application Name' (with validation rules), 'API Usage' (with dropdown and search), and 'Open API Service Environment' (with dropdown and validation). A red exclamation mark icon is present in several validation boxes.

애플리케이션 이름

애플리케이션 이름

- 네이버 아이디로 로그인할 때 사용자에게 표시되는 이름이므로 가급적 10자 이내의 간결한 이름을 사용해주세요.
- 40자 이내의 영문, 한글, 숫자, 공백문자, "-", "_" 만 입력 가능합니다.

선택하세요.

검색

비로그인 오픈 API 서비스 환경

환경 추가

- 필수인 품을 채우고~

- [비로그인 오픈 API 서비스 환경] 설정을 확인해 주세요.

애플리케이션 등록 (API 이용신청)

애플리케이션의 기본 정보를 등록하면, 좌측 [내 애플리케이션](#) 메뉴의 서브 메뉴에 등록하신 애플리케이션 이름으로 서브 메뉴가 만들어집니다.

애플리케이션 이름

PinkWink



- 네이버 아이디로 로그인할 때 사용자에게 표시되는 이름이므로 가급적 10자 이내의 간결한 이름을 사용해주세요.
- 40자 이내의 영문, 한글, 숫자, 공백문자, "-", "_" 만 입력 가능합니다.

사용 API

선택하세요.



검색

비로그인 오픈 API
서비스 환경

환경 추가



애플리케이션 이름

PinkWink ✓

- 네이버 아이디로 로그인할 때 사용자에게 표시되는 이름이므로 가급적 10자 이내의 간결한 이름을 사용해주세요.
- 40자 이내의 영문, 한글, 숫자, 공백문자, "-", "_"만 입력 가능합니다.

선택하세요. ▼

사용 API

검색 X

환경 추가 ▼

WEB 설정 X ^

웹 서비스 URL (최대 10개)

http://localhost + ✓

- 텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제됩니다.

비로그인 오픈 API
서비스 환경

우리가 접근하는 주소가 <http://localhost>이다. Jupyter Notebook

URI)

비로그인 오픈 API 서비스 환경

환경 추가



WEB 설정

X ^

웹 서비스 URL (최대 10개)

http://localhost



- 텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제 됩니다.
- http와 https는 구분하지 않습니다.
- www는 빼고 입력해 주세요. 예) http://naver.com
- 서브 도메인이 있으면 대표 도메인명만 입력해 주세요. (예: http://naver.com)
- 하이브리드 앱은 location.href 객체 출력 값은 입력하면 됩니다. (예: file:///로컬 URI)

등록하기

취소

내 애플리케이션 ^

PinkWink API

PinkWink

애플리케이션 등록

Clova Platform Console β

API 제휴 신청

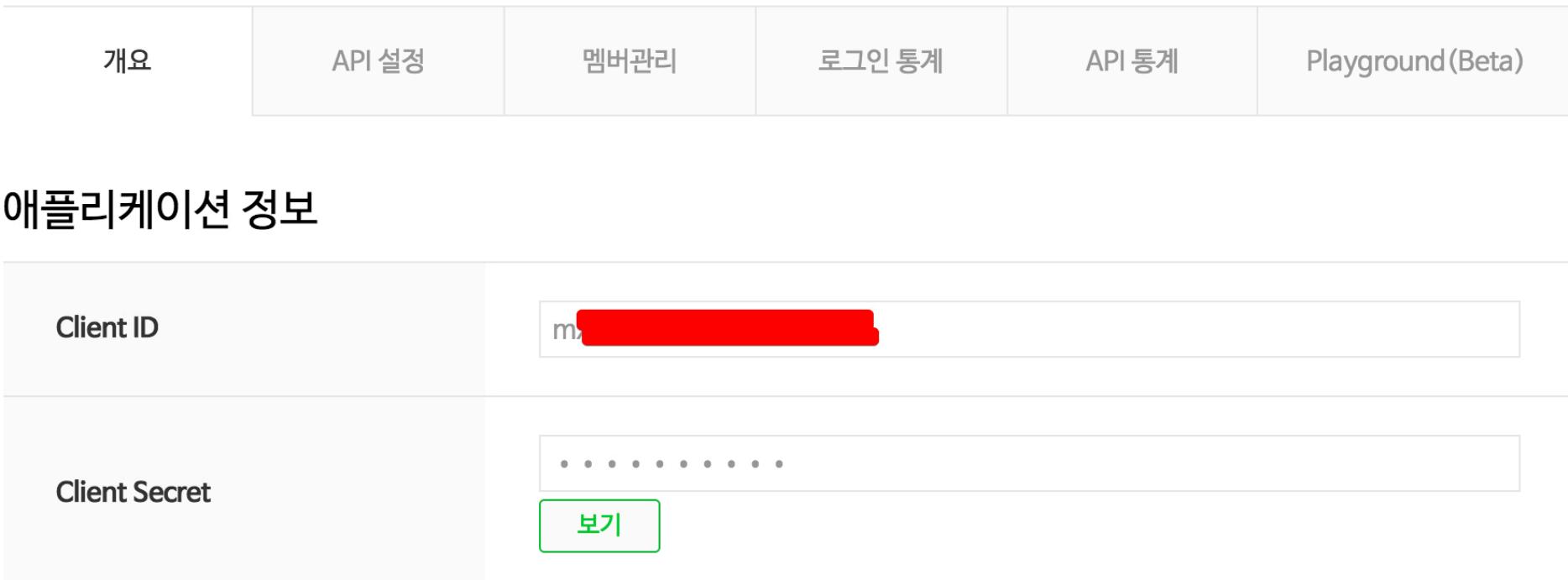
계정 설정

PinkWink

개요 API 설정 멤버관리 로그인 통계 API 통계 Playground(Beta)

애플리케이션 정보

Client ID	m.....
Client Secret 보기



- 여기 **ID**와 **Secret** 정보는 잘 보관하자

네이버 API 어떻게 쓰나 ~~~ 슬쩍 한번 보기

네이버 아이디로 로그인

지도

파파고

서비스 API

데이터랩

검색

- 블로그

- 뉴스

- 책

- 성인 검색어 판별

- 백과사전

- 영화

- 카페글

- 지식iN

- 지역

- 오타변환

- 웹문서

- 이미지

0.API 호출 예제

예제 실행 전에 아래 1.준비사항 항목들을 꼭 체크하시길 바랍니다.

Java

PHP

Node.js

Python

C#

```
# 네이버 검색 API예제는 블로그를 비롯 전문자료까지 호출방법이 동일하므로 blog검색만 대표로 예제를 올렸습니다.  
# 네이버 검색 Open API 예제 - 블로그 검색  
import os  
import sys  
import urllib.request  
client_id = "YOUR_CLIENT_ID"  
client_secret = "YOUR_CLIENT_SECRET"  
encText = urllib.parse.quote("검색할 단어")  
url = "https://openapi.naver.com/v1/search/blog?query=" + encText # json 결과  
# url = "https://openapi.naver.com/v1/search/blog.xml?query=" + encText # xml 결과  
request = urllib.request.Request(url)  
request.add_header("X-Naver-Client-Id",client_id)  
request.add_header("X-Naver-Client-Secret",client_secret)  
response = urllib.request.urlopen(request)  
rescode = response.getcode()  
if(rescode==200):  
    response_body = response.read()  
    print(response_body.decode('utf-8'))  
else:  
    print("Error Code:" + rescode)
```

- 일단, 과히 친절하지 않지만 API에 대한 Python 예제가 있다

2. API 기본 정보

메서드	인증	요청 URL	출력 포맷
GET	-	https://openapi.naver.com/v1/search/blog.xml	XML
GET	-	https://openapi.naver.com/v1/search/blog.json	JSON

```
In [1]: import urllib.request  
client_id = 'mXwxin9tet7Ygwrada2M'  
client_secret = "FA7pQsqMFy"  
encText = urllib.parse.quote("파이썬")  
url = "https://openapi.naver.com/v1/search/blog?query=" + encText # json 결과  
request = urllib.request.Request(url)  
request.add_header("X-Naver-Client-Id",client_id)  
request.add_header("X-Naver-Client-Secret",client_secret)
```

```
In [1]:  
import urllib.request  
client_id = 'mXwxin9tet7YgwradA2M'  
client_secret = "FA7pQsqMFy"  
encText = urllib.parse.quote("파이썬")  
url = "https://openapi.naver.com/v1/search/blog?query=" + encText # json 결과  
request = urllib.request.Request(url)  
request.add_header("X-Naver-Client-Id",client_id)  
request.add_header("X-Naver-Client-Secret",client_secret)
```

- **urllib**: http 프로토콜에 따라서 서버의 요청/응답을 처리하기 위한 모듈
- **urllib.request**: 클라이언트의 요청을 처리하는 모듈
- **urllib.parse**: url 주소에 대한 분석

3. 요청 변수

요청 변수	타입	필수 여부	기본값	설명
query	string	Y	-	검색을 원하는 문자열로서 UTF-8로 인코딩한다.
display	integer	N	10(기본값), 100(최대)	검색 결과 출력 건수 지정
start	integer	N	1(기본값), 1000(최대)	검색 시작 위치로 최대 1000까지 가능
sort	string	N	sim(기본값), date	정렬 옵션: sim (유사도순), date ('날짜 순')

In [2]:

```
response = urllib.request.urlopen(request)
rescode = response.getcode()
if(rescode==200):
    response_body = response.read()
    print(response_body.decode('utf-8'))
else:
    print("Error Code:" + rescode)

{
"lastBuildDate": "Wed, 06 Jun 2018 15:41:18 +0900",
"total": 112122,
"start": 1,
"display": 10,
"items": [
{
"title": "<b>파이썬</b> 설치하기와 파이참(IDE툴)설치방법",
"link": "http://blog.naver.com/luckynoon77RedirectIndex.nhn?logNo=221100716152"
```

- `response.getcode()`에서 정상이라면 200이 반환된다

서비스 API

데이터랩

검색

- 블로그
- 뉴스
- 책
- 성인 검색어 판별
- 백과사전
- 영화
- 카페글
- 지식iN
- 지역
- 오타변환
- 웹문서
- 이미지

2. API 기본 정보

메서드	인증	요청 URL	출력 포맷	설명
GET	-	https://openapi.naver.com/v1/search/book.xml	XML	책 기본 검색
GET	-	https://openapi.naver.com/v1/search/book_adv.xml	XML	책 상세 검색
GET	-	https://openapi.naver.com/v1/search/book.json	JSON	책 기본 검색

In [3]:

```
encText = urllib.parse.quote("파이썬")
url = "https://openapi.naver.com/v1/search/book?query=" + encText # json 결과
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)
rescode = response.getcode()
print(response.read().decode('utf-8'))
```

```
{
    "lastBuildDate": "Wed, 06 Jun 2018 15:43:59 +0900",
    "total": 630,
    "start": 1,
    "display": 10,
    "items": [
        {
            "title": "<b>파이썬</b>으로 데이터 주무르기 (독특한 예제를 통해 배우는 데이터 분석 입문)",
            "link": "http://book.naver.com/bookdb/book_detail.php?bid=12898027",
            "image": "http://bookthumb.phinf.naver.net/cover/128/980/12898027.jpg?type=m1&udate=20180316",
```

- 네이버 책 정보를 얻을 수 있다.

Cells: 24 / 59,

In [4]:

```
encText = urllib.parse.quote("파이썬")
url = "https://openapi.naver.com/v1/search/movie?query=" + encText # json 결과
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)
rescode = response.getcode()
print(response.read().decode('utf-8'))
```

```
{
  "lastBuildDate": "Wed, 06 Jun 2018 15:45:18 +0900",
  "total": 1,
  "start": 1,
  "display": 1,
  "items": [
    {
      "title": "<b>파이썬</b> 앤 가드",
```

- 네이버 무비 정보를 얻을 수 있다.

In [5]:

```
encText = urllib.parse.quote("파이썬")
url = "https://openapi.naver.com/v1/search/cafearticle?query=" + encText # json 결과
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)
print(response.read().decode('utf-8'))
```

{
 "title": "파이썬 소수출력 코딩좀 알려주세요..",
 "link": "http://cafe.naver.com/suhui/19985967",
 "description": "컴공은 아니지만 교필이라 듣는수업인데요 파이썬과제 정말 한개도 모르겠어요.. 이번주과제가 소수출력하라는게 혹시 코딩하시는분 저 좀 도와주세요...ㅜㅜ ===== [필독] 답변 받은 후 질문글 무단...",
 "cafename": "수만휘-수능날만점시험지를휘날리자★...",
 "cafeurl": "http://cafe.naver.com/suhui"

- 네이버 카페 정보를 얻을 수 있다.

In [6]:

```
encText = urllib.parse.quote("파이썬")
url = "https://openapi.naver.com/v1/search/shop?query=" + encText # json 결과
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)
print(response.read().decode('utf-8'))
```

```
"title": "닥스 이라운드몰 가방 acc <b>파이톤</b>패턴 가죽 클러치백 WCBA7E804BK",
"link": "http://shopping.naver.com/gate.nhn?id=14445706064",
"image": "https://shopping-phinf.pstatic.net/main_1444570/14445706064.20180603033717.jpg",
"lprice": "143870",
"hprice": "225540",
"mallName": "네이버",
"productId": "14445706064",
"productType": "1"
```

- 네이버 쇼핑 정보를 얻을 수 있다.

In [7]:

```
encText = urllib.parse.quote("파이썬")
url = "https://openapi.naver.com/v1/search/encyc?query=" + encText # json 결과
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)
print(response.read().decode('utf-8'))
```

{
"title": "파이썬",
"link": "https://terms.naver.com/entry.nhn?docId=3580815&cid=59088&categoryId=59096",
"description": "'파이썬'이다. 간결한 문법으로 입문자가 이해하기 쉽고, 다양한 분야에 활용할 수 있기 때문이다. 이 외에도 파이썬은 머신러닝, 그래픽, 웹 개발 등 여러 업계에서 선호하는 언어로 꾸준히... ",
"thumbnail": "http://openapi-dbscthumb.phinf.naver.net/4749_000_1/20170118193349632_0CHSSS5Y6.png/01_16.png?type=m160_160"

- 네이버 백과사전 정보를 얻을 수 있다.

In [8]:

```
encText = urllib.parse.quote("몰스킨")
url = "https://openapi.naver.com/v1/search/shop?query=" + encText # json 결과
request = urllib.request.Request(url)
request.add_header("X-Naver-Client-Id",client_id)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)
print(response.read().decode('utf-8'))
```

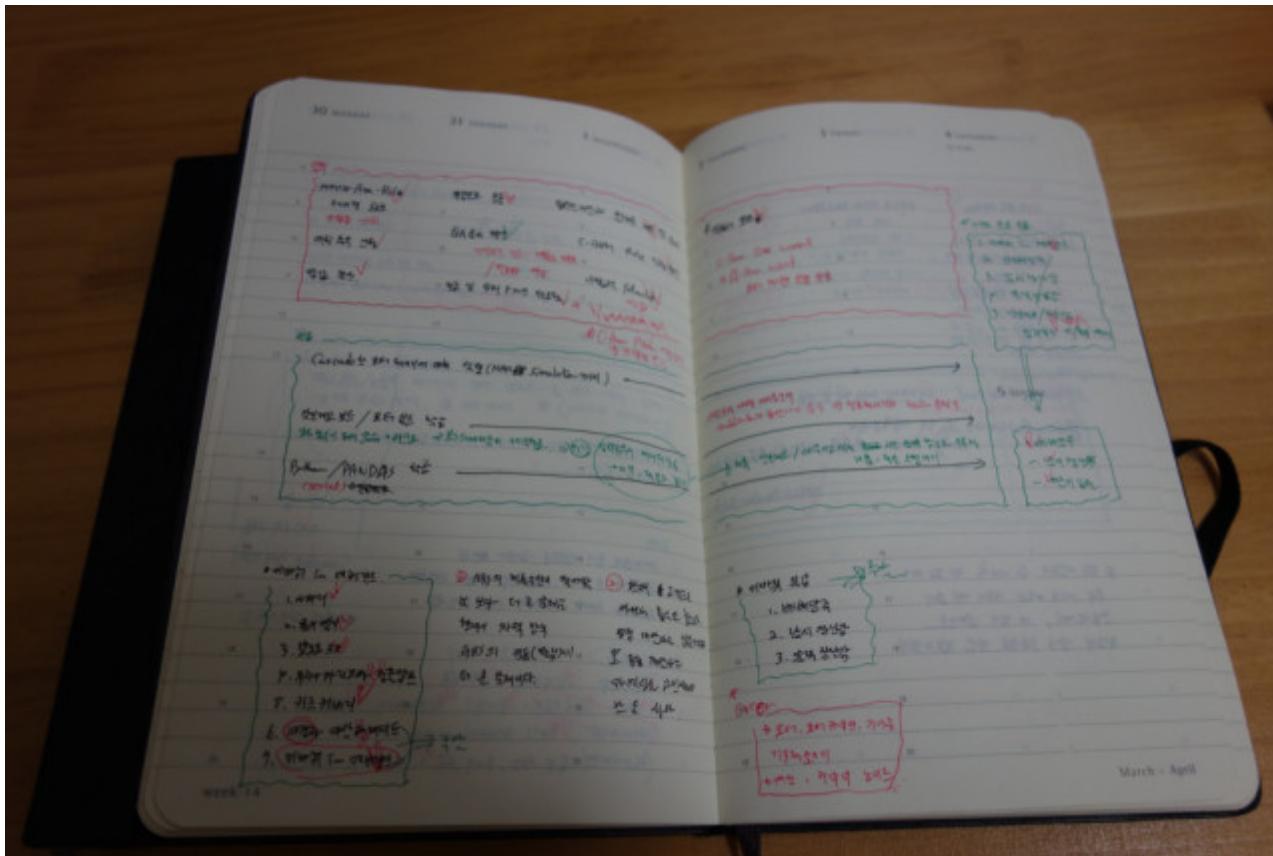
```
{
  "lastBuildDate": "Wed, 06 Jun 2018 15:50:26 +0900",
  "total": 21229,
  "start": 1,
  "display": 10,
  "items": [
    {
      "title": "<b>몰스킨</b> 클래식 룰드 줄지",
```

- 쇼핑에서 몰스킨을 검색해보자

Bent gij vandaag wifelangt niet gesproken
 met my plezier - Doe is die weer terug gekomen
 in Paris - my stede na de zonck van Cottier?
 Zoudt gij by gescreven horen ons voor my welken
 gretens - en by een meer Hollantse taal
 jou een bezoek vieren hem - Doe ik my trouwe
 reis begeerlyk heff evenlyk - my blyghe
 genoegh en dater - Ik wan niet dat gheen
 ammeren konst - Dat niet te vergelijken de juue
 hem verschieden dingen te regel hebben nog
 over landen - Kusgh hy de lithograph heb noch
 gelyken? Ik sall graech wat ons gehelt op
 meeuw huis en een matras ik vond altyd veel
 aantrekkelijkern dan die hy niet vult dingen en
 gevuld just in juttien van kint en oorprontelsch
 Engher is vermaerd met een Karatels
 Onnedaand Karatels is naer een tekening
 die ik van morgen vrydag begon en die later
 dag op toegewiende - So my scher wie ic
 besta die ik tot nog toe het gemaakte althans
 van lecht en bruin - In smin u het Karatels
 afteken ik onmagh op dat papier want den Karatels
 die tot diezelde Kretsel te tekennen komde ic liek
 gy een in jeur jutt wat ic door de verandering
 van 't lecht op 't sletten won - Dat figuree
 is gynghed tegen 't lecht in en um 't smin te
 gewen konst als anders nay by hen den een
 contour doen - Door heel uit
 een anteken figuer de lumine het moedle
 Karatels liggt er de Kraetels in harmonie met
 elkaer en in onderting rapport - En brengt
 deze opvalting mede ghele wel de moedeligheden
 vunck weergeven van 't geen men van oogen heff
 doch tress noch als anders waar men veel
 meer te blokken heeft namlyk de Karatels
 en figuer zulc le pectaten en het lecht
 zulc le pectaten dat het Karatels 't best en
 meer uitkomen wil konst - 't geen men beeld en of binner
 heel moet meer soe gevuld moet hebben dan op right van 't lecht
 dat moet het figuer van 't smin te tekennen



• 반고흐의 몰스킨 노트의 메모



- 일상을 적고 기록하는 것은 참으로 중요한듯...

In [9]:

```
import datetime

def get_request_url(API_url):
    client_id = 'mXwxin9tet7YgwradA2M'
    client_secret = "FA7pQsqMFy"

    request = urllib.request.Request(API_url)
    request.add_header("X-Naver-Client-Id", client_id)
    request.add_header("X-Naver-Client-Secret", client_secret)

    response = urllib.request.urlopen(request)
    if response.getcode() == 200:
        print ("[%s] Url Request Success" % datetime.datetime.now())
        return response.read().decode('utf-8')
    else:
        print("[" + str(datetime.datetime.now()) + "] Error -----")
        return None
```

- 놀래지 말자. 여기서 우리가 모르는 것은 없다

```
In [10]:  
encText = urllib.parse.quote("몰스킨")  
API_url = "https://openapi.naver.com/v1/search/shop.json?query=" + encText  
print(get_request_url(API_url))
```

```
[2018-06-06 15:58:52.782956] Url Request Success  
{  
    "lastBuildDate": "Wed, 06 Jun 2018 15:58:52 +0900",  
    "total": 21229,  
    "start": 1,  
    "display": 10,  
    "items": [  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_1_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        },  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_2_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        },  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_3_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        },  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_4_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        },  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_5_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        },  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_6_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        },  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_7_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        },  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_8_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        },  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_9_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        },  
        {  
            "title": "Moleskine Classic Notebook - Ruled - Large - Black - Hard Cover (5 x 8.25) (Classic Notebooks)",  
            "link": "https://www.amazon.co.jp/dp/B00000HJLW/ref=cm_sw_r_cp_apa_180606155852782956_1_10_0_0_0_0?ie=UTF8&psc=1",  
            "image": "https://m.media-amazon.com/images/I/51yOOGzXfYL._AC_UF800x800_.jpg",  
            "price": 10000,  
            "discount": 0,  
            "rating": 4.5,  
            "review": 10000  
        }  
    ]  
}
```

- `get_request_url` 함수를 이용해서 API url을 던지면 결과를 얻도록
- 몰스킨의 쇼핑몰 정보를 얻으로 가보자

```
In [11]: def get_search_result(api_node, search_text, start_num, disp_num):

    base = "https://openapi.naver.com/v1/search"
    node = "/" + api_node + ".json"
    param_query = "?query=" + urllib.parse.quote(search_text)
    param_start = "&start=" + str(start_num)
    param_disp = "&display=" + str(disp_num)

    url = base + node + param_query + param_start + param_disp

    print(url)
```

```
In [12]: get_search_result('shop', 'TEST', 10, 3)
```

https://openapi.naver.com/v1/search/shop.json?query=TEST&start=10&display=3

- 그리고 하나더, `url`을 만드는 함수도 만들자

In [13]:

```
import json

def get_search_result(api_node, search_text, start_num, disp_num):
    base = "https://openapi.naver.com/v1/search"
    node = "/" + api_node + ".json"
    param_query = "?query=" + urllib.parse.quote(search_text)
    param_start = "&start=" + str(start_num)
    param_disp = "&display=" + str(disp_num)

    url = base + node + param_query + param_start + param_disp

    getting_data = get_request_url(url)

    if (getting_data == None):
        return None
    else:
        return json.loads(getting_data)
```

- 방금 함수에 내용을 좀 더 추가하자 – **json**으로 출력을 내도록 ...

In [14]:

```
search_result = get_search_result('shop', '몰스킨', 1, 2)
```

[2018-06-06 16:00:51.588606] Url Request Success

In [15]:

```
search_result
```

```
{'lastBuildDate': 'Wed, 06 Jun 2018 16:00:51 +0900',
'total': 21229,
'start': 1,
'display': 2,
'items': [{ 'title': '<b>몰스킨</b> 클래식 룰드 줄지',
'link': 'http://shopping.naver.com/gate.nhn?id=7398991731',
'image': 'https://shopping-phinf.pstatic.net/main_7398991/7398991731.20180112135400.jpg',
```

- 만든 함수는 이렇게 사용하면 된다.
- 아~ 알 수 없는 이유로 출력 순서가 다를 수 있습니다.~

In [16]:

```
search_result['items']
```

```
[{'title': '<b>몰스킨</b> 클래식 룰드 줄지',
 'link': 'http://shopping.naver.com/gate.nhn?id=7398991731',
 'image': 'https://shopping-phinf.pstatic.net/main_7398991/7398991731.20180112135400.jpg',
 'lprice': '15700',
 'hprice': '35300',
 'mallName': '네이버',
 'productId': '7398991731',
 'productType': '1'},
 {'title': '<b>몰스킨</b> 까이에 엑스라지 다이어리',
 'link': 'http://shopping.naver.com/gate.nhn?id=10010203158',
 'image': 'https://shopping-phinf.pstatic.net/main_1001020/10010203158.20160704125951.jpg',
 'lprice': '8800',
 'hprice': '34100',
 'mallName': '네이버',
 'productId': '10010203158',
 'productType': '1'}]
```

```
In [17]: search_result['start'], search_result['total'], search_result['display']

(1, 21229, 2)
```

```
In [18]: search_result['items'][0]['title']

'<b>몰스킨</b> 클래식 룰드 줄지'
```

```
In [19]: search_result['items'][0]['lprice'], search_result['items'][0]['hprice']

('15700', '35300')
```

- 정보를 얻는 방법을 확인하자

In [39]:

```
search_result['items'][0]['title']
```

'몰스킨 클래식 룰드 줄지'

In [43]:

```
search_result['items'][0]['title'].replace("<b>", "")
```

'몰스킨 클래식 룰드 줄지'

- 여기서 태그를 없앨 수 있을까? → **replace** (왜? string 객체니까)

In [45]:

```
def delete_tag(input_str):
    input_str = input_str.replace("<b>", "")
    input_str = input_str.replace("</b>", "")
    return input_str
```

- 이것도 함수로 만들어 두자.

In [46]:

```
import pandas as pd

def get_fields(post):
    title = []
    link = []
    lprice = []
    hprice = []

    for each in post:
        title.append(delete_tag(each['title']))
        link.append(each['link'])
        lprice.append(each['lprice'])
        hprice.append(each['hprice'])

    result_pd = pd.DataFrame({'title':title, 'lprice':lprice,
                             'hprice':hprice, 'link':link},
                            columns=['title','lprice','hprice','link'])

    return result_pd
```

In [47]:

```
get_fields(search_result[ 'items' ])
```

	title	lprice	hprice	link
0	몰스킨 클래식 룰드 줄지	15700	40530	http://search.shopping.naver.com/gate.nhn?id=7...
1	몰스킨 까이에 엑스라지 다이어리	8800	34100	http://search.shopping.naver.com/gate.nhn?id=1...

In [48]:

```
search_result = get_search_result('shop', '몰스킨', 1, 2)
tmp1 = get_fields(search_result['items'])
tmp1
```

[2018-08-15 11:56:07.471393] Url Request Success

	title	lprice	hprice	link
0	몰스킨 클래식 룰드 줄지	15700	40530	http://search.shopping.naver.com/gate.nhn?id=7...
1	몰스킨 까이에 엑스라지 다이어리	8800	34100	http://search.shopping.naver.com/gate.nhn?id=1...

In [49]:

```
search_result = get_search_result('shop', '몰스킨', 3, 2)
tmp2 = get_fields(search_result['items'])
tmp2
```

[2018-08-15 11:56:21.217085] Url Request Success

	title	lprice	hprice	link
0	KM 몰스킨 클래식 룰드 줄지 소프트커버 라지 노트북 9788	19400	0	http://search.shopping.naver.com/gate.nhn?id=1...
1	몰스킨 2017 위클리 18개월 그린 하드 다이어리	20000	228500	http://search.shopping.naver.com/gate.nhn?id=1...

In [50]:

```
tmp1.append(tmp2)
```

	title	lprice	hprice	link
0	몰스킨 클래식 룰드 줄지	15700	40530	http://search.shopping.naver.com/gate.nhn?id=7...
1	몰스킨 까이에 엑스라지 다이어리	8800	34100	http://search.shopping.naver.com/gate.nhn?id=1...
0	KM 몰스킨 클래식 룰드 줄지 소프트커버 라지 노트북 9788	19400	0	http://search.shopping.naver.com/gate.nhn?id=1...
1	몰스킨 2017 위클리 18개월 그린 하드 다이어리	20000	228500	http://search.shopping.naver.com/gate.nhn?id=1...

In [51]:

```
result_mol = []
for n in range(1,100,10):
    search_result = get_search_result('shop', '몰스킨', n, 10)
    result_mol.append(get_fields(search_result['items']))

result_mol = pd.concat(result_mol)
```

```
[2018-08-15 11:57:14.582595] Url Request Success
[2018-08-15 11:57:14.730617] Url Request Success
[2018-08-15 11:57:14.836913] Url Request Success
[2018-08-15 11:57:14.947669] Url Request Success
[2018-08-15 11:57:15.087177] Url Request Success
[2018-08-15 11:57:15.245817] Url Request Success
[2018-08-15 11:57:15.372885] Url Request Success
[2018-08-15 11:57:15.498227] Url Request Success
[2018-08-15 11:57:15.666942] Url Request Success
[2018-08-15 11:57:15.832148] Url Request Success
```

- 10개씩 디스플레이하고 총 100개를 수집하자

In [52]:

```
result_mol.head( )
```

		title	lprice	hprice	link
0	몰스킨 클래식 룰드 줄지		15700	40530	http://search.shopping.naver.com/gate.nhn?id=7...
1	몰스킨 까이에 엑스라지 다이어리		8800	34100	http://search.shopping.naver.com/gate.nhn?id=1...
2	KM 몰스킨 클래식 룰드 줄지 소프트커버 라지 노트북 9788	19400	0		http://search.shopping.naver.com/gate.nhn?id=1...
3	몰스킨 2017 위클리 18개월 그린 하드 다이어리	20000	228500		http://search.shopping.naver.com/gate.nhn?id=1...
4	몰스킨 2018 스누피 다이어리	12000	0		http://search.shopping.naver.com/gate.nhn?id=1...

- 와우~~~~

In [53]:

```
result_mol = result_mol.reset_index(drop=True)  
result_mol
```

		title	lprice	hprice	link
0	몰스킨 클래식 룰드 줄지		15700	40530	http://search.shopping.naver.com/gate.nhn?id=7...
1	몰스킨 까이에 엑스라지 다이어리		8800	34100	http://search.shopping.naver.com/gate.nhn?id=1...
2	KM 몰스킨 클래식 룰드 줄지 소프트커버 라지 노트북 9788		19400	0	http://search.shopping.naver.com/gate.nhn?id=1...
3	몰스킨 2017 위클리 18개월 그린 하드 다이어리		20000	228500	http://search.shopping.naver.com/gate.nhn?id=1...
4	몰스킨 2018 스누피 다이어리		12000	0	http://search.shopping.naver.com/gate.nhn?id=1...
5	[한정판]몰스킨 슈퍼마리오 하드룰드L 풀스크린 블루		35739	0	http://search.shopping.naver.com/gate.nhn?id=1...
6	몰스킨 스타벅스 2017 플래너 데일리 핑크		17000	36400	http://search.shopping.naver.com/gate.nhn?id=1...
7	2018년 몰스킨 다이어리 한정판 포켓/L 사이즈		27600	0	http://search.shopping.naver.com/gate.nhn?id=1...
8	몰스킨 피너츠 데일리 2018 코랄오렌지 라지 55532		8990	0	http://search.shopping.naver.com/gate.nhn?id=1...
9	[한정판]몰스킨 슈퍼마리오 하드 룰드P 게임보이 그린		25540	0	http://search.shopping.naver.com/gate.nhn?id=1...

```
In [54]:
```

```
result_mol['lprice'] = result_mol['lprice'].astype('float')
result_mol['hprice'] = result_mol['hprice'].astype('float')
result_mol.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
title      100 non-null object
lprice     100 non-null float64
hprice     100 non-null float64
link       100 non-null object
dtypes: float64(2), object(2)
memory usage: 3.2+ KB
```

In [29]:

```
writer = pd.ExcelWriter("../data/05_molskin_diary_in_naver_shop.xlsx",
                      engine='xlsxwriter')
result_mol.to_excel(writer, sheet_name='Sheet1')

workbook = writer.book
worksheet = writer.sheets['Sheet1']
worksheet.set_column('A:A', 4)
worksheet.set_column('B:B', 60)
worksheet.set_column('C:C', 10)
worksheet.set_column('D:D', 10)
worksheet.set_column('E:E', 50)

worksheet.conditional_format('C2:C101', {'type': '3_color_scale'})

writer.save()
```

- 엑셀로 조금 신경써서 저장해보자~~~~

A	B	C	D	E
63	Moleskine Volant Journal (Set of 2), Large, Plain, Powder Blue	37000	0	http://search.shopping.naver.com/gate.nhn?id=15051118459
62	몰스킨 볼란트 플레인 (무지) 노트북 라지사이즈 소프트커버 Moleskine	18300	0	http://search.shopping.naver.com/gate.nhn?id=15010255193
63	KM 몰스킨 볼란트 플레인 노트북 소프트커버 라지 Moleskine	18300	0	http://search.shopping.naver.com/gate.nhn?id=15010255212
64	[추가 5% 캐시백][Hmall]몰스킨 (현대Hmall) 정품 무료각인 몰스킨/한정판	35640	0	http://search.shopping.naver.com/gate.nhn?id=14999486767
65	몰스킨 [18슈퍼마리오]롤드/게임보이 그린 P	28600	0	http://search.shopping.naver.com/gate.nhn?id=14900748609
66	몰스킨 2017 한정판 어린왕자 다이어리 라지 하드커버	8000	0	http://search.shopping.naver.com/gate.nhn?id=14847755216
67	[한정판]몰스킨 슈퍼마리오 하드 롤드P 게임보이 그린	25812	0	http://search.shopping.naver.com/gate.nhn?id=14936344225
68	[한정판] 몰스킨 14피너츠 하드커버 노트북 롤드-라지	34381	0	http://search.shopping.naver.com/gate.nhn?id=14772946584
69	[한정판]몰스킨 슈퍼마리오 하드 롤드P 게임보이 그린	26600	0	http://search.shopping.naver.com/gate.nhn?id=14908312054
70	KM 몰스킨 볼란트 플레인 노트북 소프트커버 라지 Moleskine	18300	0	http://search.shopping.naver.com/gate.nhn?id=12036166934
71	몰스킨 2019 위클리 18M 다이어리 하드L 스칼렛레드	31110	0	http://search.shopping.naver.com/gate.nhn?id=14246494677
72	Moleskine 2-Pack Classic Black Notebooks (pack of 2)	105400	0	http://search.shopping.naver.com/gate.nhn?id=14785938931
73	[한정판]몰스킨 슈퍼마리오 하드롤드L 풀스크린 블루	36492	0	http://search.shopping.naver.com/gate.nhn?id=14939912276
74	[핫트랙스] [18닥터수스]롤드/블루 L	37620	0	http://search.shopping.naver.com/gate.nhn?id=14992808177
75	@@@정품 무료각인@@@[몰스킨/한정판]닥터수스 그린치-롤드-화이트/라지	32440	0	http://search.shopping.naver.com/gate.nhn?id=14998455955
76	[18아톰]롤드/다크그레이 L 노트	39600	0	http://search.shopping.naver.com/gate.nhn?id=15058918056
77	[몰스킨]몰스킨 메신저백 클래식 레더 컬렉션 슬림 블랙	272600	0	http://search.shopping.naver.com/gate.nhn?id=15006723883
78	몰스킨 (현대Hmall) 정품 무료각인 [몰스킨/한정판]닥터수스 그린치-롤드	33140	0	http://search.shopping.naver.com/gate.nhn?id=14999466349
79	몰스킨 (현대Hmall) 정품 무료각인 몰스킨/한정판 닥터수스 그린치-롤드	33150	0	http://search.shopping.naver.com/gate.nhn?id=14998368312
80	몰스킨 Professional Notebook XL Hard, Aster Grey (8051272891386)	72480	0	http://search.shopping.naver.com/gate.nhn?id=14829057179
81	[추가 5% 캐시백][Hmall]몰스킨 (현대Hmall) 정품 무료각인 몰스킨/한정판	35640	0	http://search.shopping.naver.com/gate.nhn?id=14999482631
82	몰스킨 [18슈퍼마리오]롤드/풀스크린 블루 L	39600	0	http://search.shopping.naver.com/gate.nhn?id=14900748611
83	[한정판] 몰스킨 14피너츠 하드커버 노트북 롤드-라지	34747	0	http://search.shopping.naver.com/gate.nhn?id=14772963590
84	몰스킨 북 저널	32080	0	http://search.shopping.naver.com/gate.nhn?id=14462045436
85	몰스킨 피너츠 레드 데일리 플래너 다이어리 라지사이즈	19000	0	http://search.shopping.naver.com/gate.nhn?id=14711290665
86	[한정판]몰스킨 슈퍼마리오 하드 롤드P 게임보이 그린	25812	0	http://search.shopping.naver.com/gate.nhn?id=14939898538
87	KM 몰스킨 피너츠 데일리 플래너 다이어리 라지사이즈(레드)	19500	0	http://search.shopping.naver.com/gate.nhn?id=14848596004
88	[핫트랙스] 몰스킨 2019위클리(18M)/블랙 하드 XL	35530	0	http://search.shopping.naver.com/gate.nhn?id=14992808645
89	[18아톰]롤드/라이트그레이 L 노트	39600	0	http://search.shopping.naver.com/gate.nhn?id=15058918055
90	@@@정품 무료배송@@@[몰스킨]블렌드 롤드-레드(라지)	29730	0	http://search.shopping.naver.com/gate.nhn?id=15008472194
91	몰스킨 (현대Hmall)[한정판]몰스킨 슈퍼마리오 하드롤드L 풀스크린 블루	34250	0	http://search.shopping.naver.com/gate.nhn?id=14997197766

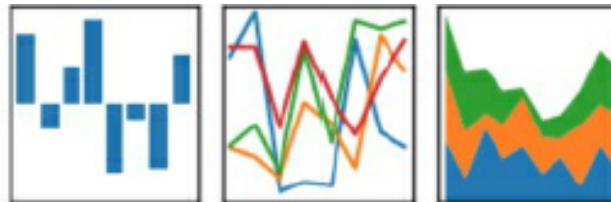
-
- 좀 더 깊이 있게는 여기를 들어가보자

http://xlsxwriter.readthedocs.io/working_with_pandas.html

Pandas 기초

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



[overview](#) // [get pandas](#) // [documentation](#) // [community](#) // [talks](#) // [donate](#)

pandas: powerful Python data analysis toolkit

- Python에서 R 만큼의 강력한 데이터 핸들링 성능을 제공하는 모듈
- 단일 프로세스에서는 최대 효율
- 코딩 가능하고 응용 가능한 엑셀로 받아들여도 됨
- 누군가는 스테로이드를 맞은 엑셀로 표현함

```
In [12]: import numpy as np  
s = pd.Series([1,3,5,np.nan,6,8])  
s
```

```
0    1.0  
1    3.0  
2    5.0  
3    NaN  
4    6.0  
5    8.0  
dtype: float64
```

- pandas의 기본은 한 줄 짜리 **Series**이다.

In [13]:

```
dates = pd.date_range('20130101', periods=6)
dates
```

```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
                 '2013-01-05', '2013-01-06'],
                dtype='datetime64[ns]', freq='D')
```

- **date_range**를 통해 편하게 시간에 접근할 수 있다

In [15]:

```
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=[ 'A','B','C','D' ] )  
df
```

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-06	-0.851687	0.927910	0.516239	0.175822

- pandas에서 가장 많이 사용하는 형태는 엑셀과 유사한 **DataFrame**이다

PANDAS의 DATAFRAME의 구조

Column Name

Value

Index

Column

	기관명	소계	2013년도 이전	2014년	2015년	2016년
0	강남구	2780	1292	430	584	932
1	강동구	773	379	99	155	377
2	강북구	748	369	120	138	204
3	강서구	884	388	258	184	81
4	관악구	1496	846	260	390	613

```
In [16]: data = {'A' : 1.,
             'B' : pd.Timestamp('20130102'),
             'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
             'D' : np.array([3] * 4,dtype='int32'),
             'E' : pd.Categorical(["test","train","test","train"]),
             'F' : 'foo' }

data
```

```
{'A': 1.0, 'B': Timestamp('2013-01-02 00:00:00'), 'C': 0      1.0
 1      1.0
 2      1.0
 3      1.0
dtype: float32, 'D': array([3, 3, 3, 3]), 'E': [test, train, test, train]
Categories (2, object): [test, train], 'F': 'foo'}
```

- 연습삼아 상수, **timestamp**, **Series**, **numpy array**, 범수형, 문자열이 있는 **dict** 형 데이터를 하나 만들었다

```
In [17]: df2 = pd.DataFrame(data)  
df2
```

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

- **dict** 형을 이용해서 바로 **pandas DataFrame**을 만들 수 있다.
- **dict**의 각 **key**가 컬럼 이름이 된다

In [19]:

```
df.head()
```

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920

- **head()** 첫 번째 몇 개의 줄을 보여준다. 디폴트=5

In [20]:

```
df.tail()
```

	A	B	C	D
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-06	-0.851687	0.927910	0.516239	0.175822

- **tail()** 마지막 번째 몇 개의 줄을 보여준다. 디폴트=5

In [21]:

```
df.index
```

```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
                 '2013-01-05', '2013-01-06'],
                dtype='datetime64[ns]', freq='D')
```

In [22]:

```
df.columns
```

```
Index(['A', 'B', 'C', 'D'], dtype='object')
```

In [23]:

```
df.values
```

```
array([[ 0.37043452, -1.05025545, -1.28894456,  1.02020848],
       [-0.1482927 ,  0.02979441,  0.02602559, -0.79518354],
       [ 0.59924096,  0.60007872, -1.31437508,  2.10130875],
       [-0.02104932, -0.28079669, -2.69244809, -0.3166462 ],
       [ 0.3279692 , -2.41526199,  3.88957478,  0.58492005],
       [-0.8516873 ,  0.92791032,  0.51623919,  0.17582208]])
```

In [24]:

```
df.describe()
```

	A	B	C	D
count	6.000000	6.000000	6.000000	6.000000
mean	0.046103	-0.364755	-0.143988	0.461738
std	0.517480	1.219762	2.276392	1.027885
min	-0.851687	-2.415262	-2.692448	-0.795184
25%	-0.116482	-0.857891	-1.308017	-0.193529
50%	0.153460	-0.125501	-0.631459	0.380371
75%	0.359818	0.457508	0.393686	0.911386
max	0.599241	0.927910	3.889575	2.101309

- 통계적 개요를 보여준다

In [26]:

df.T

	2013-01-01 00:00:00	2013-01-02 00:00:00	2013-01-03 00:00:00	2013-01-04 00:00:00	2013-01-05 00:00:00	2013-01-06 00:00:00		
	A	B	C	D	A	B	C	D
A	0.370435	-0.148293	0.599241	-0.021049	6.000000	6.000000	6.000000	6.000000
B	-1.050255	0.029794	0.600079	-0.280797	0.046103	-0.364755	-0.143988	0.461738
C	-1.288945	0.026026	-1.314375	-2.692448	0.517480	1.219762	2.276392	1.027885
D	1.020208	-0.795184	2.101309	-0.316646	-0.851687	-2.415262	-2.692448	-0.795184
					-0.116482	-0.857891	-1.308017	-0.193529
					0.153460	-0.125501	-0.631459	0.380371
					0.359818	0.457508	0.393686	0.911386
					0.599241	0.927910	3.889575	2.101309

- 행과 열을 바꾸는 transpose 기능도 있다

Pandas 기초

In [28]:

```
df.sort_values(by='B')
```

	A	B	C	D
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-06	-0.851687	0.927910	0.516239	0.175822

In [29]:

```
df.sort_values(by='B', ascending=False)
```

	A	B	C	D
2013-01-06	-0.851687	0.927910	0.516239	0.175822
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-05	0.327969	-2.415262	3.889575	0.584920

Pandas 응용 - slice

Pandas - slice

In [25]:

df

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-06	-0.851687	0.927910	0.516239	0.175822

```
In [30]: df['A']
```

```
2013-01-01    0.370435
2013-01-02   -0.148293
2013-01-03    0.599241
2013-01-04   -0.021049
2013-01-05    0.327969
2013-01-06   -0.851687
Freq: D, Name: A, dtype: float64
```

- 특정 컬럼 선택하기

In [31]:

```
df[0:3]
```

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309

- 특정 행(index) 선택하기

```
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
                 '2013-01-05', '2013-01-06'],
                dtype='datetime64[ns]', freq='D')
```

In [32]:

```
df.loc[dates[0]]
```

```
A    0.370435
B   -1.050255
C   -1.288945
D    1.020208
Name: 2013-01-01 00:00:00, dtype: float64
```

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-06	-0.851687	0.927910	0.516239	0.175822

- 특정 행의 **index** 이름으로 찾기

In [33]:

```
df.loc[:,['A','B']]
```

	A	B
2013-01-01	0.370435	-1.050255
2013-01-02	-0.148293	0.029794
2013-01-03	0.599241	0.600079
2013-01-04	-0.021049	-0.280797
2013-01-05	0.327969	-2.415262
2013-01-06	-0.851687	0.927910

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-06	-0.851687	0.927910	0.516239	0.175822

- 행과 열의 범위를 이름으로 지정
- :만 사용하면 전체를 의미

In [34]:

```
df.loc['20130102':'20130104',['A','B']]
```

	A	B
2013-01-02	-0.148293	0.029794
2013-01-03	0.599241	0.600079
2013-01-04	-0.021049	-0.280797

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-06	-0.851687	0.927910	0.516239	0.175822

- 행과 열의 범위를 이름으로 지정

```
In [35]: df.loc['20130102',['A','B']]
```

```
A    -0.148293  
B     0.029794  
Name: 2013-01-02 00:00:00, dtype: float64
```

```
In [36]: df.loc[dates[0], 'A']
```

```
0.37043451811779354
```

In [37]:

df.iloc[3]

```
A    -0.021049  
B    -0.280797  
C    -2.692448  
D    -0.316646  
Name: 2013-01-04 00:00:00, dtype: float64
```

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-06	-0.851687	0.927910	0.516239	0.175822

- iloc는 행이나 열의 순서로 범위를 지정

In [38]:

```
df.iloc[3:5, 0:2]
```

	A	B
2013-01-04	-0.021049	-0.280797
2013-01-05	0.327969	-2.415262

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-06	-0.851687	0.927910	0.516239	0.175822

- 순서를 범위로 할 때는
 - $m:n$ 일 때, m 부터 $n-1$ 까지를 의미

Pandas - slice

In [39]:

```
df.iloc[[1,2,4],[0,2]]
```

	A	C
2013-01-02	-0.148293	0.026026
2013-01-03	0.599241	-1.314375
2013-01-05	0.327969	3.889575

In [40]:

```
df.iloc[1:3,:]
```

	A	B	C	D
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309

Pandas - slice

```
In [41]: df.iloc[:,1:3]
```

	B	C
2013-01-01	-1.050255	-1.288945
2013-01-02	0.029794	0.026026
2013-01-03	0.600079	-1.314375
2013-01-04	-0.280797	-2.692448
2013-01-05	-2.415262	3.889575
2013-01-06	0.927910	0.516239

```
In [42]: df.iloc[1,1]
```

0.029794412034582867

In [43]:

```
df[df['A'] > 0]
```

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-05	0.327969	-2.415262	3.889575	0.584920

	A	B	C	D
2013-01-01	0.370435	-1.050255	-1.288945	1.020208
2013-01-02	-0.148293	0.029794	0.026026	-0.795184
2013-01-03	0.599241	0.600079	-1.314375	2.101309
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646
2013-01-05	0.327969	-2.415262	3.889575	0.584920
2013-01-06	-0.851687	0.927910	0.516239	0.175822

- 조건문의 적용도 가능함

```
In [44]: df[df > 0]
```

	A	B	C	D
2013-01-01	0.370435	NaN	NaN	1.020208
2013-01-02	NaN	0.029794	0.026026	NaN
2013-01-03	0.599241	0.600079	NaN	2.101309
2013-01-04	NaN	NaN	NaN	NaN
2013-01-05	0.327969	NaN	3.889575	0.584920
2013-01-06	NaN	0.927910	0.516239	0.175822

- 이런 조건문도 가능

```
In [47]:  
a = [1,2,3]  
b = a  
b[1] = 0  
b
```

```
[1, 0, 3]
```

```
In [48]:  
a
```

```
[1, 0, 3]
```

- Python에서의 `copy`는 참조 복사로 조심해야 한다

```
In [49]:  
c = [1,2,3]  
d = c.copy()  
d[1] = 0  
d
```

```
[1, 0, 3]
```

```
In [50]:
```

```
c
```

```
[1, 2, 3]
```

- 이럴 때는 **copy()** 옵션으로 복사한다

Pandas - slice

```
In [45]: df2 = df.copy()
```

```
In [46]: df2['E'] = ['one', 'one', 'two', 'three', 'four', 'three']
df2
```

	A	B	C	D	E
2013-01-01	0.370435	-1.050255	-1.288945	1.020208	one
2013-01-02	-0.148293	0.029794	0.026026	-0.795184	one
2013-01-03	0.599241	0.600079	-1.314375	2.101309	two
2013-01-04	-0.021049	-0.280797	-2.692448	-0.316646	three
2013-01-05	0.327969	-2.415262	3.889575	0.584920	four
2013-01-06	-0.851687	0.927910	0.516239	0.175822	three

Pandas 응용 - 변경

In [51]:

```
data = {'a':[1,2,3,4], 'b':[0,1,0.1,0.2], 'sub':['1st', '2nd', '3rd', '4th']}
```

```
data
```

```
{'a': [1, 2, 3, 4], 'b': [0, 1, 0.1, 0.2], 'sub': ['1st', '2nd', '3rd', '4th']}
```

In [52]:

```
df3 = pd.DataFrame(data)
```

```
df3
```

	a	b	sub
0	1	0.0	1st
1	2	1.0	2nd
2	3	0.1	3rd
3	4	0.2	4th

- 컬럼 순서를 변경하고 싶다면

In [53]:

```
df3 = pd.DataFrame(data, columns=[ 'sub' , 'a' , 'b' ] )  
df3
```

	sub	a	b
0	1st	1	0.0
1	2nd	2	1.0
2	3rd	3	0.1
3	4th	4	0.2

- 인덱스의 순서를 변경하고 싶다면

```
In [54]: df3 = df3.reindex(index=[0,2,1,3])
df3
```

	sub	a	b
0	1st	1	0.0
2	3rd	3	0.1
1	2nd	2	1.0
3	4th	4	0.2

```
In [55]: df3['c'] = np.nan  
df3
```

	sub	a	b	c
0	1st	1	0.0	NaN
2	3rd	3	0.1	NaN
1	2nd	2	1.0	NaN
3	4th	4	0.2	NaN

```
In [56]: df3.loc[1,['c']] = 2  
df3
```

	sub	a	b	c
0	1st	1	0.0	NaN
2	3rd	3	0.1	NaN
1	2nd	2	1.0	2.0
3	4th	4	0.2	NaN

```
In [57]: df3['d']= df3['a'] + df3['b']
df3
```

	sub	a	b	c	d
0	1st	1	0.0	NaN	1.0
2	3rd	3	0.1	NaN	3.1
1	2nd	2	1.0	2.0	3.0
3	4th	4	0.2	NaN	4.2

- 일반적인 **low level language** 유저라면 꽤 놀래야 하는데.

```
In [58]: df3.loc[4] = np.nan  
df3
```

	sub	a	b	c	d
0	1st	1.0	0.0	NaN	1.0
2	3rd	3.0	0.1	NaN	3.1
1	2nd	2.0	1.0	2.0	3.0
3	4th	4.0	0.2	NaN	4.2
4	NaN	NaN	NaN	NaN	NaN

```
In [59]: df3.loc[4] = ['5th', 'a', 'b', 'c', 'd']  
df3
```

	sub	a	b	c	d
0	1st	1	0	NaN	1
2	3rd	3	0.1	NaN	3.1
1	2nd	2	1	2	3
3	4th	4	0.2	NaN	4.2
4	5th	a	b	c	d

- 행 추가

재미있는거 한 번 해볼까? - 네이버 얼굴 익식

API 이용 안내

Clova

Clova Platform

Clova A.I. APIs

- Clova Face Recognition

네이버 아이디로 로그인

지도

파파고

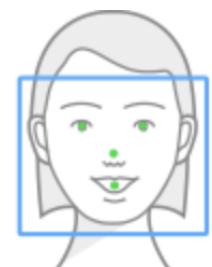
서비스 API

Clova Face Recognition (CFR)

N 트윗 공유하기 246개

얼굴과 관련된 다양한 정보를 제공하는 얼굴 인식 API

입력된 비전 데이터를 통해 얼굴을 인식하거나 얼굴 감지를 이용한 애플리케이션을 만들 때 유용한 API입니다.
이미지 속의 얼굴과 가장 닮은 유명인을 찾거나, 얼굴의 윤곽과 눈/코/입 위치, 표정 값을 얻을 수 있습니다.



유명인 얼굴 인식 및 정확한 얼굴 감지 기능

네이버가 보유한 대량의 이미지 DB를 통해 국내에서 가장 뛰어난 유명인 얼굴 인식과 정확한 얼굴 감지 기능을 제공합니다. 두 가지 기능을 통해 비전(Vision) 정보를 새로운 형태의 인터페이스로 사용할 수 있습니다.

손쉬운 사용

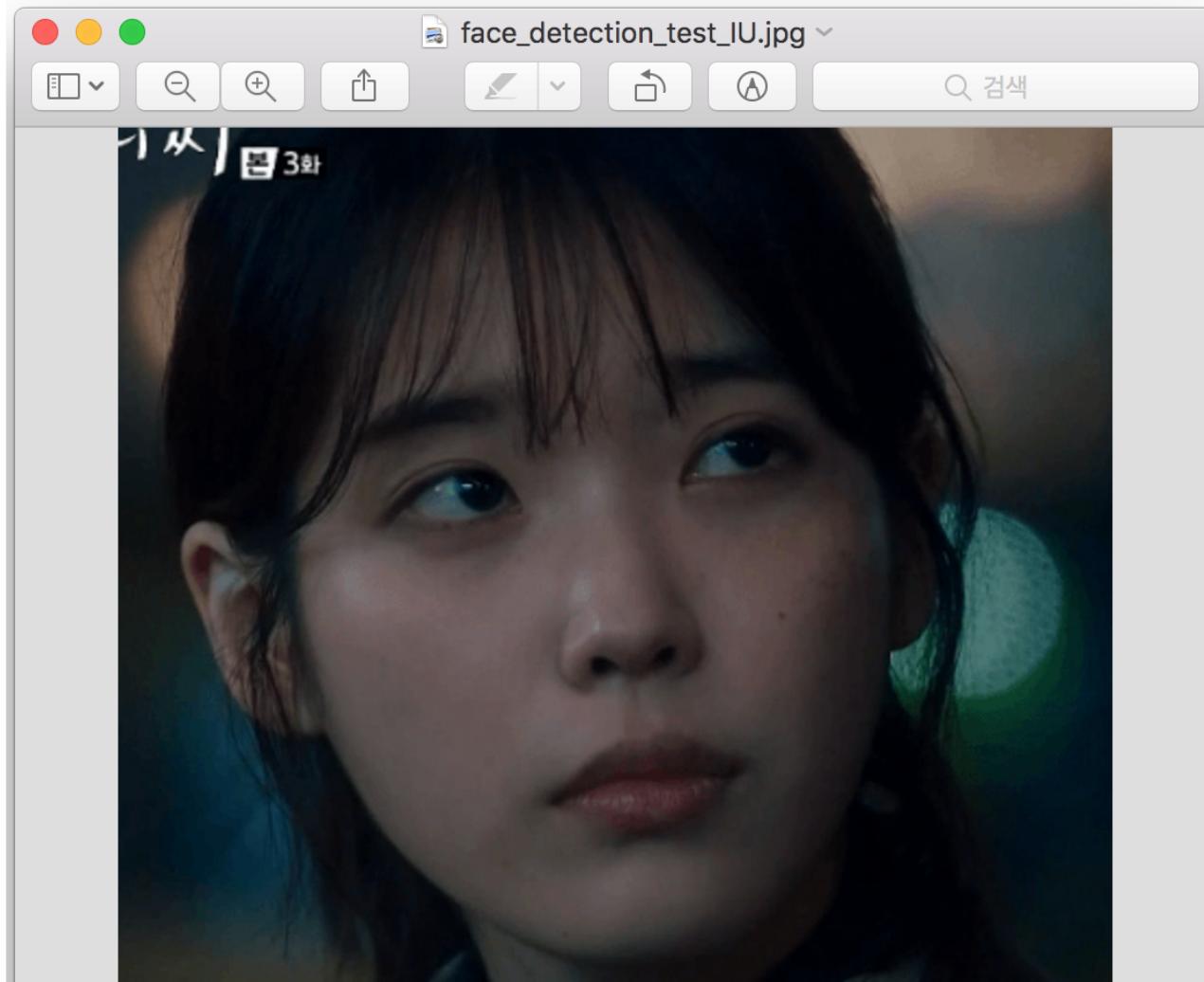
웹 기반의 콘솔을 통해 쉽게 사용할 수 있습니다. RESTful 형태로 지원되는 API로 고객의 서비스에 얼굴 인식 및 얼굴 감지 기능을 간단히 적용할 수 있어 편리합니다.

- * Clova의 인공지능 기술이 적용되었습니다.
- * 유명인 얼굴인식은 모든 유명인에 대한 인식을 보장하지는 않습니다.
- * 이미지 파일 사이즈는 2M 이하로 제한합니다.
- * 처리한도 : 1,000건 /일

[오픈 API 이용 신청](#)

[개발 가이드 보기](#)

- 여기서 새로 신청해서 얼굴 인식을 위한 키를 다시 받아야 합니다.~



- 나의 아저씨와 같은 **well-made** 드라마가 또 언제 나타날지... 이지안 - 전화해

In [30]:

```
import requests
client_id = "7X7chg7mtLCi7eK0uAuS"
client_secret = "tbhFBAh00"
url = "https://openapi.naver.com/v1/vision/celebrity" # 유명인 얼굴인식
files = {'image': open('../data/face_detection_test_IU.jpg', 'rb')}
headers = {'X-Naver-Client-Id': client_id, 'X-Naver-Client-Secret': client_secret}
response = requests.post(url, files=files, headers=headers)
rescode = response.status_code
if(rescode==200):
    print (response.text)
else:
    print("Error Code:" + rescode)
```

```
{"info": {"size": {"width": 914, "height": 836}, "faceCount": 1}, "faces": [{"celebrity": {"value": "아이유", "confidence": 0.486239}}]}
```

- 별다를 것 없다. **URL** 던지고, 사진도 던지면 된다.
- 아이유라고 알아봐 준다... (48.6%)



- 죄송합니다. ㅠㅠ.

```
In [31]: url = "https://openapi.naver.com/v1/vision/celebrity" # 유명인 얼굴인식  
files = {'image': open('../data/face_detection_test_PW.jpg', 'rb')}  
headers = {'X-Naver-Client-Id': client_id, 'X-Naver-Client-Secret': client_secret }  
response = requests.post(url, files=files, headers=headers)  
rescode = response.status_code  
if(rescode==200):  
    print (response.text)  
else:  
    print("Error Code:" + rescode)
```

```
{"info": {"size": {"width": 748, "height": 1034}, "faceCount": 1}, "faces": [{"celebrity": {"value": "윤민수", "confidence": 0.108725}}]}
```



- 후야 ~~~ 미안....

코드를 설명하기 보단 흐름을 한 번 봐주세요 ~

In [32]:

```
url = "https://openapi.naver.com/v1/vision/face" # 얼굴감지
files = {'image': open('../data/face_detection_test_IU.jpg', 'rb')}
headers = {'X-Naver-Client-Id': client_id, 'X-Naver-Client-Secret': client_secret }
response = requests.post(url, files=files, headers=headers)
rescode = response.status_code
if(rescode==200):
    print (response.text)
else:
    print("Error Code:" + rescode)
```

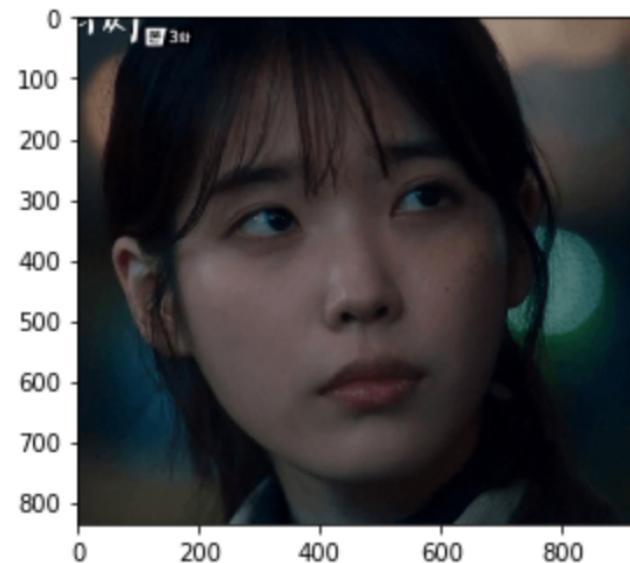
```
{"info": {"size": {"width": 914, "height": 836}, "faceCount": 1}, "faces": [{"roi": {"x": 193, "y": 192, "width": 518, "height": 518}, "landmark": {"leftEye": {"x": 320, "y": 323}, "rightEye": {"x": 574, "y": 288}, "nose": {"x": 459, "y": 453}, "leftMouth": {"x": 392, "y": 620}, "rightMouth": {"x": 571, "y": 595}}, "gender": {"value": "female", "confidence": 0.996178}, "age": {"value": "17~21", "confidence": 0.469397}, "emotion": {"value": "neutral", "confidence": 0.999103}, "pose": {"value": "frontal_face", "confidence": 0.999678}}]}
```

In [33]:

```
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg  
import matplotlib.patches as patches  
import numpy as np  
%matplotlib inline
```

In [34]:

```
img = mpimg.imread('../data/face_detection_test_IU.jpg')  
imgplot = plt.imshow(img)
```



In [35]:

```
import json  
detect_result = json.loads(response.text)  
detect_result
```

```
{'info': {'size': {'width': 914, 'height': 836}, 'faceCount': 1},  
 'faces': [{['roi']: {'x': 193, 'y': 192, 'width': 518, 'height': 518},  
            'landmark': {'leftEye': {'x': 320, 'y': 323},  
                         'rightEye': {'x': 574, 'y': 288},  
                         'nose': {'x': 459, 'y': 453},  
                         'leftMouth': {'x': 392, 'y': 620},  
                         'rightMouth': {'x': 571, 'y': 595}},  
            'gender': {'value': 'female', 'confidence': 0.996178},  
            'age': {'value': '17~21', 'confidence': 0.469397},  
            'emotion': {'value': 'neutral', 'confidence': 0.999103},  
            'pose': {'value': 'frontal_face', 'confidence': 0.999678}]}}
```

In [36]:

```
detect_summary = detect_result['faces'][0]
detect_summary.keys()

dict_keys(['roi', 'landmark', 'gender', 'age', 'emotion', 'pose'])
```

In [37]:

```
detect_summary['roi']

{'x': 193, 'y': 192, 'width': 518, 'height': 518}
```

In [38]:

```
x, y, w, h = detect_summary['roi'].values()  
x, y, w, h
```

(193, 192, 518, 518)

In [39]:

```
gender, gen_confidence = detect_summary['gender'].values()  
gender, gen_confidence
```

('female', 0.996178)

In [40]:

```
emotion, emotion_confidence = detect_summary['emotion'].values()
emotion, emotion_confidence
```

('neutral', 0.999103)

In [41]:

```
age, age_confidence = detect_summary['age'].values()
age, age_confidence
```

('17~21', 0.469397)

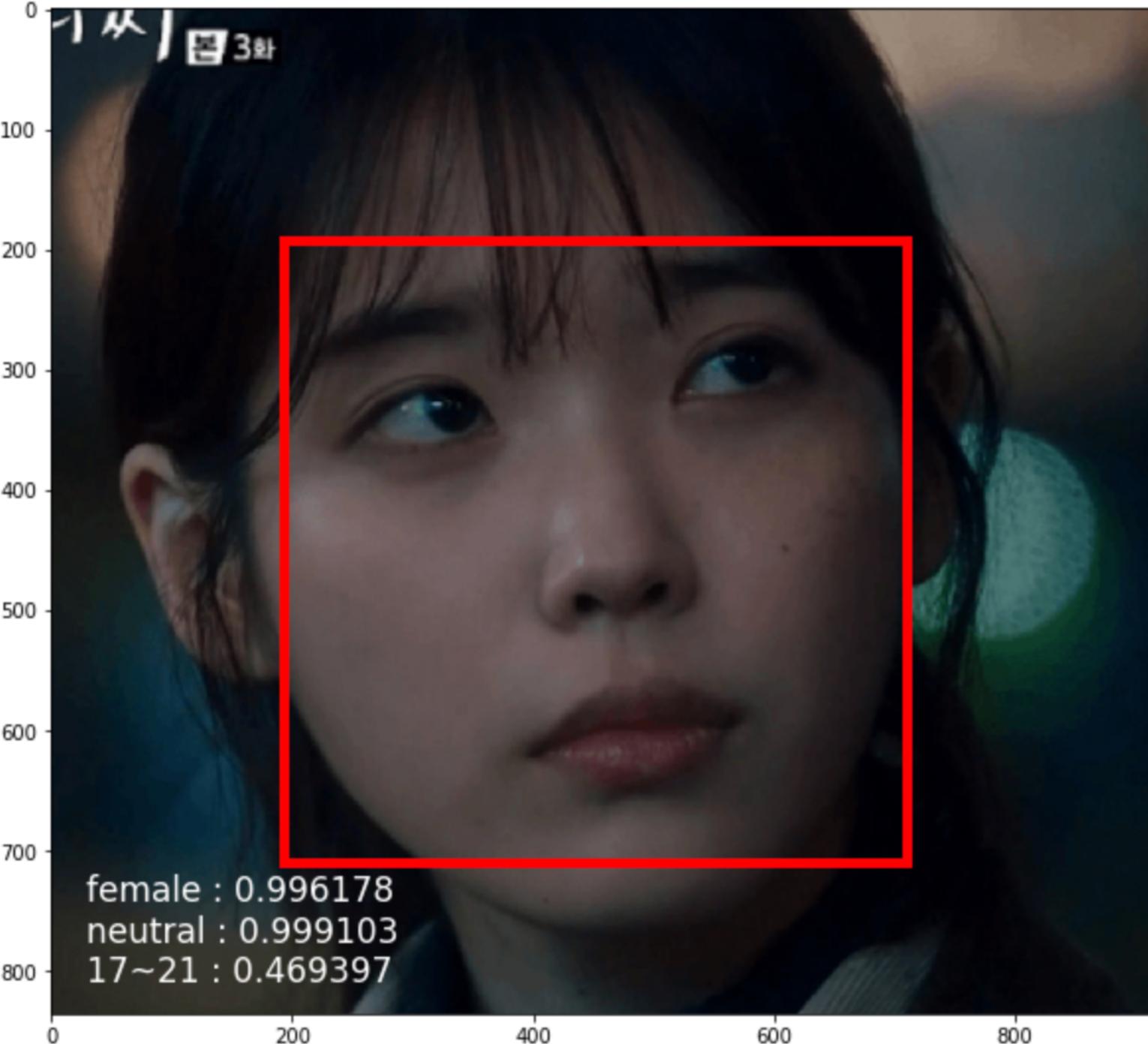
In [42]:

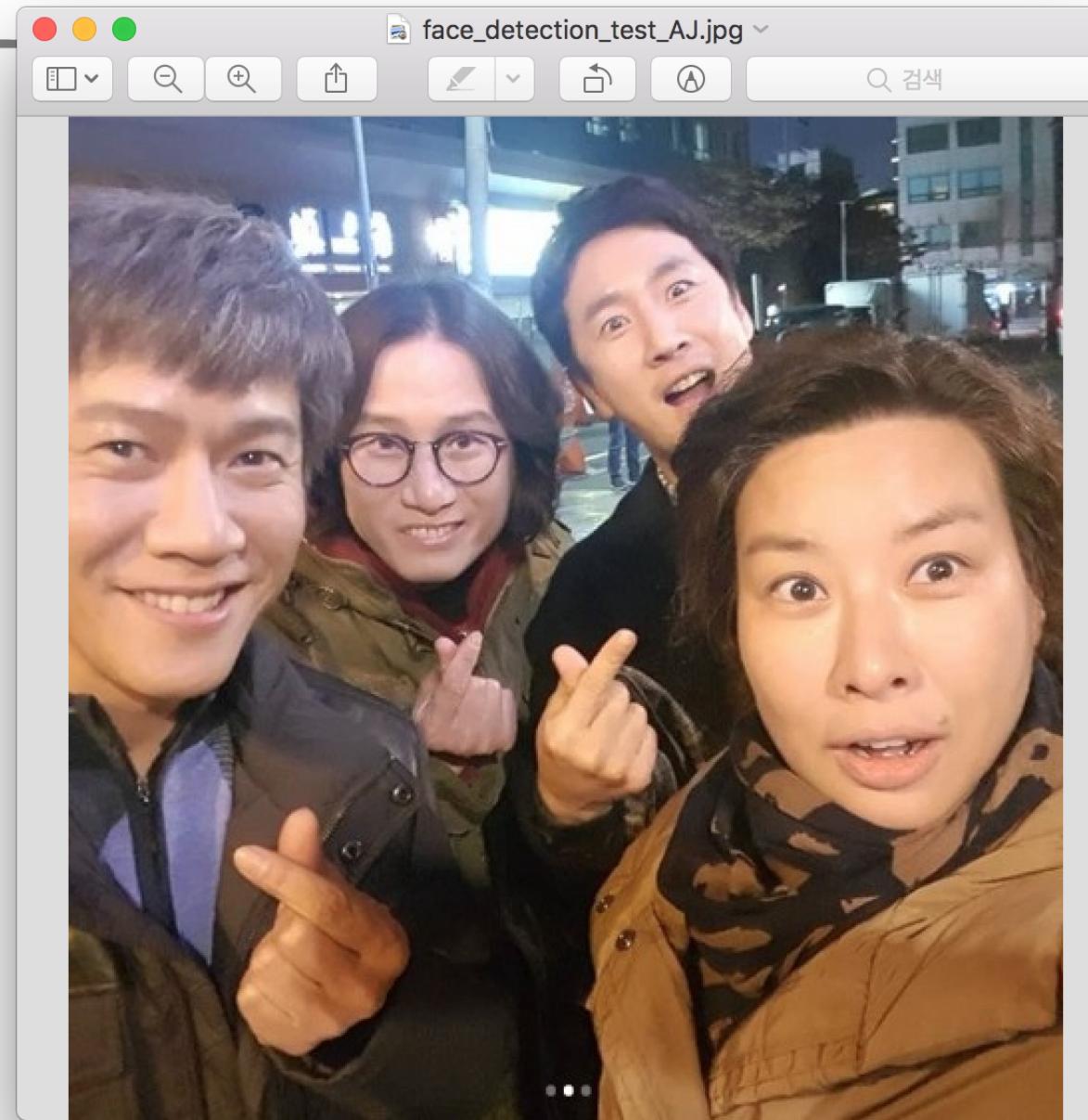
```
x, y, w, h = detect_summary['roi'].values()
gender, gen_confidence = detect_summary['gender'].values()
emotion, emotion_confidence = detect_summary['emotion'].values()
age, age_confidence = detect_summary['age'].values()
```

In [43]:

```
fig,ax = plt.subplots(figsize=(10,10))
ax.imshow(img)
rect_face = patches.Rectangle((x,y),w,h,
                             linewidth=5,
                             edgecolor='r',
                             facecolor='none')
ax.add_patch(rect_face)

annotation = gender + ' : ' + str(gen_confidence) + \
             '\n' + emotion + ' : ' + str(emotion_confidence) + \
             '\n' + age + ' : ' + str(age_confidence)
plt.figtext(0.15, 0.17, annotation, wrap=True, fontsize=17, color='white')
plt.show()
```





In [44]:

```
url = "https://openapi.naver.com/v1/vision/face" # 얼굴감지
files = {'image': open('../data/face_detection_test_AJ.jpg', 'rb')}
headers = {'X-Naver-Client-Id': client_id, 'X-Naver-Client-Secret': client_secret }
response = requests.post(url, files=files, headers=headers)
rescode = response.status_code
if(rescode==200):
    print (response.text)
else:
    print("Error Code:" + rescode)

onfidence":0.545355},"emotion": {"value": "surprise", "confidence": 0.996858}, "pose": {"valu
e": "frontal_face", "confidence": 0.999743}}, {"roi": {"x": 140, "y": 143, "width": 82, "height": 8
2}, "landmark": {"leftEye": {"x": 159, "y": 163}, "rightEye": {"x": 198, "y": 162}, "nose": {"x": 18
1, "y": 191}, "leftMouth": {"x": 162, "y": 201}, "rightMouth": {"x": 200, "y": 200}}, "gender": {"val
ue": "female", "confidence": 0.99999}, "age": {"value": "32~36", "confidence": 0.617098}, "emoti
on": {"value": "smile", "confidence": 0.99748}, "pose": {"value": "frontal_face", "confidence": 0.999743}}
```

In [45]:

```
detect_result = json.loads(response.text)  
detect_result
```

```
{'info': {'size': {'width': 500, 'height': 504}, 'faceCount': 4},  
'faces': [{ 'roi': {'x': 335, 'y': 195, 'width': 138, 'height': 138},  
    'landmark': {'leftEye': {'x': 366, 'y': 235},  
        'rightEye': {'x': 437, 'y': 225},  
        'nose': {'x': 402, 'y': 277},  
        'leftMouth': {'x': 382, 'y': 314},  
        'rightMouth': {'x': 437, 'y': 311}},  
    'gender': {'value': 'female', 'confidence': 0.999764},  
    'age': {'value': '36~40', 'confidence': 0.545355},  
    'emotion': {'value': 'surprise', 'confidence': 0.996858},
```

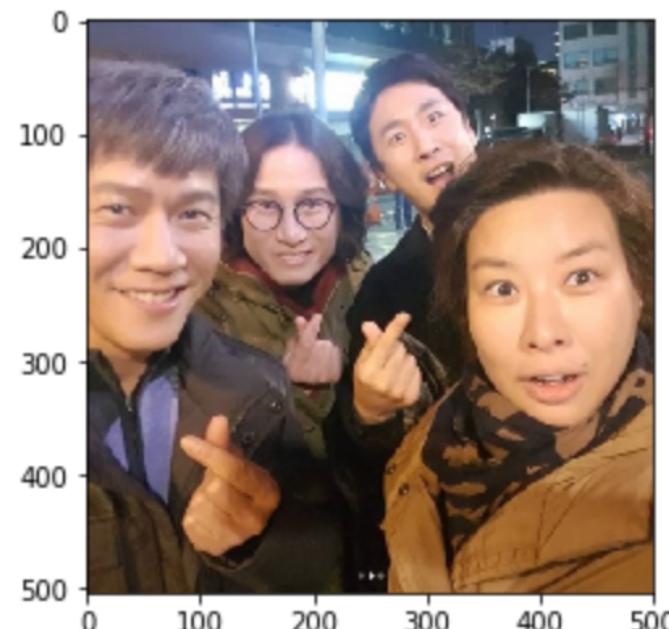
In [46]:

```
detect_result['faces']
```

```
[{'roi': {'x': 335, 'y': 195, 'width': 138, 'height': 138},  
 'landmark': {'leftEye': {'x': 366, 'y': 235},  
 'rightEye': {'x': 437, 'y': 225},  
 'nose': {'x': 402, 'y': 277},  
 'leftMouth': {'x': 382, 'y': 314},  
 'rightMouth': {'x': 437, 'y': 311}},  
 {'gender': {'value': 'female', 'confidence': 0.999764},  
 'age': {'value': '36~40', 'confidence': 0.545355},  
 'emotion': {'value': 'surprise', 'confidence': 0.996858},  
 'pose': {'value': 'frontal_face', 'confidence': 0.999743}},  
 {'roi': {'x': 140, 'y': 143, 'width': 82, 'height': 82},  
 'landmark': {'leftEye': {'x': 159, 'y': 163},  
 'rightEye': {'x': 181, 'y': 162}}]
```

In [47]:

```
img = mpimg.imread('..../data/face_detection_test_AJ.jpg')  
imgplot = plt.imshow(img)
```



In [48]:

```
fig,ax = plt.subplots(figsize=(10,10))
ax.imshow(img)

for each in detect_result['faces']:
    x, y, w, h = each['roi'].values()
    gender, gen_confidence = each['gender'].values()
    emotion, emotion_confidence = each['emotion'].values()
    age, age_confidence = each['age'].values()

    rect_face = patches.Rectangle((x,y),w,h, linewidth=5,
                                  edgecolor='r', facecolor='none')
    ant_letter = gender[0] + ' , ' + emotion + ' , ' + age
    plt.text(x,y-10, ant_letter, size=16, color='red')
    ax.add_patch(rect_face)

plt.show()
```

