

## Chapter 03. 데이터 수집, 분석과 시각화

---

### 01. 페이스북 API를 활용한 빈도분석

---

### 02. 공공 데이터 API를 활용한 데이터 기반 추천

---

### 03. 웹서비스 데이터를 활용한 지리정보 기반 시각화

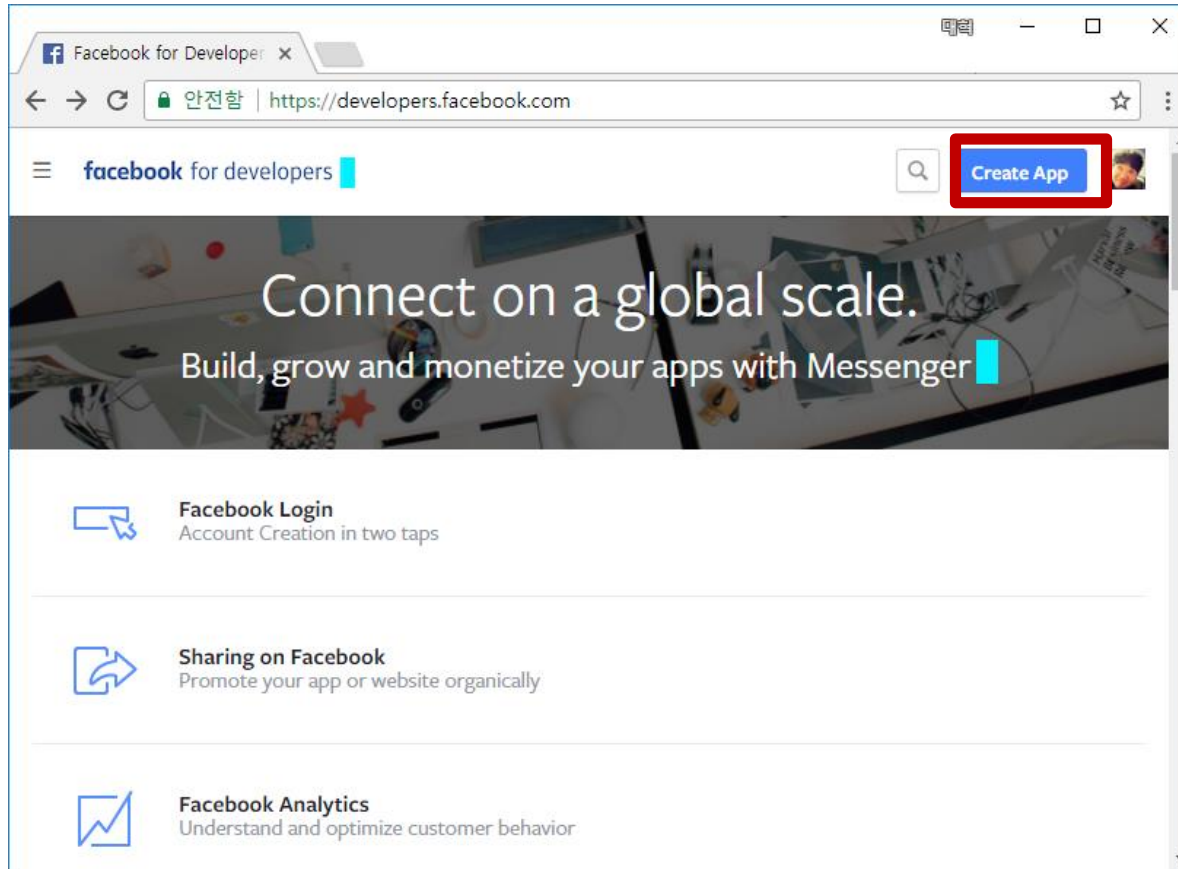
---

# 1. 페이스북 API를 활용한 빈도분석

## 1.1 페이스북 API 사용하기

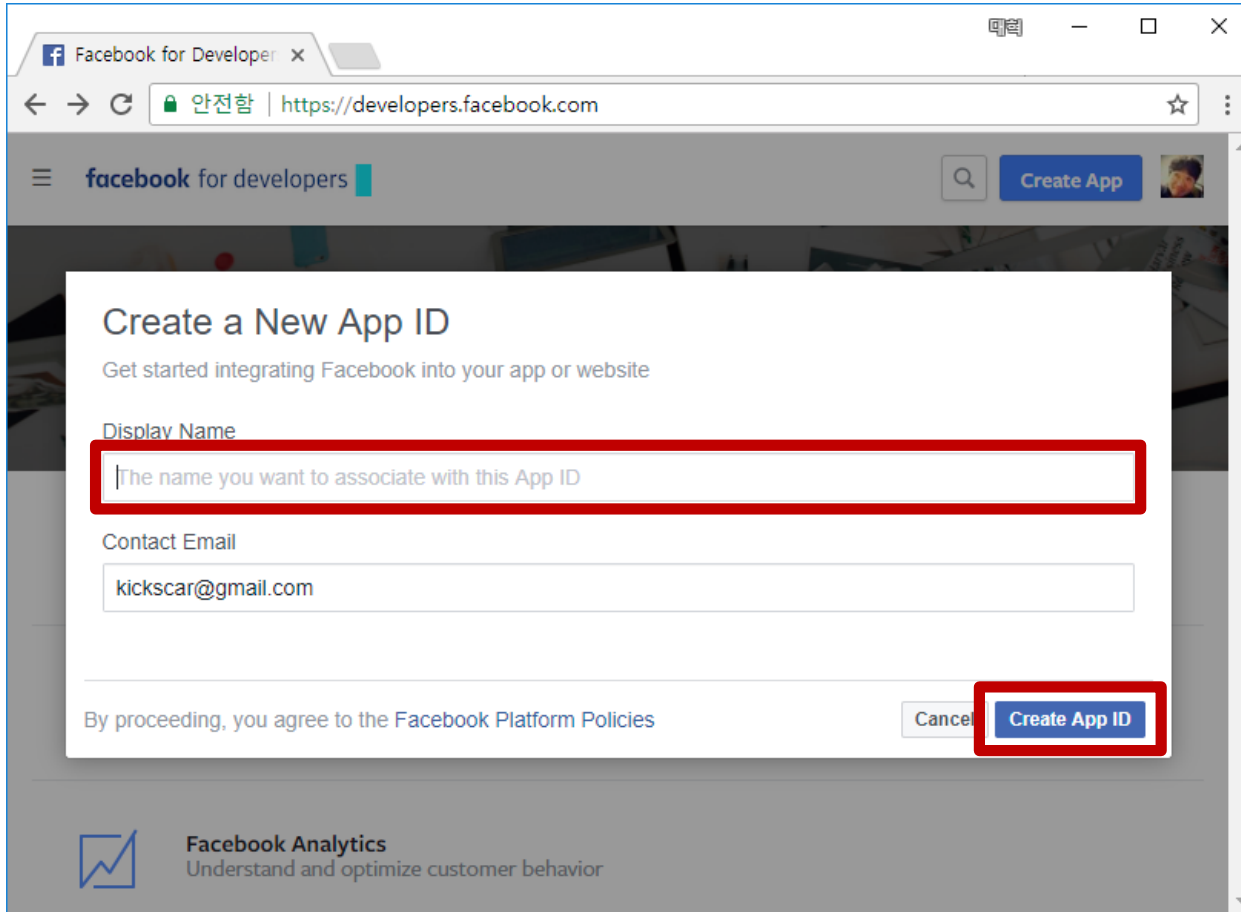
### 1.1.1 페이스북 앱ID 등록

1) 개발자 페이지 [<https://developers.facebook.com>]



# 1. 페이스북 API를 활용한 빈도분석

## 2) 새로운 앱 만들기



Facebook for Developer x

← → ↺ | 안전함 | https://developers.facebook.com

facebook for developers

Create App

### Create a New App ID

Get started integrating Facebook into your app or website

Display Name

The name you want to associate with this App ID

Contact Email

kickscar@gmail.com

By proceeding, you agree to the Facebook Platform Policies

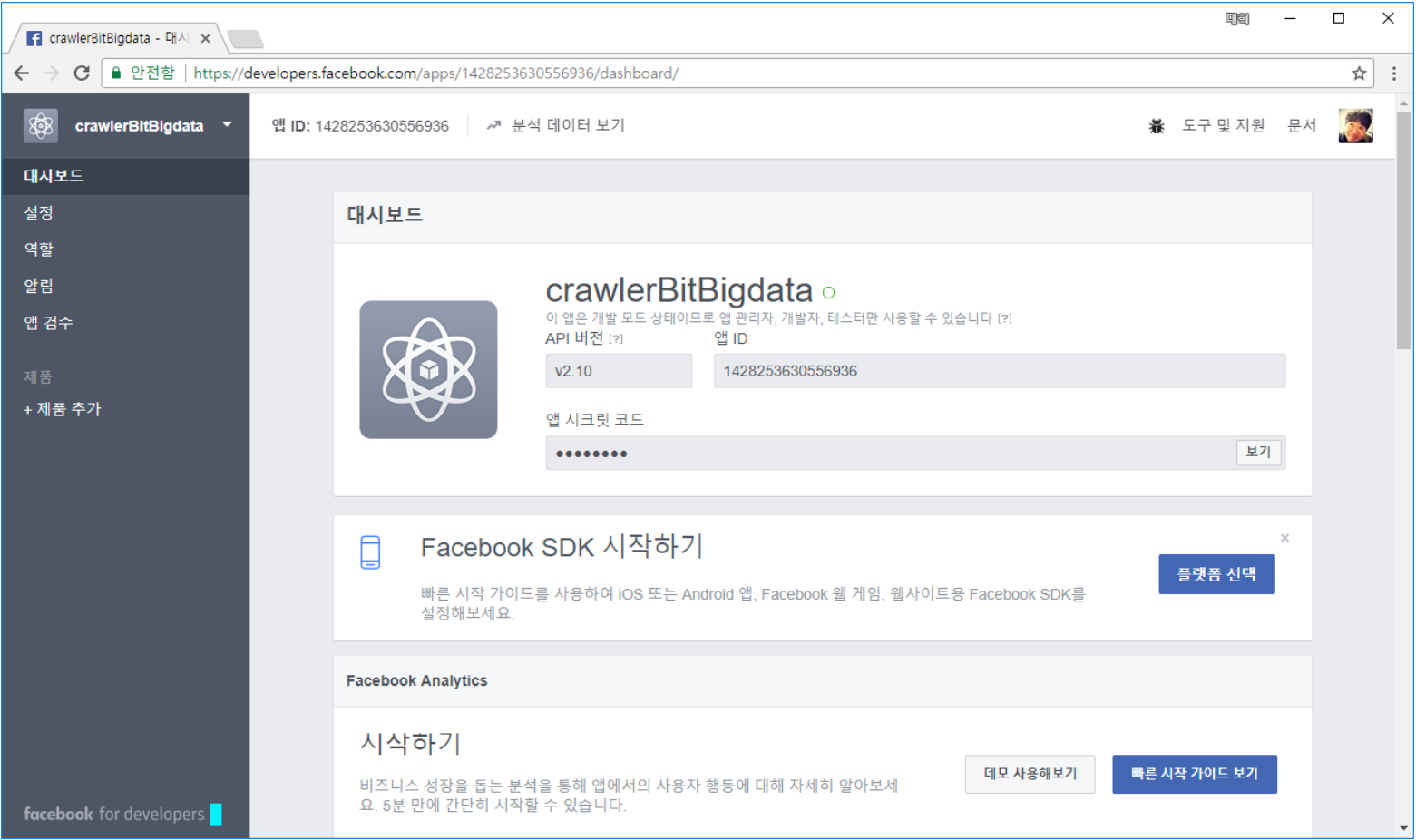
Cancel Create App ID

Facebook Analytics  
Understand and optimize customer behavior

앱의 고유이름으로 다른 사람이 사용하지 않는 고유 이름을 입력한다.

# 1. 페이스북 API를 활용한 빈도분석

## 3) 앱(crawlerBitBigdata) 관리 페이지 - 대시보드



단순하게 페이스북 페이지를 크롤링 하기 때문에 별도의 제품을 추가할 필요가 없다.

# 1. 페이스북 API를 활용한 빈도분석

## 1.2 페이스북 데이터 수집하기

### 1.2.1 JSON 통신 모듈 만들기

1) urllib.request 모듈을 이용한 HTTP 통신

[test\_http.py]

```
import sys
from urllib.request import Request, urlopen
from datetime import *

try:
    url = 'http://localhost:8088/mysite03'

    resp = urlopen(Request(url))
    resp_body = resp.read().decode('utf-8')
    print(resp_body)
except Exception as e:
    print("%s : %s" % (e, datetime.now()), file=sys.stderr)
```

에러가 발생할 경우도 테스트 해보자.

# 1. 페이스북 API를 활용한 빈도분석

2) json 모듈을 이용한 json 데이터 통신  
[test\_json.py]

```
import sys
from urllib.request import Request, urlopen
from datetime import *
import json
try:
    url = 'http://localhost:8088/mysite03/guestbook/api/list'

    resp = urlopen(Request(url))
    resp_body = resp.read().decode('utf-8')
    json_result = json.loads(resp_body)

    print(json_result)
except Exception as e:
    print("%s : %s" % (e, datetime.now()), file=sys.stderr)

{'result': 'success', 'message': None, 'data': [{'no': 12, 'name': '안대혁', 'password': None, 'message': 'ㅎㅇ!', 'regDate': '2017-07-03'}, {'no': 11, 'name': '마이콜', 'password': None, 'message': 'ㅎㅇ!', 'regDate': '2017-07-03'}, {'no': 10, 'name': '마이콜', 'password': None, 'message': 'ㅎㅇ!', 'regDate': '2017-07-03'}, ... ]}
```

# 1. 페이스북 API를 활용한 빈도분석

## 3) json\_request 모듈 작성하고 테스트 하기

[json\_result.py]

```
def json_request_error(e):  
    print("%s : %s" % (e, datetime.now()), file=sys.stderr)  
  
def json_request(url="", encoding='utf-8', success=None, error=json_request_error):  
    try:  
        resp = urlopen(Request(url))  
        if resp.getcode() == 200:  
            resp_body = resp.read().decode(encoding)  
            resp_json = json.loads(resp_body)  
            if callable(success) is False:  
                return resp_json  
            success(resp_json)  
    except Exception as e:  
        callable(error) and error("%s %s" % (str(e), url))
```

- 키워드 파라미터와 파라미터에 디폴트 값 세팅을 해서 함수 사용에 편리함과 간결성을 준다.
- 비동기 처리와 유사하게 JSON 응답 처리와 통신 중 에러 처리는 외부 함수로 전달 가능하게 해서 재사용성을 높인다.
- 응답 처리함수를 전달하지 않으면 JSON응답(dict 타입)을 반환 하도록 한다.
- 기본 에러 처리 함수를 제공한다.

# 1. 페이스북 API를 활용한 빈도분석

## 1.2.2 페이스북 Graph API 란?

1) 소셜 그래프라는 명칭에서 유래

2) 다음 항목으로 분류하여 해당 데이터를 가져올 수 있다.

노드(Node) : 기본적으로 사용자, 사진, 페이지, 댓글과 같은 항목

에지(Edge) : 페이지의 사진, 사진의 댓글 등 항목 간 연결 링크

필드(Field) : 생일, 페이지의 이름 등의 실제 항목에 대한 정보

3) 소셜 그래프에서 데이터를 가져오거나 게시하는 HTTP 기반의 API

4) 노드와 에지에 관하여 HTTP GET 요청을 통해 항목 데이터를 가져온다.

5) 대부분의 그래프 요청은 액세스 토큰을 사용해야 한다.



# 1. 페이스북 API를 활용한 빈도분석

## 1.2.3 페이스북 특정 페이지의 포스트 크롤링하기

### 1) 액세스 토큰

개발자 페이지에 등록했던 앱의 아이디와 등록과 함께 발급받은 앱 시크릿 코드의 조합

**access token => app's id | app's secret code**

```
ACCESS_TOKEN = '%s|%s' % ("1428253630556936", "ba8f09769d565fc8646398885bdb20ae")
```

#### 대시보드



**crawlerBitBigdata** ○

이 앱은 개발 모드 상태이므로 앱 관리자, 개발자, 테스터만 사용할 수 있습니다 [?]

API 버전 [?]

v2.10

앱 ID

1428253630556936

앱 시크릿 코드

ba8f09769d565fc8646398885bdb20ae

재설정

# 1. 페이스북 API를 활용한 빈도분석

## 2) 그래프 API URL

페이스 API 요청 url은 다음과 같은 형식을 가진다.

```
https://graph.facebook.com/v2.8/[Node, Edge]?parameters
```

보통, `https://graph.facebook.com/v2.8` 가 기본(base)가 되는 url이며,

다음 해당 Node, Edge 가 붙는다.

마지막으로 요청 Node, Edge에 대한 파라미터가 Query String 형태로 붙게 된다.

페이스북 API 요청에 대한 헬퍼 함수들을 작성해 보자.

[api\_fb.py]

```
BASE_URL_FB_API = 'https://graph.facebook.com/v2.8'
ACCESS_TOKEN = '%s|%s' % ("1428253630556936", "ba8f09769d565fc8646398885bdb20ae")

def fb_gen_url(base=BASE_URL_FB_API, node="", **param):
    return '%s/%s/?%s' % (base, node, urlencode(param))
```

노드와 쿼리 스트링은 API에 따라 달라지기 때문에 함수 파라미터로 받아 API 요청 url을 만들어 낸다.

base 와 node를 뺀 나머지 파라미터는 이름=값 형식으로 여러 파라미터를 dict으로 받아 `urlencode()` 함수를 통해 쿼리 스트링으로 만든다.

# 1. 페이스북 API를 활용한 빈도분석

[test\_api\_fb.py]

다음과 같은 조건으로 앞의 fb\_gen\_url() 함수를 테스트 해보자

**node**

jtbcnews/posts (jtbcnews 페이지의 포스트들을 가져오는 노드이다)

**parameter**

fields='id,message,link,name,type,shares,reactions,created\_time,comments.limit(0).summary(true).limit(0).summary(true)'

since=2016-01-01

until=2017-04-05

limit=50

access\_token=...

## 참고

**fields** : id(포스트 ID), comments(댓글정보), created\_time(생성일), from(포스트한 사용자 프로필), link(링크 정보), message(포스트 내용), name(링크의 이름), object\_id(업로드한 사진 , 동영상 id), parent\_id(부모 포스트), picture(포함된 사진들의 링크), place(포스트한 위치), reactions(리액션 정보), shares(공유한 숫자), type(포스트의 형식{link, status, photo, video, offer}), updated\_time(최종 수정일)

**since, until** : 페이스북에서는 유닉스 타임 스탬프, 'YYYY-MM-DD' 형식의 날짜 포맷 스트링

**limit** : 한 번 요청에 가져올 포스트 수

# 1. 페이스북 API를 활용한 빈도분석

[crawler\_fb.py]

```
from api_fb import *

pagename = 'jtbcnews'
since = '2016-01-01'
until = '2017-04-31'

url = fb_gen_url(
    node=pageid + '/posts',
    fields='id,message,link,name,type,shares,,created_time,\
reactions.limit(0).summary(true),\
comments.limit(0).summary(true)',
    since=since,
    until=until,
    limit=50,
    access_token=ACCESS_TOKEN)

print(url)
```

# 1. 페이스북 API를 활용한 빈도분석

주의 할 것은 노드중 pagename 이다.

페이스북 서비스 페이지에서는 페이지 이름이 URL에 쓰이지만 API URL에서는 페이지 이름 대신 페이지 아이디를 사용해야 한다.  
API를 이용해서 페이지 이름을 페이지 아이디로 바꿔보자.

[api\_fb.py]

```
def fb_name_to_id(pagename):  
    url = fb_gen_url(node=pagename, access_token=ACCESS_TOKEN)  
    json_result = json_request(url)  
    return json_result.get('id')
```

테스트 코드를 다음과 같이 바꾸고 테스트 해보자

```
url = fb_gen_url( node=fb_name_toid(pagename) + '/posts', fields='.....
```

# 1. 페이스북 API를 활용한 빈도분석

## 3) 포스트 가져오기 작성

특정 페이지의 특정 기간내의 포스트 가져오는 함수를 작성해 보자

[api\_fb.py]

```
def fb_fetch_posts(pagename, since, until):

    pageid = fb_name_to_id(pagename)

    url = fb_gen_url(
        node=pageid + '/posts',
        fields='id,message,link,name,type,shares,created_time,\
reactions.limit(0).summary(true),\
comments.limit(0).summary(true)',
        since=since,
        until=until,
        limit=LIMITS_REQUEST,
        access_token=ACCESS_TOKEN)

    json_result = json_request(url=url)
    print(json_result )
```

## 1. 페이스북 API를 활용한 빈도분석

## JSON 응답 객체 분석

```
String parse: 3629 errors
JS error fails

[{"data": [
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { },
    { }
],
'paging': {
  'cursors': {
    'before': "Q2c4UlpXNTBYM0YxWlhKNVgzTjBiM0o1WDJsaOR5TXLOREF5TmpNME1ESTJPVGs1TVRnNk5EazNName6TmprME56Y3pNREl3T0Rrek18OE1ZAMEJwDNOMGIZsjVYYmxrRHlBeU5EQXL0ak0wTURJMk9UazVNVGhmTVRVmJUSUVXpOVFl3TVRFM0lqWTROUTHFZAEdsdFpRWLoxc2VBQVE9PQZDZ",
    'after': "Q2c4UlpXNTBYM0YxWlhKNVgzTjBiM0o1WDJsaORSUXLOREF5TmpNME1ESTJPVGs1TVRnNkxUMTRNVEEyTmpZAek5qQXdNREV6TURVeU5qSVBER0ZA3YVY5emRH0XllVjlwMKE4ZA0lqUXdNa1l6tkRBUE5qazVPVEU0HpfFMUSUTTRNVGMwTmprRMK56YzRNeklQQkhScGjXVudxZAFE1bUFFFPQZDZ"
  },
  'next': "https://graph.facebook.com/v2.10/240263402659918/posts?access_token=1428253630556936%7Cba8f09769d565fc8646398885bdb20ae&fields=id%2Cmessage%2Clink%2Cname%2Ctype%2Cshares%2Ccreations%2Ccreated_time%2Ccomments.limit%280%29.summary%28true%29.limit%280%29.summary%28true%29&since=2017-01-01&until=2017-10-06&limit=20&after=Q2c4UlpXNTBYM0YxWlhKNVgzTjBiM0o1WDJsaOR5UXLOREF5TmpNME1ESTJPVGs1TVRnNkxUMTRNVEEyTmpZAek5qQXdNREV6TURVeU5qSVBER0ZA3YVY5emRH0XllVjlwMKE4ZA0lqUXdNa1l6tkRBUE5qazVPVEU0HpfFMUSUTTRNVGMwTmprRMK56YzRNeklQQkhScGjXVudxZAFE1bUFFFPQZDZ"
}
}
```

1. 응답 객체는 dict 타입의 객체이다
2. 'data' 키와 매핑된 list 객체가 있다.
3. 'paging' 키와 매핑된 dict 객체가 있다.

# 1. 페이스북 API를 활용한 빈도분석

1. fb\_fetch\_post(...) 함수는 결과의 'data'키의 list를 반환 하는 함수로 수정한다.
2. paging 키 이름의 사전 객체는 페이지 조회를 위한 용도로 사용된다.
  - cursor 속성의 안의 after, before 는 API url에 파라미터로 추가하여 이전 요청과 이 후 요청을 가능하게 한다.
  - next와 previous 속성은 cursor 파라미터가 포함된 전체 url이다.
  - 기존 url에 cursor 파라미터를 추가 하던지 next, previous 속성의 값, 즉 url를 사용하던지 상관없다.
  - 끝과 처음을 판별하기 위해서는 next 또는 previous의 존재 유무로 판단하면 된다.
3. generator를 사용해서 fb\_fetch\_post(...) 함수를 for 푸르안에서 사용 가능하도록 수정한다.

```
def fb_fetch_posts(pagename, since, until):  
    ...  
    ...  
    while True:  
        json_result = json_request(url=url)  
  
        paging = json_result.get('paging')  
        url = paging.get('next')  
  
        if url is None:  
            break  
  
        yield json_result.get('data')
```



# 1. 페이스북 API를 활용한 빈도분석

4. 테스트 코드는 다음과 같다.

[crawler\_fb.py]

```
from api_fb import *

pagename = 'jtbcnews'
since = '2017-01-01'
until = '2017-10-06'

for posts in fb_fetch_posts(pagename, since, until):
    print(posts)
```

# 1. 페이스북 API를 활용한 빈도분석

## 4) 크롤러 작성

앞에서 작성했던 api\_fb 모듈의 함수들을 사용해서 크롤러를 작성한다.

- 응답 JSON(dict)을 빈도 분석을 위해 단순화 시킨다.
- 파일로 저장한다.

[crawler\_fb.py]

크롤러 기본 코드 이다.

```
from api_fb import *

if __name__ == '__main__':
    pagename = 'jtbcnews'
    since = '2017-10-03'
    until = '2017-10-06'
    result = []

    for posts in fb_fetch_posts(pagename, since, until):
        for post in posts:
            preprocess_post(post)
        result += posts

    print(result)
```

# 1. 페이스북 API를 활용한 빈도분석

## 4) 크롤러 작성

preprocess\_post() 함수에서 각각의 post의 사전 구조를 단순화 시키고 분석을 위한 데이터 처리도 하게 된다.

### # 공유수

```
if 'shares' not in post:
    post['count_shares'] = 0
else:
    post['count_shares'] = post['shares']['count']
    del post['shares']
```

### # 전체 리액션 수

```
if 'reactions' not in post:
    post['count_reactions'] = 0
else:
    post['count_reactions'] = post['reactions']['summary']['total_count']
    del post['reactions']
```

### # 전체 리액션 수

```
if 'reactions' not in post:
    post['count_reactions'] = 0
else:
    post['count_reactions'] = post['reactions']['summary']['total_count']
    del post['reactions']
```

# 1. 페이스북 API를 활용한 빈도분석

시간 변경하기

- 페이스북은 UTC(협정 세계시, 또는 그리니치 평균시 GMT)를 사용한다.
- KST는 UTC 보다 +9 이다. (국제시 -> 국내시 변환)

```
# KST = UTC+9
```

```
kst = datetime.datetime.strptime(post['created_time'], '%Y-%m-%dT%H:%M:%S+0000')
```

```
kst = kst + datetime.timedelta(hours=+9)
```

```
post['created_time'] = kst.strftime('%Y-%m-%d %H:%M:%S')
```

# 1. 페이스북 API를 활용한 빈도분석

파일로 저장하기

```
def restore_result(page_name, from_date, to_date, obj_data):  
  
    filename = 'fb_%s_%s_%s.json' % (page_name, from_date, to_date)  
  
    with open(filename, 'w', encoding='utf8') as outfile:  
        json_string = json.dumps(obj_data, indent=4, sort_keys=True, ensure_ascii=False)  
        outfile.write(json_string)
```

파이썬의 list, dict 등의 객체를 JSON 문자열로 바꾸는데, json.dumps() 함수를 사용한다.

- indent는 여러행으로 보기좋게 JSON 문자열을 만든다.
- sort\_key 는 JSON안의 속성(key)를 정렬한다.
- ensure\_ascii 는 false로 하여 ascii가 아닌 문자가 escape 되는 것을 막는다.

# 1. 페이스북 API를 활용한 빈도분석

## 1.2.4 형태소 분석 기반의 빈도분석

### 1) 형태소 분석

- 어떤 대상 어절의 모든 가능한 분석 결과를 추출하는 것을 말한다.
- 문장의 주요 단어(색인어)를 추출하는 것
- 즉, 문장 중에 명사에 해당하는 부분만 추출해야 하는데 한글은 영어권 문장보다 상대적으로 어렵다.

예)

복합명사 : 눈발 -> 눈 + 발, 눈발

눈물 -> 눈 + 물, 눈물

조사 : 나는 -> 나 + 는

어미 : 나는 -> 날다 + 는

- 다양한 형태의 활용과 의미 분석이 필요하기 때문에 자연어 처리 및 형태소 분석은 단순한 문제가 아니다.
- 국내 한글 형태소 분석 오픈소스 : KoNLPy, 은전한닢

# 1. 페이스북 API를 활용한 빈도분석

## 2) KoNLPy 설치 및 확인

1. java 1.7 이상 설치 ( 반드시 JAVA\_HOME 환경 변수 설정 할 것)

2. jpye1 설치

java로 작성된 모듈을 사용해야 하기 때문에 JPyep(0.5.7)이상 설치 해야 한다.

pip(파이썬 패키지 관리자) 프로그램을 통해서 설치한다.

```
pip install jpye1
```

만약, Microsoft Visual C++ 14.0 required 오류가 발생하면

- Visual Studio 를 설치 한다.

<http://landinghub.visualstudio.com/visual-cpp-build-tools> 에서 다운로드 받아서 설치하자.

- wheel 파일을 설치한다.

<http://www.lfd.uci.edu/~gohlke/pythonlibs/> 에서 파이썬 버전에 맞는 jpye1 설치 whl 파일을  
다운받아 설치한다.

```
C:\Users\Wkicks\Downloads>python -m pip install JPyep1-0.6.2-cp36-cp36m-win_amd64.whl
```

```
Processing c:\Users\Wkicks\Downloads\jpye1-0.6.2-cp36-cp36m-win32.whl
```

```
Installing collected packages: JPyep1
```

```
Successfully installed JPyep1-0.6.2
```

# 1. 페이스북 API를 활용한 빈도분석

## 2. jpye1 설치

설치 확인

```
C:\Users\Wkicks\Downloads>pip list  
astroid (1.5.3)  
colorama (0.3.9)  
isort (4.2.15)  
JPype1 (0.6.2)  
lazy-object-proxy (1.3.1)  
mccabe (0.6.1)  
pip (9.0.1)  
pylint (1.7.2)  
setuptools (28.8.0)  
six (1.10.0)  
wrapt (1.10.11)
```

## 3. KoNLPy 설치

```
C:\Users\Wkicks\Downloads>pip install KoNLPy  
Collecting KoNLPy  
  Downloading konlpy-0.4.4-py2.py3-none-any.whl (22.5MB)  
    100% |#####| 22.5MB 61kB/s  
Installing collected packages: KoNLPy  
Successfully installed KoNLPy-0.4.4
```



# 1. 페이스북 API를 활용한 빈도분석

## 4. KoNLPy 설치 확인

[test\_konlpy.py]

```
from konlpy.tag import Kkma

kkma = Kkma()

sentences = kkma.sentences(u'네, 안녕하세요. 반갑습니다.')
print(sentences)

nouns = kkma.nouns(u'명사만을 추출하여 다빈도 분석을 합니다.')
print(nouns)

pos = kkma.pos(u'오류보고는 실행환경, 에러메세지와함께 설명을 최대한상세히!^^')
print(pos)
```

# 1. 페이스북 API를 활용한 빈도분석

## 3) 빈도 분석(Frequency Analysis)

1. 앞에서 페이스북 API를 이용해 특정 페이지의 포스트 데이터들 중 message 내용을 형태소 분석을 통해 단어를 추출하고 단어를 카운트해서 단어들의 빈도를 구한다.
2. 통계 분석에 변수(단어)의 빈도를 도수라 하고 이를 표로 정리한 것을 도수분포표라고 한다.
3. 도수분포표는 통계 분석의 가장 기본이 되는 빈도 분석에서 가장 기초가 되는 데이터라 할 수 있다
4. 통계 분석에서 빈도 분석의 정의
  - 통계 분석 기법 중 가장 기본이 되는 분석방법이다. 즉, 수집한 자료의 특성을 파악하기 위한 첫 번째 단계로 데이터의 분포현황을 파악할 수 있다.
  - 변수들의 빈도, 중심경향치, 분포도 등 변수의 개략적 특성을 살펴보는 분석방법이다.
  - 빈도분석은 기술통계량(평균, 표준편차, 분산, 표준오차, 중앙값 ...)을 제공해 주고 막대그래프 (Bar Chart), 원형그래프(Pie Chart), 히스토그램 (Histogram) 등의 그래프를 제공하여 통계량을 보다 쉽게 이해할 수 있게 한다.

# 1. 페이스북 API를 활용한 빈도분석

수집된 포스트 데이터에서 message 데이터에서 단어 추출 및 빈도 카운팅 하기.

1. analysis 패키지에 analyzer.py 모듈을 추가 한다.

2. json\_to\_str(filename, key)

json포맷의 파일 이름을 입력 받아 각각의 아이템에서 key에 해당하는 value를 공백문자를 제거하고 하나의 문자열로 합치는 함수이다.

```
def json_to_str(filename, key):  
    jsonfile = open(filename, 'r', encoding='utf-8')  
    jsondata = json.loads(jsonfile.read())  
  
    data = ""  
    for items in jsondata:  
        value = items.get(key)  
        if value is None:  
            continue  
  
        data = ' '.join((data, (re.sub(r'^\w', '', value))))  
  
    return data
```

# 1. 페이스북 API를 활용한 빈도분석

## 3. count\_world(data)

입력 받은 문자열에서 단어를 추출하고 빈도수를 세서 단어별 빈도수의 dict 타입으로 리턴하는 함수이다.

```
def count_wordfreq(data):  
    twitter = Twitter()  
    nouns = twitter.nouns(data)  
    count = Counter(nouns)  
  
    return count
```

## 4. 테스트 해보자

\_\_main\_\_.py 에 다음 코드를 추가 하고 테스트 한다.

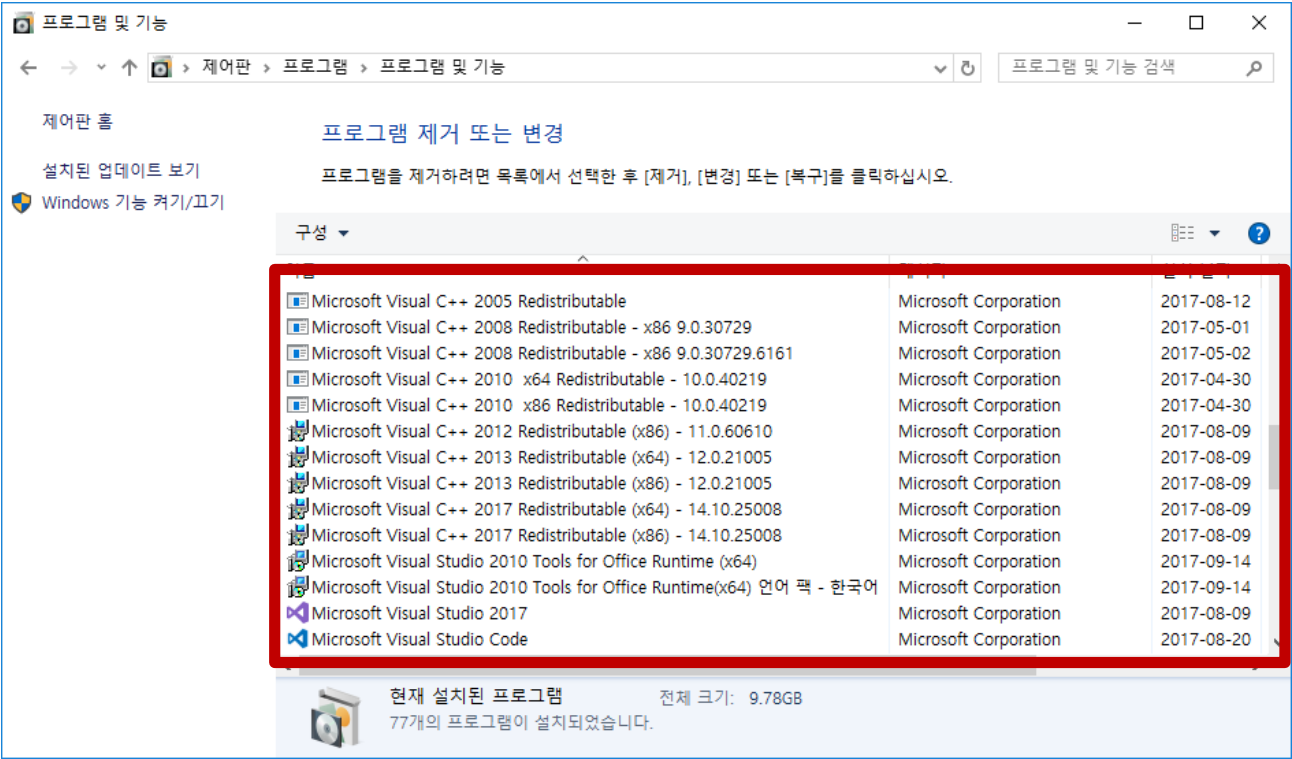
```
# analysis  
for item in items:  
    data = analysis.json_to_str(item.get('resultfile'), 'message')  
    item['count'] = analysis.count_wordfreq(data)  
  
# test  
for item in items:  
    print(item)
```

# 1. 페이스북 API를 활용한 빈도분석

## 1.2.5 matplotlib을 이용한 그래프 그리기

### 1) 파이썬 그래프 모듈: matplotlib

- 파이썬에서는 그래프 기능 지원모듈 중 가장 일반적인 모듈
- 설치
  1. 파이썬 3.4(windows) 이상인 경우 Microsoft Visual C++ (2008,2010) 배포 패키지(Redistributable)가 필요하다.
  2. 확인



# 1. 페이스북 API를 활용한 빈도분석

– 설치

3. Microsoft Visual C++ (2008,2010) 배포 패키지(Redistributable)가 설치가 되어 있지 않으면 다운로드해서 설치한다.

Visual Studio 2017용 Visual C++ 재배포 가능 패키지(x64)

<https://go.microsoft.com/fwlink/?LinkId=746572>

Visual Studio 2017용 Visual C++ 재배포 가능 패키지(x86)

<https://go.microsoft.com/fwlink/?LinkId=746571>

Microsoft Visual C++ 2015 재배포 가능 패키지 Update 3

<https://www.microsoft.com/en-us/download/details.aspx?id=53840>

Visual Studio 2013용 Visual C++ 재배포 가능 패키지

<https://www.microsoft.com/ko-kr/download/details.aspx?id=40784>

[http://download.microsoft.com/download/f/8/d/f8d970bd-4218-49b9-b515-e6f1669d228b/vcredist\\_x64.exe](http://download.microsoft.com/download/f/8/d/f8d970bd-4218-49b9-b515-e6f1669d228b/vcredist_x64.exe)

[http://download.microsoft.com/download/f/8/d/f8d970bd-4218-49b9-b515-e6f1669d228b/vcredist\\_x86.exe](http://download.microsoft.com/download/f/8/d/f8d970bd-4218-49b9-b515-e6f1669d228b/vcredist_x86.exe)

## 1. 페이스북 API를 활용한 빈도분석

## - 설치

#### 4) matplotlib 설치

```
C:\Users\Wkicks\Downloads>pip install matplotlib
```

## Collecting matplotlib

Downloading matplotlib-2.0.2-cp36-cp36m-win\_amd64.whl (8.9MB)

100% | 8.9MB 159kB/s

## Collecting pytz (from matplotlib)

Downloading pytz-2017.2-py2.py3-none-any.whl (484kB)

100% | 491kB 1.1MB/s

Collecting pyparsing!=2.0.4,!2.1.2,!2.1.6,>=1.5.6 (from matplotlib)

Downloading pyparsing-2.2.0-py2.py3-none-any.whl (56kB)

[illegible]

Requirement already satisfied: numpy>=1.7.1 in c:\wpython\wpython36\lib\site-packages (from matplotlib)

### Collecting `cycler>=0.10` (from matplotlib)

Downloading `cycler-0.10.0-py2.py3-none-any.whl`

## Collecting python-dateutil (from matplotlib)

Downloading python\_dateutil-2.6.1-py2.py3-none-any.whl (194kB)

100% | 194kB 1.1MB/s

## Collecting six $\geq$ 1.10 (from matplotlib)

Downloading six-1.11.0-py2.py3-none-any.whl

Installing collected packages: pytz, pyparsing, six, cyclcr, python-dateutil, matplotlib

Successfully installed cyclcr-0.10.0 matplotlib-2.0.2 pyparsing-2.2.0 python-dateutil-2.6.1 pytz-2017.2 six-1.11.0

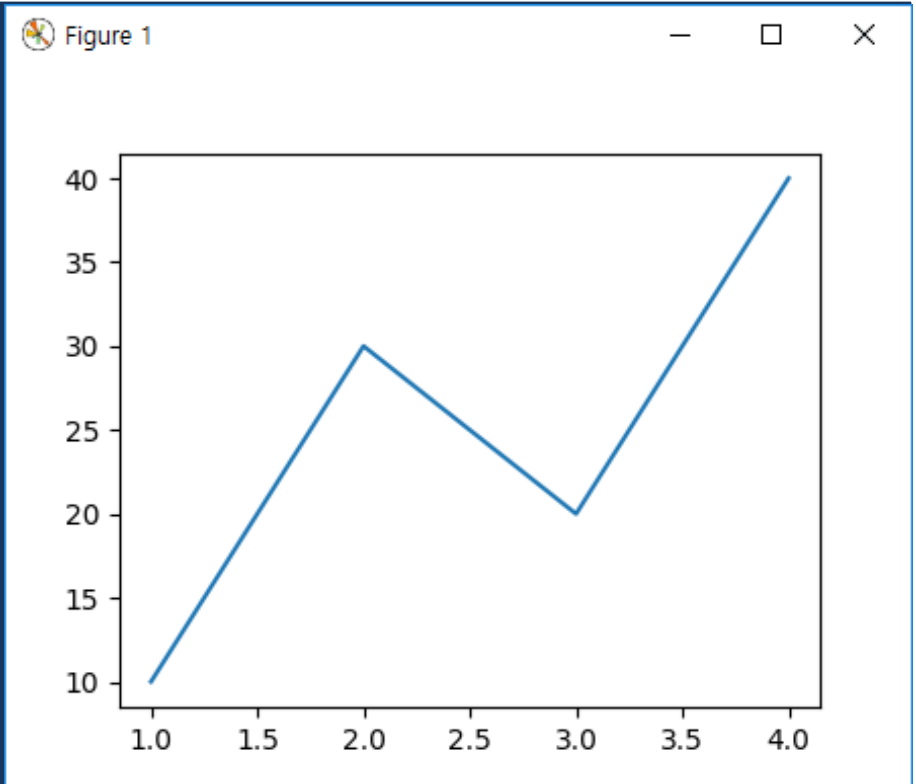
# 1. 페이스북 API를 활용한 빈도분석

- 설치

5. numpy, pyparsing, cyclr 패키지가 함께 설치된다. 테스트 해보자

[test\_matplotlib.py] - ex1

```
def ex1():  
    plt.plot([1, 2, 3, 4], [10, 30, 20, 40])  
    plt.show()  
  
if __name__ == '__main__':  
    ex2()
```





# 1. 페이스북 API를 활용한 빈도분석

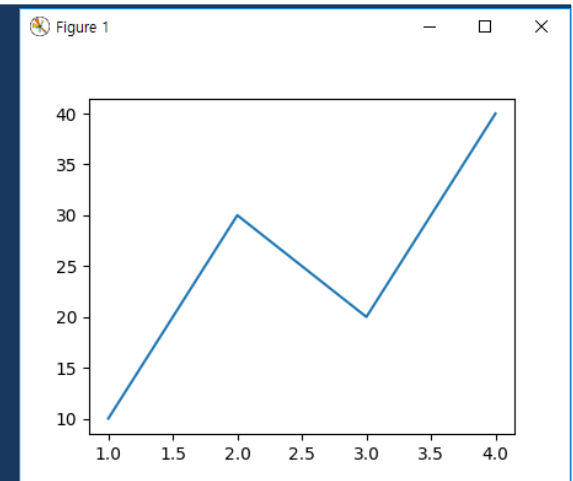
## 2) matplotlib API 살펴보기

### 1. Figure 와 Subplot(서브플롯)

- matplotlib에서 그래프는 Figure 객체내에 존재한다.
- Figure 객체를 생성하고 add\_subplot을 통해 서브플롯을 추가 하게 된다.

[test\_matplotlib.py] - ex2

```
def ex2():  
    fig = plt.figure()  
  
    sp1 = fig.add_subplot(1, 1, 1)  
    sp1.plot([1, 2, 3, 4], [10, 30, 20, 40])  
  
    plt.show()
```



이 예제는 ex1과 결과가 같다.

add\_subplot의 파라미터의 의미는 크기가 1X1 이고 첫 번째 플롯을 선택하겠다는 의미이다.

# 1. 페이스북 API를 활용한 빈도분석

[test\_matplotlib.py] - ex2

크기가 1 X 2 인 Figure 2개의 플롯을 추가한 예이다. 직접 확인해보자.

```
fig = plt.figure()
```

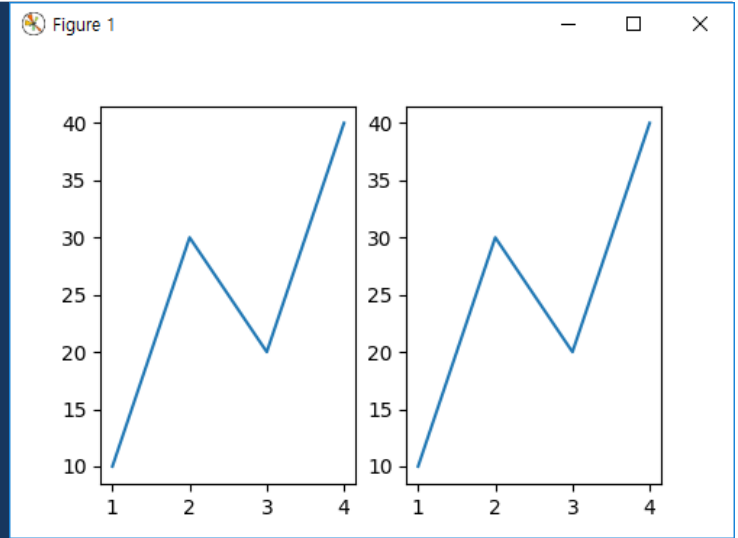
```
sp1 = fig.add_subplot(1, 2, 1)
```

```
sp1.plot([1, 2, 3, 4], [10, 30, 20, 40])
```

```
sp2 = fig.add_subplot(1, 2, 2)
```

```
sp2.plot([1, 2, 3, 4], [10, 30, 20, 40])
```

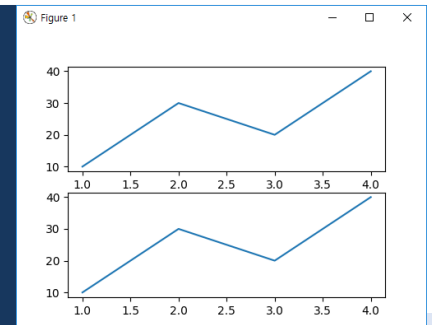
```
plt.show()
```



ex1 에서 plt.plot()은 서브플롯이 없기 때문에 서브플롯을 자동으로 생성해서 Figure 와 서브플롯 생성되는 과정을 숨긴다.

Figure 와 서브플롯을 직접 생성하게 되면, plt.plot()은 가장 최근 Figure와 서브플롯에 그래프를 추가하게 된다.

다음 그림과 같이 플롯이 2개가 추가된 그래프를 그려보자.



# 1. 페이스북 API를 활용한 빈도분석

[test\_matplotlib.py] - ex3

numpy 이용해서 다양한 그래프를 그려보자.

```
fig = plt.figure()
```

```
sp1 = fig.add_subplot(2, 2, 1)
```

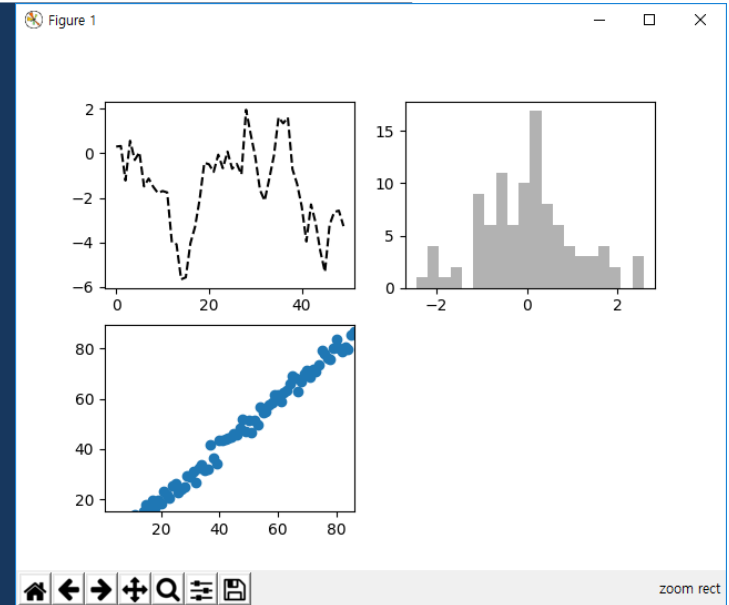
```
sp1.plot(randn(50).cumsum(), 'k--')
```

```
sp2 = fig.add_subplot(2, 2, 2)
```

```
sp2.hist(randn(100), bins=20, color='k', alpha=0.3)
```

```
sp3 = fig.add_subplot(2, 2, 3)
```

```
sp3.scatter(np.arange(100), np.arange(100) + 3 * randn(100))
```



서브플롯이 하나만 있더라도 Figure를 생성하는 일은 보통 하게된다.

보통, plt.subplots 이라는 메서드를 사용하게 된다.

plt.subplots Numpy 배열에 서브플롯 객체를 추가하여 반환한다.

# 1. 페이스북 API를 활용한 빈도분석

[test\_matplotlib.py] - ex4

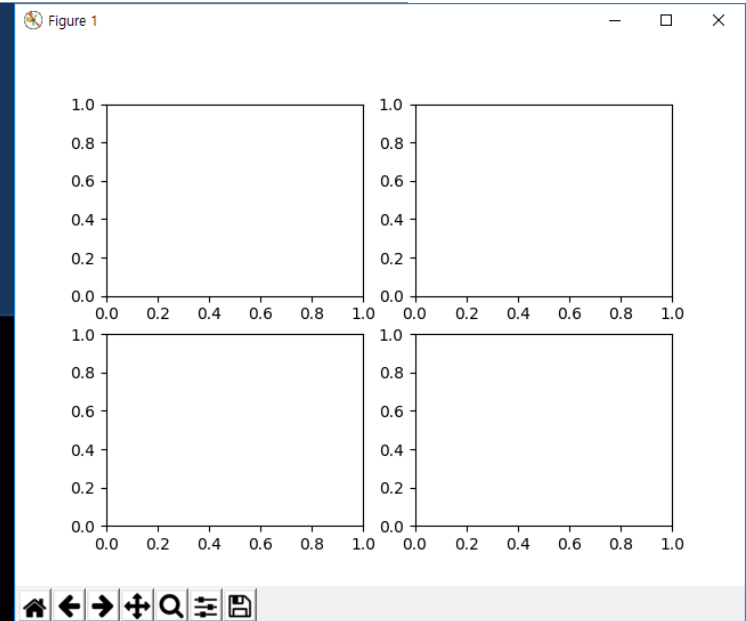
plt.subplots를 사용해 보자.

```
fig, subplots = plt.subplots(2, 2)
```

```
print(subplots)
```

```
plt.show()
```

```
[[<matplotlib.axes._subplots.AxesSubplot object at 0x000001C3E6005DD8>  
 <matplotlib.axes._subplots.AxesSubplot object at 0x000001C3E9D49320>]  
 [<matplotlib.axes._subplots.AxesSubplot object at 0x000001C3E9DE42B0>  
  <matplotlib.axes._subplots.AxesSubplot object at 0x000001C3E9DEEB38>]]
```



AxesSubplot 타입의 2차원 배열이 반환되는 것을 알 수 가 있다. subplots 메서드의 인자는 다음과 같다.

nrows : 서브플롯의 로우 수

ncols : 서브플롯의 칼럼 수

sharex : 서브플롯이 x축 눈금을 함께 쓴다.

sharey : 서브플롯이 y축 눈금을 함께 쓴다.

앞의 ex3과 같은 그래플이 그려지도록 수정 해보자.

# 1. 페이스북 API를 활용한 빈도분석

## 2. 서브플롯 간격 조절하기

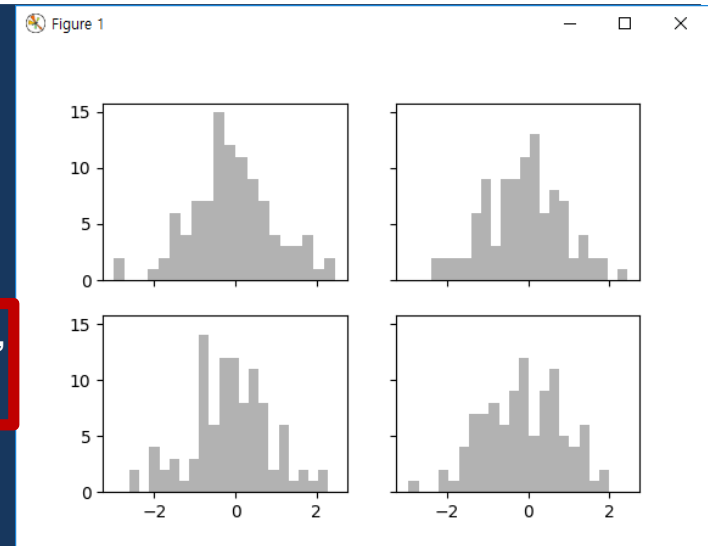
- matplotlib은 서브플롯 간에 적당한 간격과 여백을 추가해준다.
- plt.subplots\_adjust 메서드를 사용해 쉽게 바꿀 수 있다.

[test\_matplotlib.py] - ex5

```
fig, subplots = plt.subplots(2, 2, sharex=True, sharey=True)
for i in range(2):
    for j in range(2):
        subplots[i, j].hist(randn(100), bins=20, color='k', alpha=0.3)
```

```
plt.subplots_adjust(left=None, bottom=None, right=None, top=None,
                    wspace=None, hspace=None)
```

```
plt.show()
```



wspace, hspace 값을 조정하면 서브플롯간의 간격 조절이 가능하다. wspace=0 , hspace=0 하고 그래프를 확인해보자.

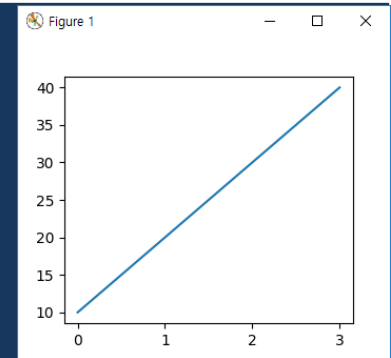
# 1. 페이스북 API를 활용한 빈도분석

## 3. 색상, 마커, 선 스타일

- plot() 메서드는 다양한 형태의 인자값을 가진다.

[test\_matplotlib.py] - ex6

```
fig, subplots = plt.subplots(1, 1)
subplots.plot([10, 20, 30, 40])
plt.show()
```

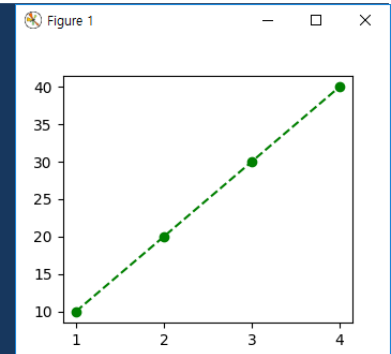


y축 값만 할당한 경우이다. x값은 내부에서 자동으로 0부터 매칭되는 것을 알 수가 있다.

- plot 메서드는 그래프의 색상과 스타일, 마커등을 문자열로 표현하여 인자로 넘겨줄 수 있다.

[test\_matplotlib.py] - ex7

```
fig, subplots = plt.subplots(1, 1)
subplots.plot([1, 2, 3, 4], [10, 20, 30, 40], 'go--')
plt.show()
```

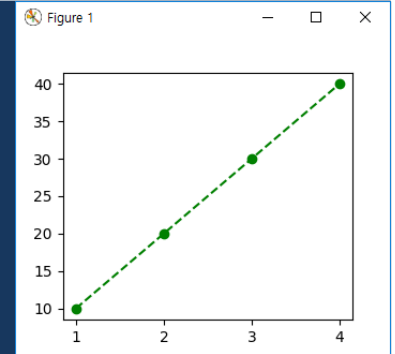


# 1. 페이스북 API를 활용한 빈도분석

- 문자열 인자 전달은 의미전달과 복잡해 보이기 때문에 잘 사용하지 않고 다음과 같이 명시적 방법을 선호한다.

[test\_matplotlib.py] - ex8

```
fig, subplots = plt.subplots(1, 1)
subplots.plot([1, 2, 3, 4], [10, 20, 30, 40], color="g", linestyle="--", marker="o")
plt.show()
```



color : 복합 문자열 전달에서는 k, r, b, g, y, .....

명시적 표현에서는 black, red, blue, green, yellow . . . . 와 #rrggbb도 가능하다

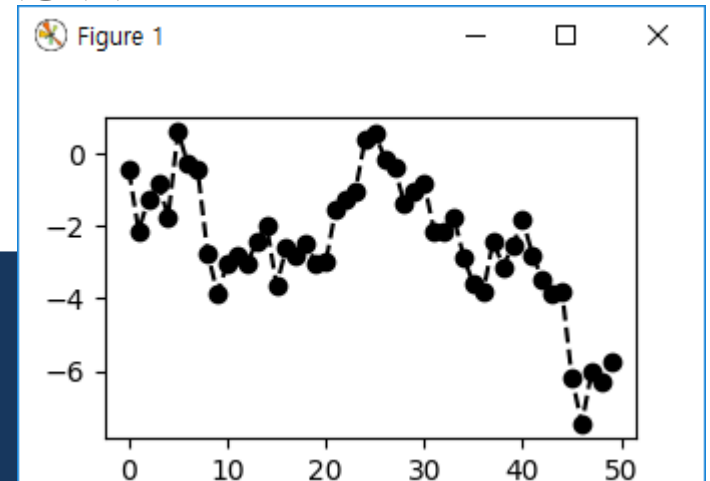
linestyle : - (solid), -- (dashed), -. (dashdot), dotted, ' '(None) 등이 가능하다.

marker : . (dot) v (화살표), o (big dot)

[test\_matplotlib.py] - ex9

명시적 방법으로 표현하여 보자.

```
fig, subplots = plt.subplots(1, 1)
subplots.plot(randn(50).cumsum(), 'ko--')
plt.show()
```



# 1. 페이스북 API를 활용한 빈도분석

- drawstyle 파라미터를 이용하면 점의 연결을 계단모양으로 바꿀수 있다.

[test\_matplotlib.py] - ex10

```
data = randn(50).cumsum()
```

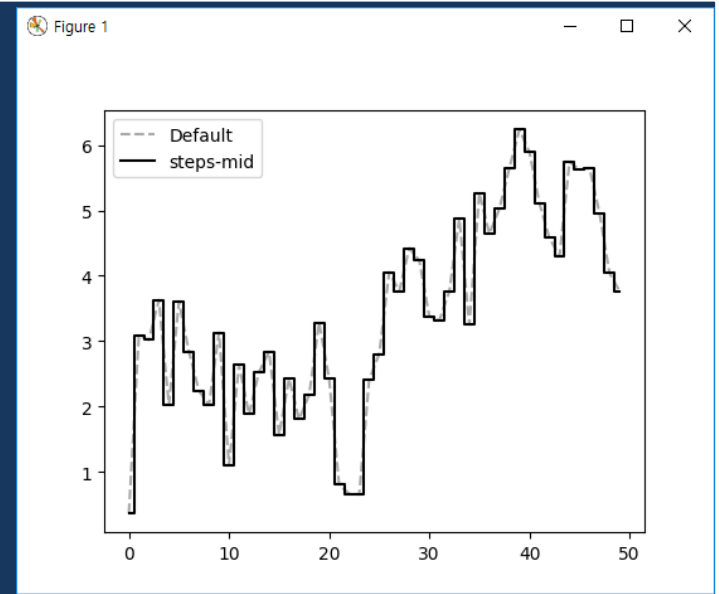
```
fig, subplots = plt.subplots(1, 1)
```

```
subplots.plot(data, color='#aaaaaa', linestyle='--', label='Default')
```

```
subplots.plot(data, 'k-', drawstyle='steps-mid', label='steps-mid')
```

```
plt.legend(loc='best')
```

```
plt.show()
```



이 예제는 하나의 서브플롯에 2개의 그래프를 그린다.

drawstyle은 다음 값들을 사용할 수 있다.

‘default’, ‘steps’, ‘steps-pre’, ‘steps-mid’, ‘steps-post’

다른 스타일을 적용해 보자.



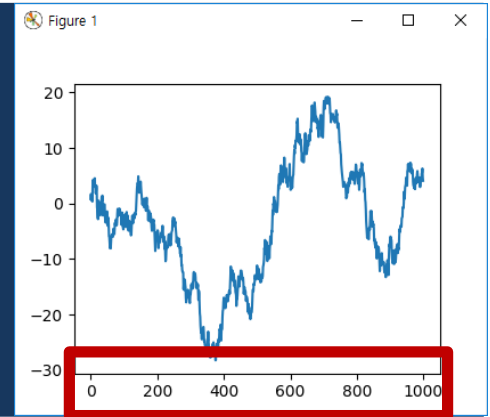
# 1. 페이스북 API를 활용한 빈도분석

## 4. 제목, 축 이름, 눈금, 눈금 이름

[test\_matplotlib.py] - ex11

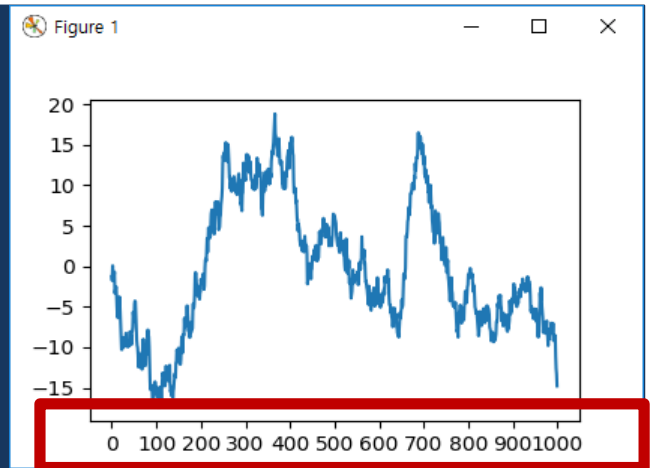
기본 예제가 되는 그래프이다.

```
fig, subplots = plt.subplots(1, 1)
subplots.plot(randn(1000).cumsum())
plt.show()
```



set\_xticks 메서드를 사용하여 x축의 눈금을 변경하여 보자.

```
fig, subplots = plt.subplots(1, 1)
subplots.plot(randn(1000).cumsum())
subplots.set_xticks([0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000])
plt.show()
```



# 1. 페이스북 API를 활용한 빈도분석

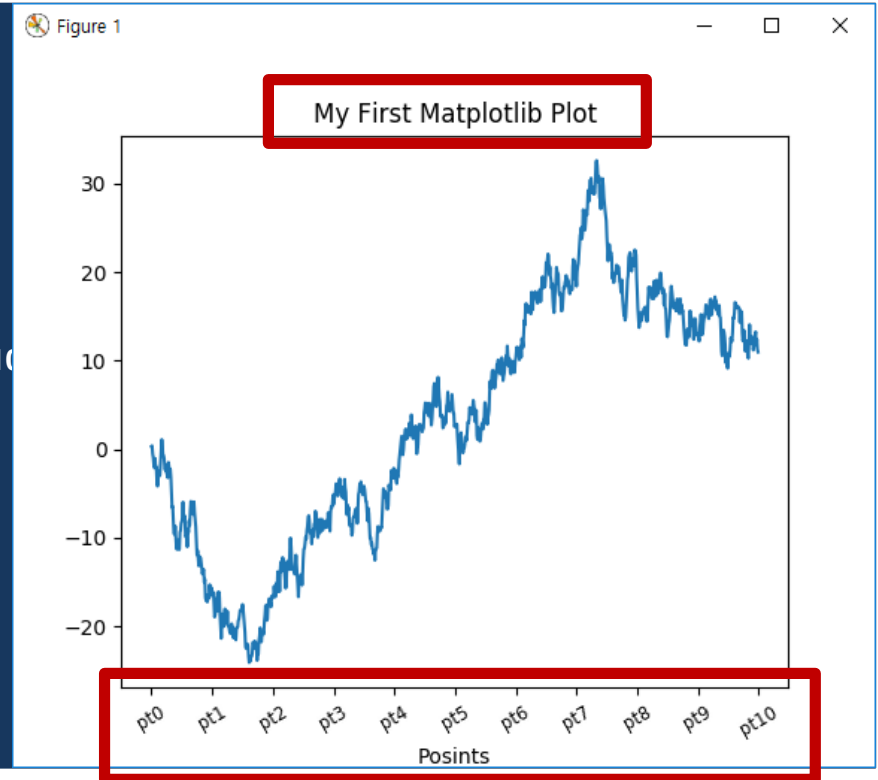
[test\_matplotlib.py] - ex11

set\_xticklabels 메서드를 사용하면 눈금에 다른 이름을 사용할 수 있다.

```
fig, subplots = plt.subplots(1, 1)
subplots.plot(randn(1000).cumsum())

subplots.set_xticks(
    [0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000])
subplots.set_xticklabels(
    ['pt0', 'pt1', 'pt2', 'pt3', 'pt4', 'pt5', 'pt6', 'pt7', 'pt8', 'pt9', 'pt10'],
    rotation=30,
    fontsize='small')
subplots.set_xlabel('Posints')
subplots.set_title('My First Matplotlib Plot')

plt.show()
```



기본 또는 set\_xticks 를 통해서 설정된 눈금수에 맞게 눈금 라벨을 지정할 수 있다.

set\_xlabel 메서드를 통해 x축 이름을, set\_title 메서드를 통해 서브플롯 이름을 설정할 수 있다.

Y축도 같은 메서드(x를 y로 바꾼)를 통해 적용할 수 있다.

# 1. 페이스북 API를 활용한 빈도분석

## 5. 범례(legend) 추가하기

그래프의 요소를 확인하기 위한 중요한 요소

가장 쉬운방법은 각 그래프에 label 파라미터를 넣기는 방법이다

[test\_matplotlib.py] - ex12

```
fig, subplots = plt.subplots(1, 1)
```

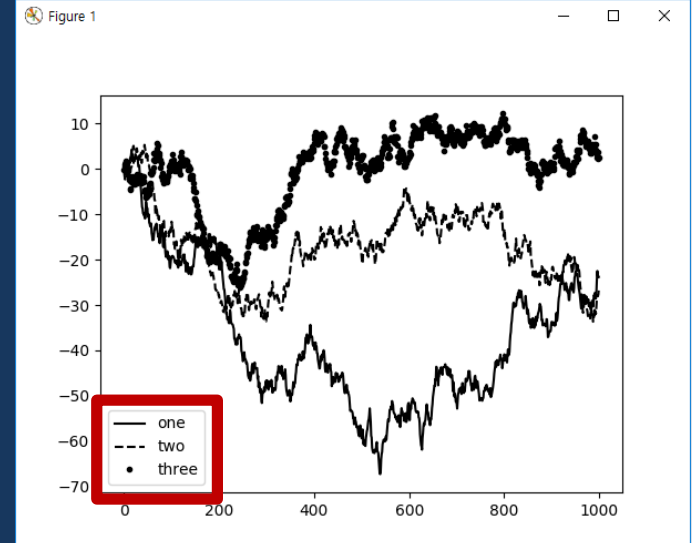
```
subplots.plot(randn(1000).cumsum(), 'k', label='one')
```

```
subplots.plot(randn(1000).cumsum(), 'k-.', label='two')
```

```
subplots.plot(randn(1000).cumsum(), 'k.', label='three')
```

```
plt.legend(loc='best')
```

```
plt.show()
```



# 1. 페이스북 API를 활용한 빈도분석

## 6. 한글처리

다음 예제를 실행 시켜보자. 한글이 깨져 나오는 것을 확인할 수 있다.

[test\_matplotlib.py] - ex12

```
fig, subplots = plt.subplots(1, 1)
```

```
subplots.plot(randn(1000).cumsum(), 'k', label='기본')
```

```
subplots.plot(randn(1000).cumsum(), 'k--', label='대시')
```

```
subplots.plot(randn(1000).cumsum(), 'k.', label='점')
```

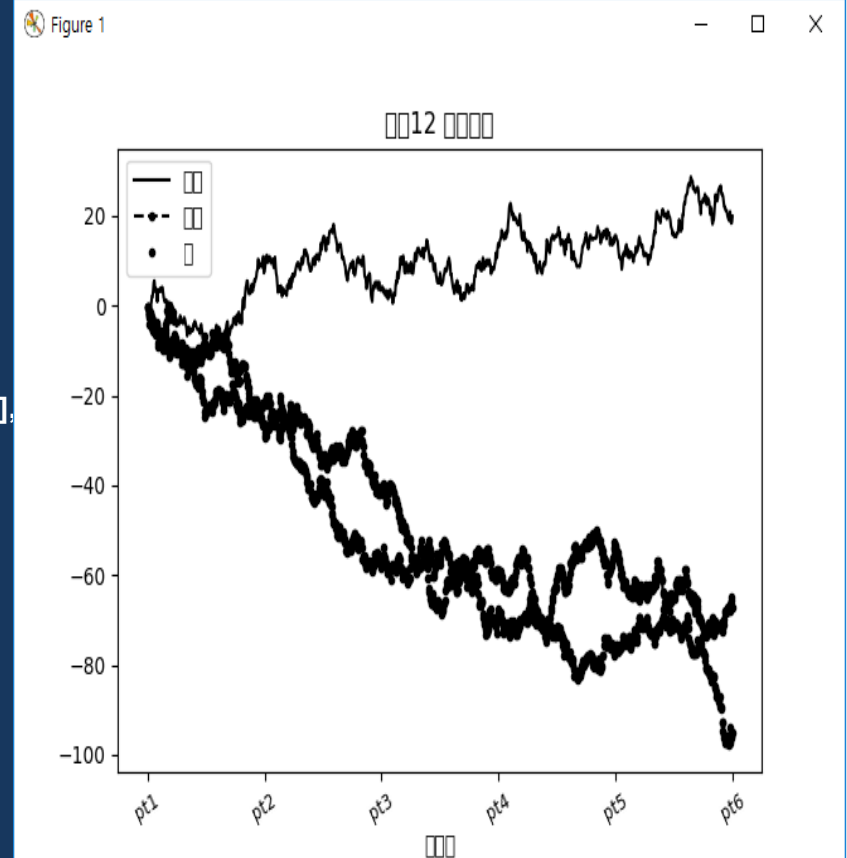
```
subplots.set_xticklabels(  
    ['pt0', 'pt1', 'pt2', 'pt3', 'pt4', 'pt5', 'pt6', 'pt7', 'pt8', 'pt9', 'pt10'],  
    rotation=30,  
    fontsize='small')
```

```
subplots.set_xlabel('포인트')
```

```
subplots.set_title('예제12 한글처리')
```

```
plt.legend(loc='best')
```

```
plt.show()
```



# 1. 페이스북 API를 활용한 빈도분석

한글 처리를 위해서는 matplotlib의 환경 설정을 해 주어야 한다.

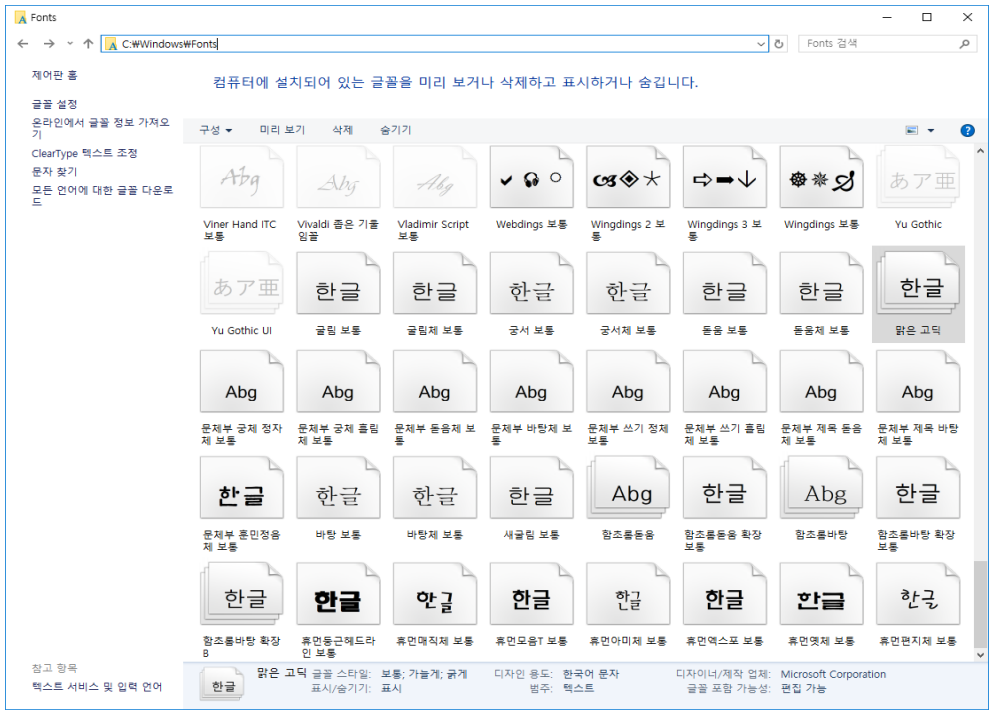
matplotlib의 환경 설정은 rc 메서드를 통해 그래프 크기, 서브플롯 간격, 색상, 글자크기, 격자(grid) 스타일 지정이 가능하다.

폰트의 경우에는 다음과 같이 설정이 가능하다.

```
font_options = { 'family': 'monospace', 'weight':'bold', 'size': 'small' }
plt.rc('font', **font_options)
```

font famly 이름을 알아야 하는데, 폰트 파일만 가지고는 알 수 없다.

C:\Windows\fonts 내용



# 1. 페이스북 API를 활용한 빈도분석

matplotlib의 font\_manager모듈을 사용하면 폰트 파일을 통해 폰트 패밀리 이름을 알 수가 있다.

[test\_matplotlib.py] - ex13

```
font_filename = 'c:/Windows/fonts/malgun.ttf'
font_name = font_manager.FontProperties(fname=font_filename).get_name()
print(font_name)
```

**Malgun Gothic**

Matplotlib에서 한글 처리를 위해서는 다음 코드가 추가되면 된다.

```
font_options = {'family': 'Malgun Gothic'}
plt.rc('font', **font_options)
plt.rc('axes', unicode_minus=False)
```

weight, size 까지 세팅할 필요가 없으면 dict 타입 파라미터로 전달하지 않아도 된다.

```
plt.rc('font', family='Malgun Gothic')
plt.rc('axes', unicode_minus=False)
```

매번 코드에서 추가하기 번거로우면 matplotlib 설정 변경 두번째 방법으로 matplotlib/mpl-data 디렉토리에 matplotlibrc파일의 FONT설정 부분을 변경해 주면 된다.

# 1. 페이스북 API를 활용한 빈도분석

## 7. 파일로 저장하기

plt.savefig 메서드를 이용해서 파일로 저장할 수 있다.

파일의 종류는 확장자를 통해 결정 된다.

[test\_matplotlib.py] - ex14

```
plt.savefig('ex14-plot.png', dpi=400, bbox_inches='tight')
```

dpi는 해상도를 조정하며, bbox\_inches는 Figure 둘레의 공백을 잘라낸다.

활성화된 Figure에 대해서는 파일로 저장이 가능하기 때문에 plt.show() 를 하지 않아도 저장이 가능하다.

BytesIO 같은 객체로 저장할 수도 있다. (출력이 파일이 아니고 메모리이다)

```
from io import BytesIO

buffer = BytesIO()
plt.savefig(buffer)
plot_data = buffer.getvalue()
```

이는 웹을 통해 동적인 그래프를 그릴 때 아주 유용하고 사용될 수 있다.

# 1. 페이스북 API를 활용한 빈도분석

## 3) 빈도분석 결과 그래프 그리기

1. visualization 패키지에 visualize.py 모듈을 추가 한다.

2. graph\_bar 함수 작성

```
def graph_bar(title=None, xlabel=None, ylabel=None, showgrid=False, values=None, ticks=None, filename=None, showgraph=True):  
    fig, subplots = plt.subplots(1, 1)  
    if values is not None and isinstance(values, collections.Sequence):  
        subplots.bar(range(len(values)), values, align='center')  
    if ticks is not None and isinstance(values, collections.Sequence):  
        subplots.set_xticks(range(len(values)))  
        subplots.set_xticklabels(ticks, rotation=70, fontsize='small')  
    if title is not None and isinstance(title, str):  
        pass  
    if xlabel is not None and isinstance(title, str):  
        pass  
    if ylabel is not None and isinstance(title, str):  
        pass  
    subplots.grid(showgrid)  
    if filename is not None and isinstance(filename, str):  
        save_filename = '%s/bar_%s' % (RESULT_DIRCOTRY, filename)  
        plt.savefig(save_filename, dpi=400, bbox_inches='tight')  
    if showgraph:  
        plt.show()
```



## 1. 페이스북 API를 활용한 빈도분석

## 2. graph\_bar 함수 작성

title, xlabel, ylabel, showgrid, values, ticks, filename, showgraph 등 다양한 파라미터를 받는다.

### 3. 테스트 하기

`__main__.py` 에 다음 코드를 추가하고 그래프를 확인해 보자

## # visualize

```
for item in items:
```

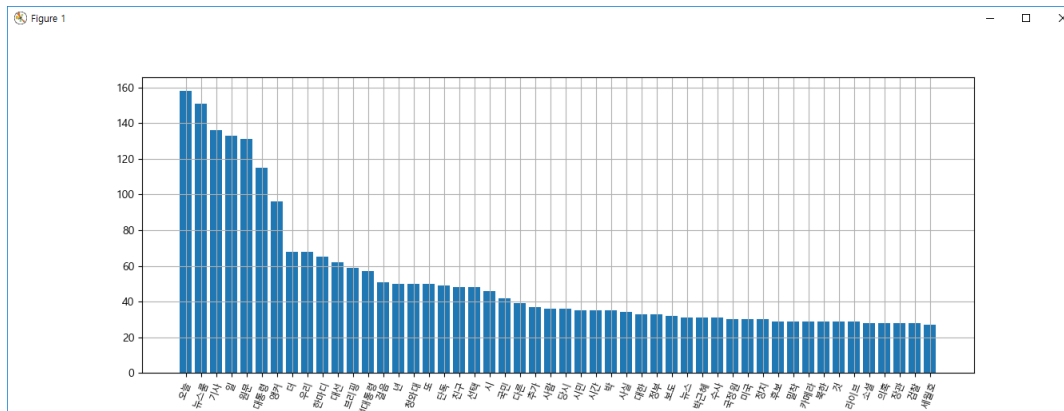
## #전처리

```
count = item['count']
```

```
count_t50 = dict(count.most_common(50))
```

```
filename = '%s_%s_%s.png' % (item['pagename'], item['since'], item['until'])
```

```
visualize.graph_bar(values=list(count_t50.values()), ticks=list(count_t50.keys()), showgrid=True, filename=filename)
```



# 1. 페이스북 API를 활용한 빈도분석

## 1.2.6 matplotlib을 이용한 그래프 그리기

### 1) 파이썬 그래프 모듈: matplotlib

- 파이썬에서는 그래프 기능 지원모듈 중 가장 일반적인 모듈
- 설치
  1. 파이썬 3.4(windows) 이상인 경우 Microsoft Visual C++ (2008,2010) 배포 패키지(Redistributable)가 필요하다.
  2. 확인

