03.페이스북_데이터 수집, 분석, 시각화

01. 데이터 수집

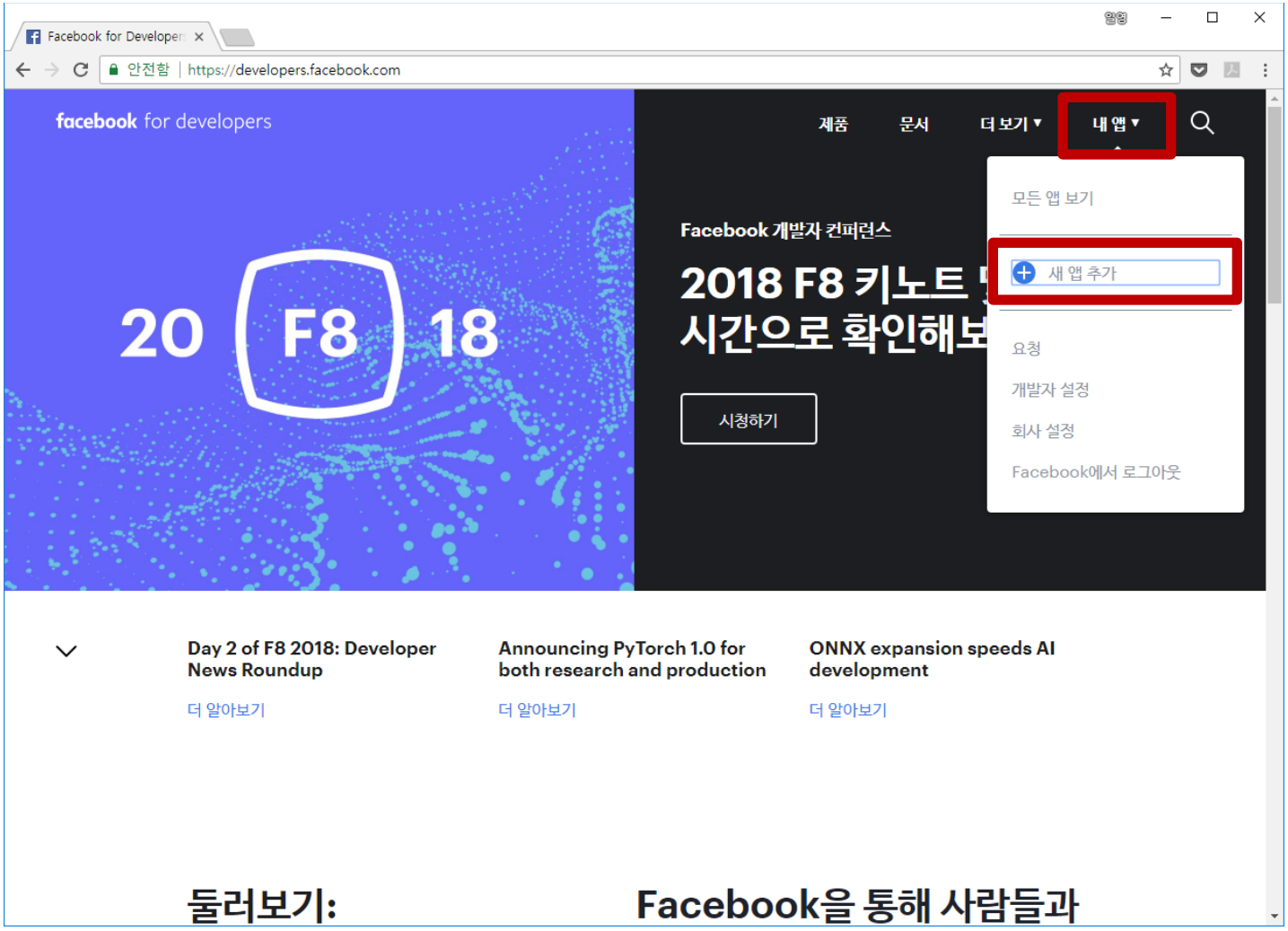
02. 데이터 분석

03. 시각화

1 데이터 수집

1.1 페이스북 앱ID등록

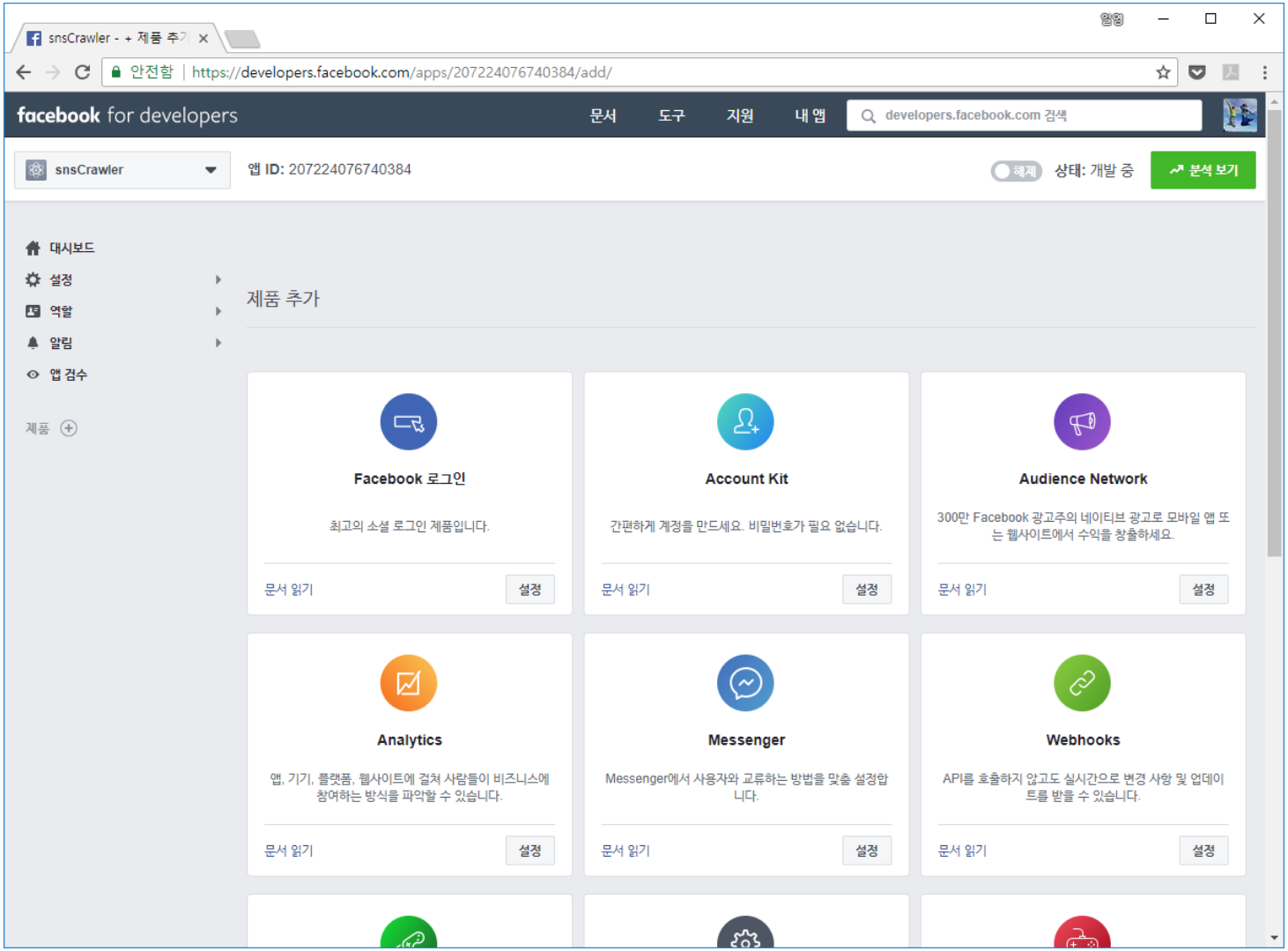
1) 개발자 페이지 [https://developers.facebook.com]



2) 새로운 앱 만들기


새 앱 ID 만들기
Facebook을 앱이나 웹사이트로 통합합니다
표시 이름
snsCrawler
연락처 이메일
자신의 이메일 주소
계속하면 Facebook 플랫폼 정책에 동의하는 것입니다
취소
앱 ID 만들기

3) 앱 관리 페이지 – 대시보드



1.2 JSON 통신 모듈 만들기

1) request 모듈을 이용한 HTTP통신

- url을 통해 요청을 하고 json형태의 정보를 받아온다

get_json_result(url) 함수

```
import requests
from datetime import datetime

def get_json_result(url):
    try:
        response = requests.get(url)

        if response.status_code == 200:
            json_result = response.json()

        return json_result

    except Exception as e:
        return '%s : Error for request [%s]' % (datetime.now(), url)
```

get_json_result(url) 사용

```
url = "http://localhost:8088/mysite4/api/gb/list"
result = get_json_result()
print(result)
```

1.3 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 댓글등 항목 간 연결 링크

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

3) 소셜 그래프에서 데이터를 가져오거나 게시하는 HTTP 기반의 API

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

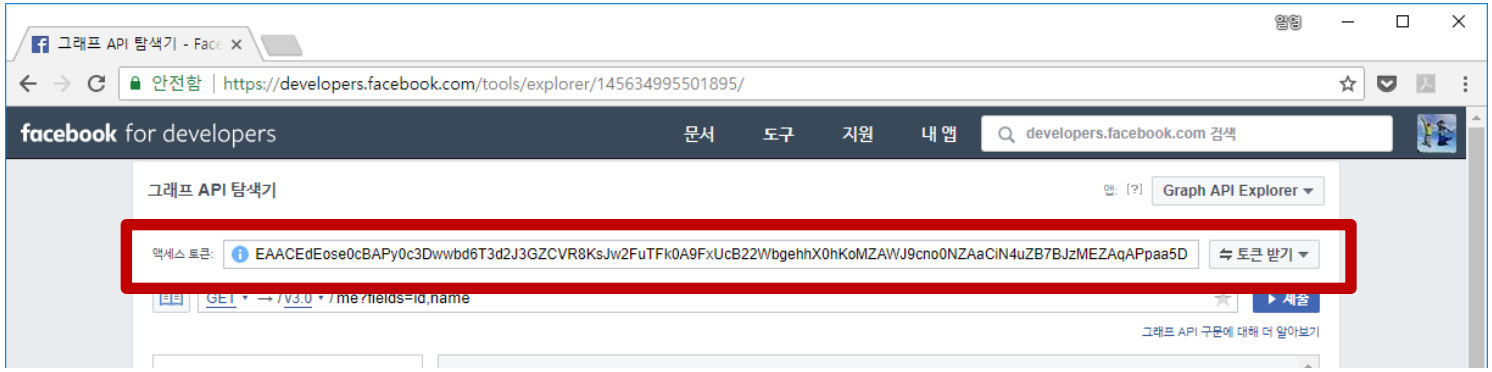
5) 대부분의 그래프 요청은 액세스 토큰을 사용해야 한다.

1.4 페이스북 포스트(jtbcnews) 크롤링하기

1) 액세스 토큰

- 개발자 페이지 [https://developers.facebook.com] 접속
- 오른쪽 상단 " 내앱 > snsCrawler " 클릭 하여 앱관리 페이지로 이동
- 상단 메뉴중 "도구 > 그래프 API 탐색기 " 선택

- 그래프 API 탐색기 페이지 상단의 엑세스 토큰값 사용



2) 그래프 API URL

페이스북 API 요청 url 은 다음과 같은 형식을 가진다.

https://graph.facebook.com/v3.0/[Node, Edge]/?parameters

Node

jtbcnews/posts (jtbcnews 페이지의 포스트들을 가져오는 노드이다)

parameters

아래와 같이 세부 파라미터 속성들이 있다.

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=...

parameters 의 세부정보

fileds : 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 : 시작일, 종료일, 'YYYY-MM-DD' 형식의 날짜 포맷

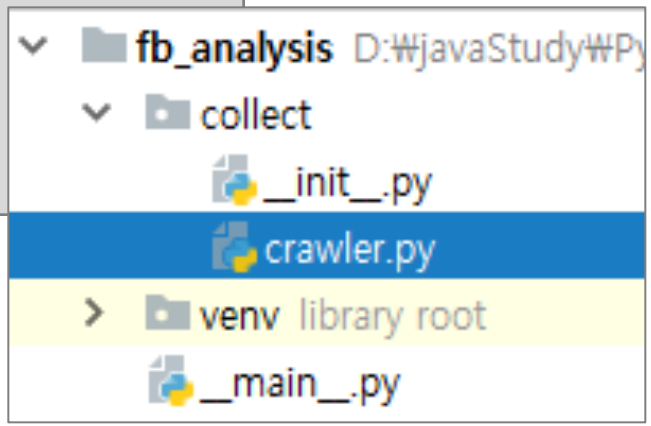
imit : 한 번 요청에 가져올 포스트 수

3) jtbcnews id 값 알아오기

페이스북 API url 에서는 페이지 이름대신 아이디를 사용해야 한다.
API를 이용해서 페이지이름을 페이지 아이디로 바꿔보자

fb_name_to_id(pagename) 함수 : 페이스북 페이지값으로 코드값 구해준다.

```
BASE_URL_FB_API = "https://graph.facebook.com/v3.0"
ACCESS_TOKEN = "자신의 토큰값"

def fb_name_to_id(pagename):

    base = BASE_URL_FB_API
    node = "/"+pagename
    params = "/?access_token=%s" % ACCESS_TOKEN
    url = base + node + params

    json_result = get_json_result(url)  # 딕셔너리로 리턴된다

    return json_result["id"]            # id 값만 리턴한다
```



4) jtbcnews 포스트 data 가져오기

특정기간 내의 jtbcnew 포스트를 가져오는 함수를 작성해 보자

fb_get_post_list(pagename, from_date, to_date) 함수

```
def fb_get_post_list(pagename, from_date, to_date):
    page_id = fb_name_to_id(pagename)

    base = BASE_URL_FB_API
    node = '/%s/posts' % page_id
    fields='/?fields=id,message,link,name,type,shares,'+₩
            'created_time,comments.limit(0).summary(true),'+₩
            'reactions.limit(0).summary(true)'
    duration='&since=%s&until=%s' % (from_date, to_date)
    parameters='&limit=%s&access_token=%s' %(LIMIT_REQUEST, ACCESS_TOKEN)

    url=base + node + fields + duration + parameters

    jsonPosts = get_json_result(url)   #포트스 정보를 딕셔너리 형태로 리턴한다( data, paging)

    return jsonPosts
```

5) jtbcnews 포스트 data (응답객체) 분석 -1

http://json.parser.online.fr/

String parse: 3629 errors | JS eval fails

SyntaxError: Unexpected token ' in JSON at position 1


```
{
  'data': [
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…},
    {…}
  ],
  'paging': {
    'cursors': {
      'before': 'Q2c4U1pXNTBYM0YxWlhKNVgzTjBiM0olWDJsa0R
      5TXlOREF5TmpNME1ESTJPVGs1TVRnNk5EazNNamd6TmprME56Y3
      pNREl3T0Rrek1BOE1ZAWEJwWDNOMGIzSjVYMmxrRHlBeU5EQXlO
      ak0wTURJMk9UazVNVGhmTVRVMU5UVXpOVFl3TVRFM01qWTROUTh
      FZAEdsdFpRWloxc2VBQVE9PQZDZD',
      'after': 'Q2c4U1pXNTBYM0YxWlhKNVgzTjBiM0olWDJsa0R5
      UXlOREF5TmpNME1ESTJPVGs1TVRnNkxUWTRNVEEyTmpZAek5qQX
      dNREV6TURVeU5qSVBER0ZA3YVY5emRHOXllVjlwWkE4ZA01qUXd
      Nall6TkRBeU5qazVPVEU0WHpFMU5UTTRNVGMwTmpRMk56YzRNek
      lQQkhScGJXVUdXZAFE1bUFFPQZDZD'
    },
    'next': 'https://graph.facebook.com/v2.10/24026340269
    9918/posts?
    access_token=1428253630556936%7Cba8f09769d565fc8646398
    885bdb20ae&fields=id%2Cmessage%2Clink%2Cname%2Ctype%2C
    shares%2Creactions%2Ccreated_time%2Ccomments.limit%280
    %29.summary%28true%29.limit%280%29.summary%28true%29&s
    ince=2017-01-01&until=2017-10-
    06&limit=20&after=Q2c4U1pXNTBYM0YxWlhKNVgzTjBiM0olWDJs
    a0R5UXlOREF5TmpNME1ESTJPVGs1TVRnNkxUWTRNVEEyTmpZAek5qQ
    XdNREV6TURVeU5qSVBER0ZA3YVY5emRHOXllVjlwWkE4ZA01qUXdNa
    ll6TkRBeU5qazVPVEU0WHpFMU5UTTRNVGMwTmpRMk56YzRNeklQQkh
    ScGJXVUdXZAFE1bUFFPQZDZD'
  }
}
```


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

5) jtbcnews 포스트 data (응답객체) 분석 -2

응답객체의 data 키 아래의 정보중 저장할 데이터는 아래와 같습니다.
한번 요청에 20개씩 가져오고 있습니다.

연산자	json data 키값
글작성일	created_time
본문	message
공유수	shares > count
반응수(좋아요등)	reactions > summary > total_count
댓글수	comments > summary > total_count

응답객체의 paging 키 아래의 next 키 정보를 이용하여 다음 데이터를 요청하고
next 키의 값이 없을때 까지 계속요청합니다.

6) 전체데이타 수집(저장)

fb_get_post_list(pagename, from_date, to_date) 함수 수정

- 응답객체의 next 키 정보를 이용하여 next 키의 값이 없을때 까지 계속 요청합니다. (while문 사용)

- data 키 정보를 이용하여 저장할 데이터만 리스트에 관리합니다.
  preprocess_post(post) 함수를 만들어 원하는 데이터만 수집

- 리스트에 관리된 데이터를 json 형태로 만들어 파일로 저장합니다.

fb_get_post_list(pagename, from_date, to_date) 함수 수정

```
def fb_get_post_list(pagename, from_date, to_date):
    ... 이전부부 생략 ...
    postList = []
    isNext = True

    while isNext:
        tmpPostList = get_json_result(url)  # '리턴은 딕셔너리로써 'data' 와 'paging' 으로 구분되어짐
        for post in tmpPostList["data"]:
            postVo = preprocess_post(post)
            postList.append(postVo)

        paging = tmpPostList.get("paging").get("next")

        if paging != None:
            url = paging
        else :
            isNext = False

    # save results to file
    with open("d:/" + pagename + ".json", 'w', encoding='utf-8') as outfile:
        json_string = json.dumps(postList, indent=4, sort_keys=True, ensure_ascii=False)
        outfile.write(json_string)

    return postList
```

파일저장

```
# save results to file
with open("d:/" + pagename + ".json", 'w', encoding='utf-8') as outfile:
    json_string = json.dumps(postList, indent=4, sort_keys=True, ensure_ascii=False)
    outfile.write(json_string)
```

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

- import json 해서 사용
- 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.

preprocess_post(post)

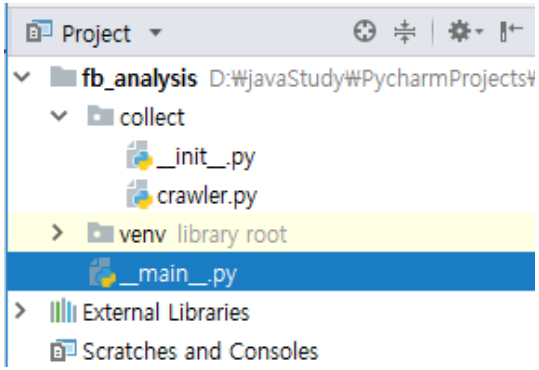
```
def preprocess_post(post) :

 # 작성일  +9시간(오차수정)
    created_time = post["created_time"]
    created_time = datetime.strptime(created_time, '%Y-%m-%dT%H:%M:%S+0000')
    created_time = created_time + timedelta(hours=+9)
    created_time = created_time.strftime('%Y-%m-%d %H:%M:%S')


    # 공유 수
    if  "shares" not in post :
        shares_count = 0
    else :
        shares_count = post["shares"]["count"]

    # 리액션 수
    if  "reactions" not in post :
        reactions_count = 0
    else :
        reactions_count = post["reactions"]["summary"]["total_count"]


  다음페이지 계속
```

preprocess_post(post)

이전페이지에서 계속

```
    # 댓글 수
    if "comments" not in post:
        comments_count = 0
    else:
        comments_count = post["comments"]["summary"]["total_count"]

    # 메세지
    if  "message" not in post :
        message_str = ""

    else :
        message_str = post["message"]

    postVo = {
                "shares_count": shares_count,
                "reactions_count": reactions_count,
                "comments_count": comments_count,
                "message_str": message_str,
                "created_time": created_time
              }
    return postVo
```

7) __main__.py 작성




```
from collect import crawler

pagename = "jtbcnews"
from_date = "2017-01-01"
to_date = "2017-02-01"

if __name__ == '__main__':

    # 수집 저장
    postList = crawler.fb_get_post_list(pagename, from_date, to_date)
    print(postList)
```

03.페이스북_데이터 수집, 분석, 시각화

01. 데이터 수집

02. 데이터 분석

03. 시각화

2 데이터 분석

2.1 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 : 논밭 -> 논 + 밭, 논밭
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

2.2 KoNLPy 설치 및 확인

1) **java 1.7 이상 설치**(반드시 JAVA_HOME 환경변수 설정 할것)

2) **jpype1 설치**

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

PIP(파이썬 패키지 관리자) 프로그램을 통해서 설치한다.

```
C:₩Users₩remys-note₩Downloads>pip install jpype1
```

Microsoft Visual C++ 14.0 required 오류가 발생하면

1)Visual Studio Build Tools 을 설치한다.

https://landinghub.visualstudio.com/visual-cpp-build-tools

2)wheel 파일을 설치한다.

https://www.lfd.uci.edu/~gohlke/pythonlibs/ 에서 파이썬 버전에 맞는 jpype1 파일을 다운받아 설치한다.

```
python -m pip install JPype1-0.6.2-cp36-cp36m-win_amd64.whl
```

3) jpype1 설치확인

```
C:₩Users₩remys-note₩Downloads>pip list
Package    Version
---------- -------
JPype1     0.6.3
numpy      1.14.3
pip        10.0.1
setuptools 39.1.0
```

4) KoNLPy 설치

```
C:₩Users₩remys-note₩Downloads>pip install KoNLPy

...
Successfully installed KoNLPy-0.4.4

C:₩Users₩remys-note₩Downloads>
```

5) KoNLPy 설치 확인

[test_konlpy.py] * 시간이 좀 걸림

```
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)
```

오류발생시

1)pyCham > File > Setting 클릭
2)왼쪽 메뉴중 Project: [프로젝트명] > Project Interpreter 을 선택한다.
3)오른쪽 상단의 + 버튼을 클릭한다.
4)설치하고 싶은 모듈을 검색하여 install 한다. → Jpype1, konlpy

2.3 빈도 분석(Frequency Analysis)

1) 빈도분석의 의미

- **앞에서 페이스북 API를 이용해 특정 페이지의 포스트 데이터들 중 message 내용을 형태소 분석을 통해 단어를 추출하고 단어를 카운트해서 단어들의 빈도를 구한다.**

- 통계 분석에 변수(단어)의 빈도를 도수라 하고 이를 표로 정리한 것을 도수분포표라고 한다

- 도수분포표는 통계 분석의 가장 기본이 되는 빈도 분석에서 가장 기초가 되는 데이터라 할 수 있다

- 통계 분석에서 빈도 분석의

• 통계 분석 기법 중 가장 기본이 되는 분석방법이다. 즉, 수집한 자료의 특성을 파악하기 위한 첫 번째 단계로 데이타의 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), 히그토그램 (Histogram) 등의 그래프를 제공하여 통계량을 보다 쉽게 이해할 수 있게 한다.

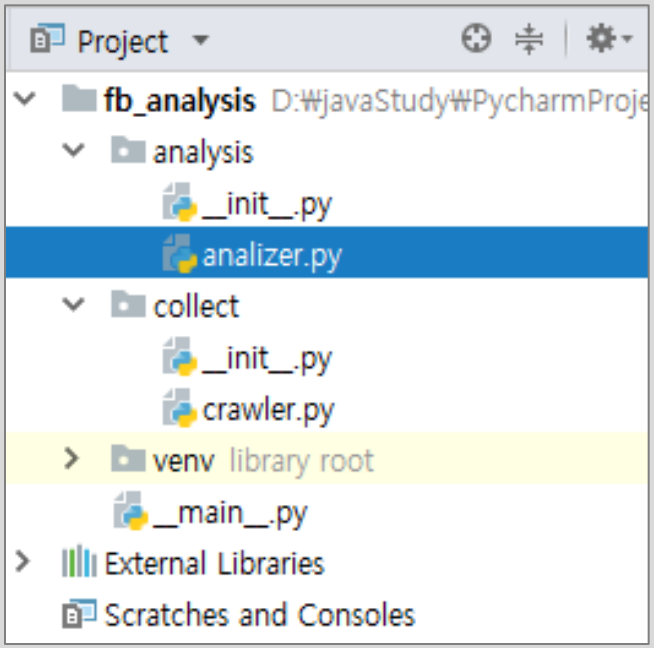
2) json_to_str(filename, key) 함수 만들기

- analysis 패키지에 analizer.py 모듈을 추가 한다.
- json_to_str(filename, key) 만든다.
- json포맷의 파일 이름을 입력 받아 각각의 아이템에서 key에 해당하는 value를 공배문자를 제거하고 하나의 문자열로 합치는 함수이다.

[analizer.py]

```
import json
import re
```

```
def json_to_str(filename, key):
    jsonfile = open(filename, 'r', encoding='utf-8')
    json_string = jsonfile.read()
    jsondata = json.loads(json_string)

    print(jsonfile)

    data = ''
    for item in jsondata:
        value = item.get(key)
        if value is None:
            continue

        data += re.sub(r'[^₩w]', '', value)
    return data
```



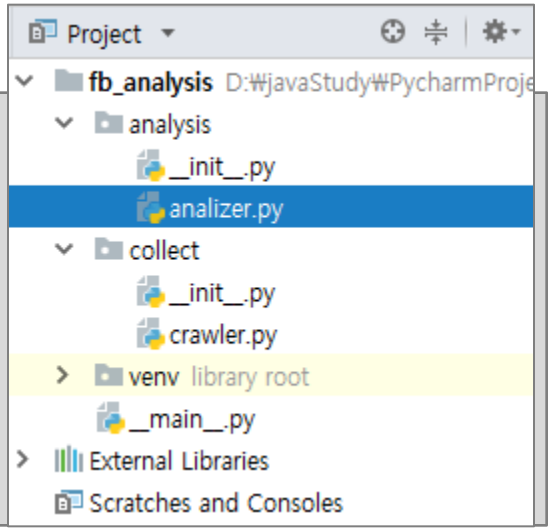
3) count_word(data) 함수 만들기

- 입력받은 문자열에서 단어를 추출하고 빈도수를 세서 단어별 빈도수와 dict 타입으로 리턴하는 함수이다.

[analizer.py]

```
from konlpy.tag import Twitter
from collections import Counter
```

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

    count = Counter(nouns)
    print(type(count))
    return count
```



4) __main__.py 작성

__main__.py 에 다음 코드를 추가한다.

```
# 분석
data = analizer.json_to_str("d:/" + pagename + ".json", 'message_str')
wordList = analizer.count_wordfreq(data)
print(wordList)
```

03.페이스북_데이터 수집, 분석, 시각화

01. 데이터 수집

02. 데이터 분석

03. 시각화

3.1 matplot을 이용한 그래프 그리기

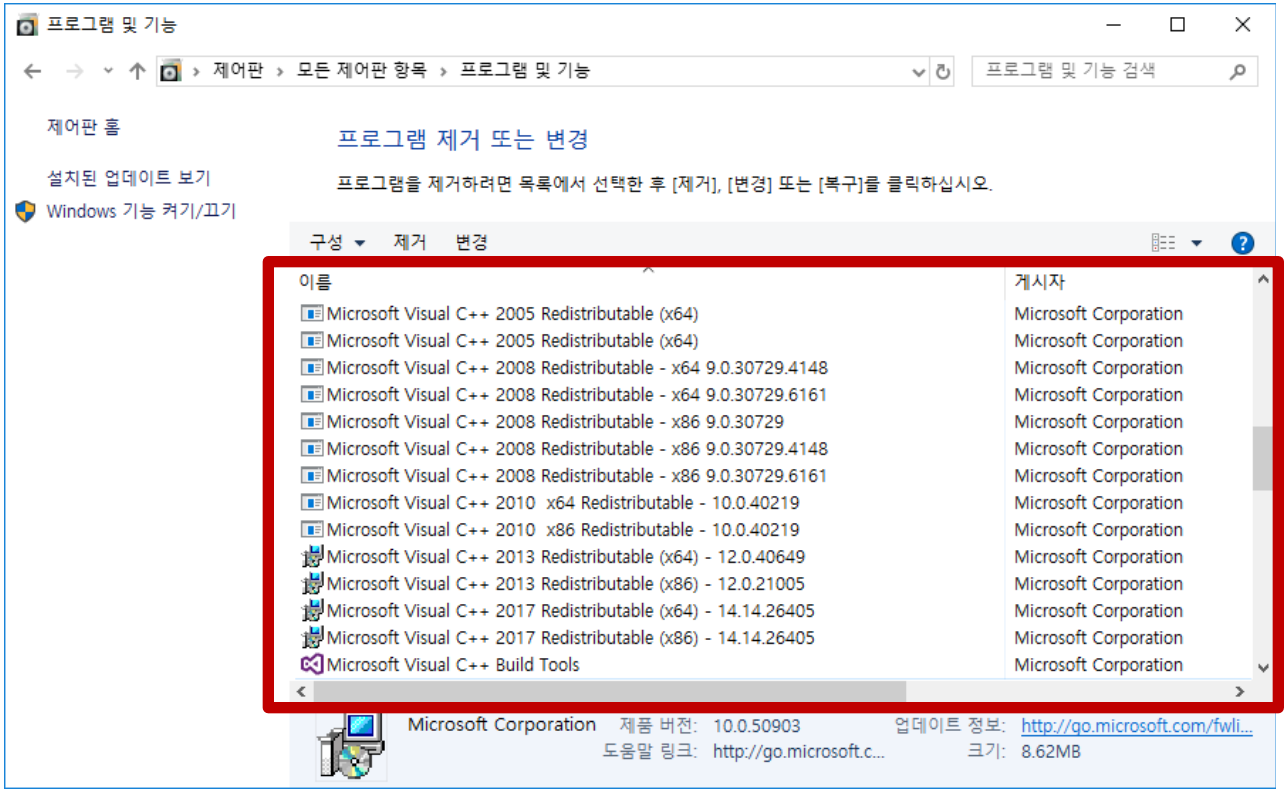
1) 파이썬 그래프 모듈: matplotlib

- 파이썬에서 그래프 기능 지원모듈 중 가장 일반적인 모듈

- 설치

•파이썬 3.4(windows) 이상인 경우 Microsoft Visual C++ (2008 or 2010) 배포 패키지(Redistiributable)가 필요하다.

•확인



- 설치

•Microsoft Visual C++ (2008,2010) 배포 패키지(Redistiributable)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_x86.exe

- 설치

•Microsoft Visual C++ (2008,2010) 배포 패키지(Redistiributable)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_x86.exe

- 설치

```
C:₩Users₩remys-note₩Downloads>pip install matplotlib
```

2) show_graph_bar(dictWords, pagename) 함수 만들기

- visualize 패키지에 visualizer.py 모듈을 추가 한다.
- **show_graph_bar(dictWords)** 만든다.

dict 형식의 단어들을 받아서 그래프로 출력하는 함수이다.

[visualizer.py]

```
import matplotlib.pyplot as plt
from matplotlib import font_manager
```

```
def show_graph_bar(dictWords, pagename):

    # 한글처리
    font_filename = 'c:/Windows/fonts/malgun.ttf'
    font_name = font_manager.FontProperties(fname=font_filename).get_name()
    print(font_name)
    plt.rc('font', family=font_name)

    #라벨처리
    plt.xlabel("주요단어")
    plt.ylabel("빈도수")
    plt.grid(True)



    다음페이지 계속
```

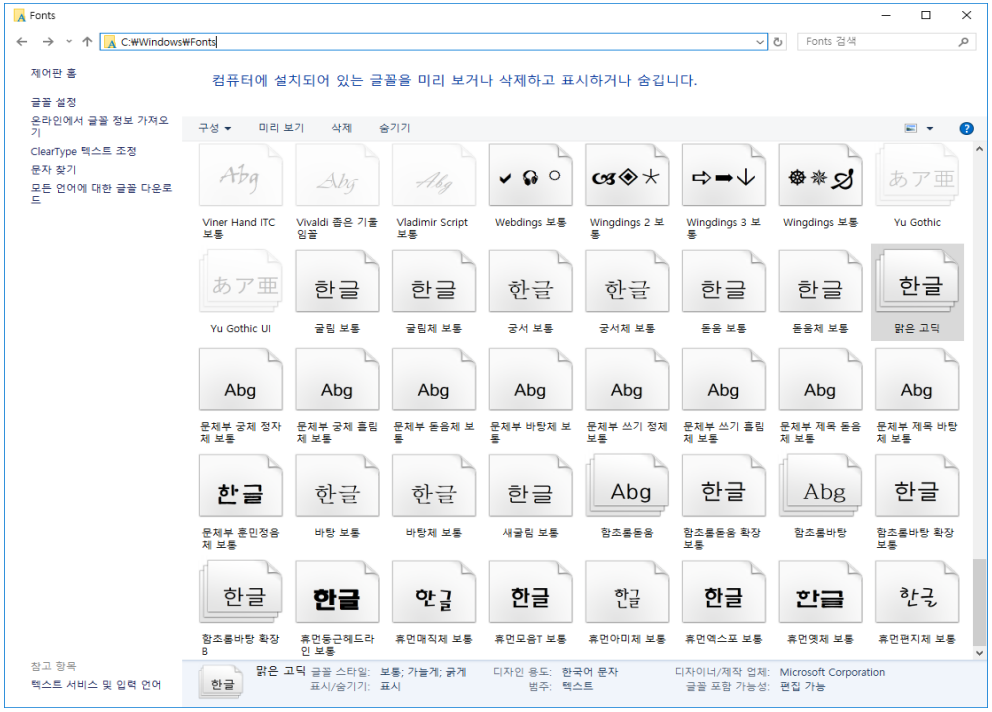
```
이전페이지에서 계속

#데이타 대입
dict_keys = dictWords.keys()
dict_values = dictWords.values()

plt.bar(range(len(dictWords)), dict_values, align='center')
plt.xticks(range(len(dictWords)), list(dict_keys), rotation=70)

plt.show()

save_filename = "D:/javaStudy/fb/%s_bar_graph.png" % pagename
plt.savefig(save_filename, dpi=400, bbox_inches='tight')
```

3) 그래프 한글처리

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

```
# 한글처리
font_filename = 'c:/Windows/fonts/malgun.ttf'
font_name = font_manager.FontProperties(fname=font_filename).get_name()
print(font_name)
plt.rc('font', family=font_name)
```

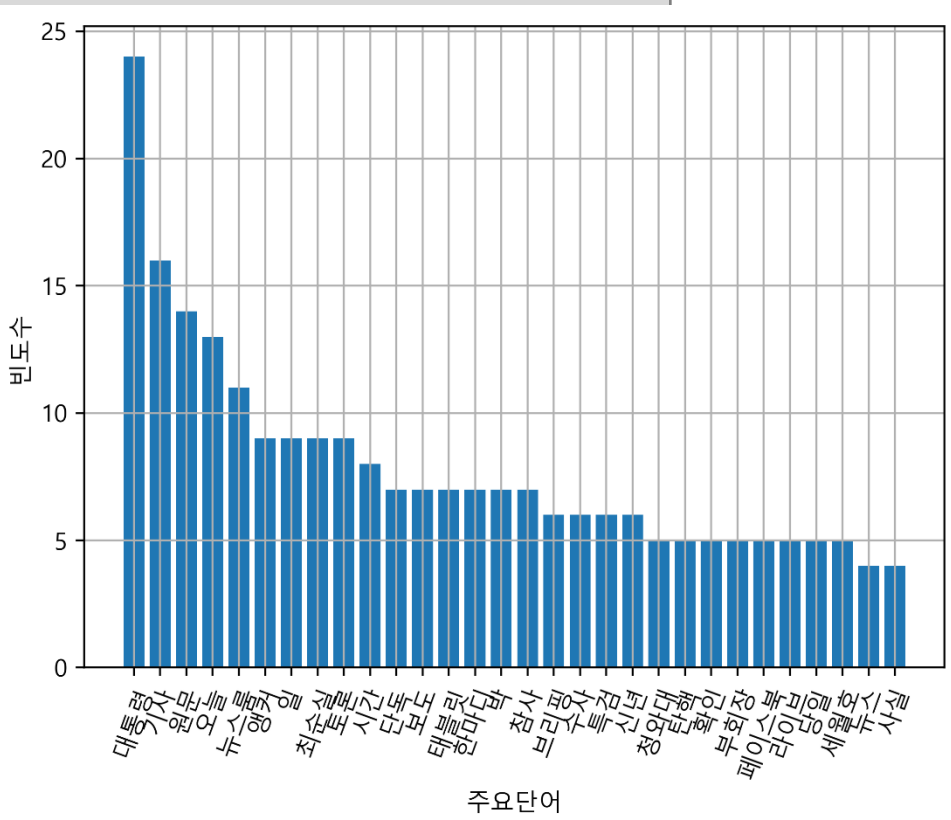
C:₩Windows₩fonts 내용



3) __main__.py 작성

__main__.py 에 다음 코드를 추가한다.

```
# 시각화
# 데이타 가공
dictWords = dict(wordList.most_common(30))

#막대그래프
visualizer.show_graph_bar(dictWords)
```



3.2 pytagcloud를 이용한 워드 클라우드 그리기

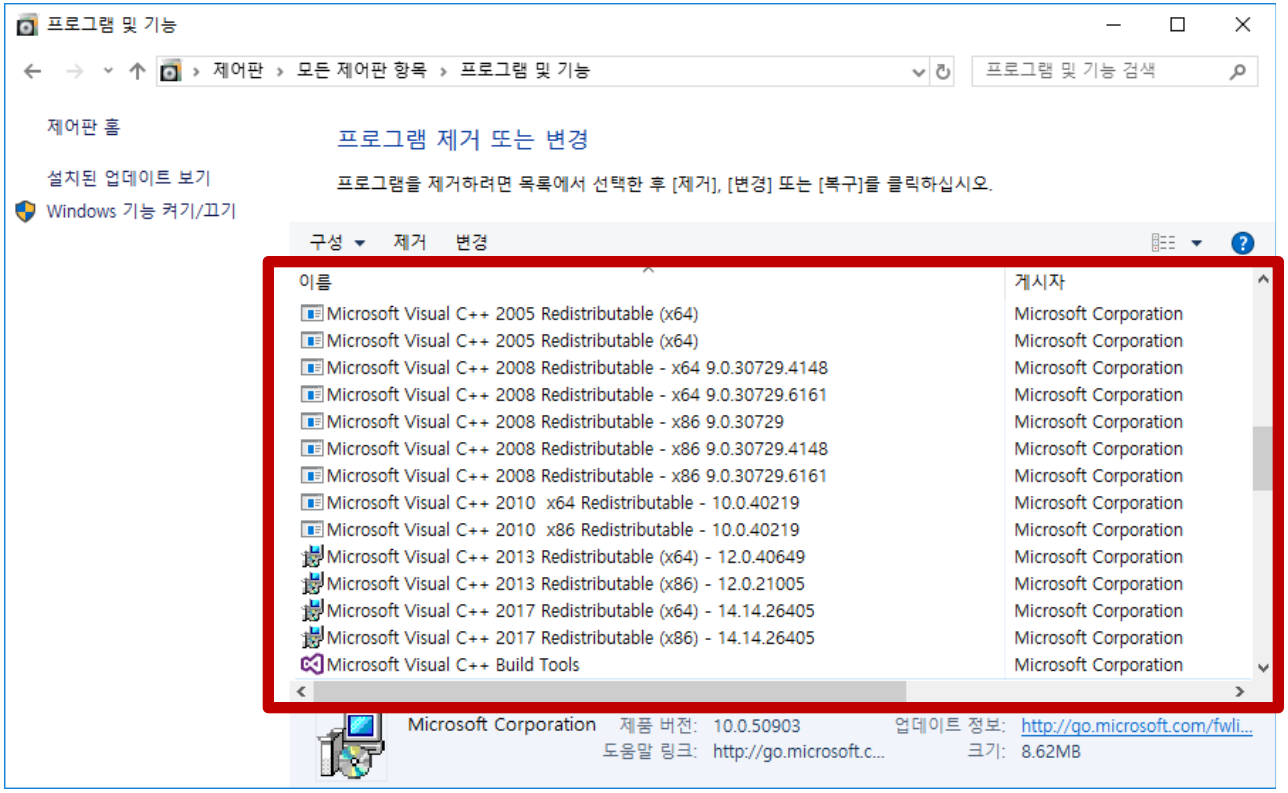
1) 파이썬 그래프 모듈: matplotlib

- 파이썬에서 그래프 기능 지원모듈 중 가장 일반적인 모듈

- 설치

•파이썬 3.4(windows) 이상인 경우 Microsoft Visual C++ (2008 or 2010) 배포 패키지(Redistiributable)가 필요하다.

•확인



2) wordcloud(dictWords, pagename) 함수 만들기

- visualize 패키지에 visualizer.py 모듈에 **wordcloud(dictWords, pagename) 함수를 만든다.**

[visualizer.py]

```
import pytagcloud
import webbrowser
```

```
# 워드크라우드
def wordcloud(dictWords, pagename):
    print(type(dictWords))
    print(dictWords)
    taglist = pytagcloud.make_tags(dictWords.items(), maxsize=80)

    save_filename = "D:/javaStudy/fb/%s_wordcloud.jpg" % pagename
    pytagcloud.create_tag_image(
        taglist,
        save_filename,
        size=(800, 600),
        fontname='korean',
        rectangular=False
    )
    webbrowser.open(save_filename)
```

3) 한글 처리

- 현재 프로젝트의 fonts 폴더로 이동합니다.

  D:₩javaStudy₩PycharmProjects₩fb_analysis**₩venv₩Lib₩site-packages₩pytagcloud₩fonts**

- 이곳에 사용할 한글 폰트를 복사합니다.

  C:₩Windows₩Fonts 의 맑은고딕 폰트 패밀리를 복사합니다.

- font.json 파일을 수정합니다. 아래의 빨간색 영역 추가

  D:₩javaStudy₩PycharmProjects₩fb_analysis**₩venv₩Lib₩site-packages₩pytagcloud₩fonts**

[fonts.json]

```
[
    {
        "name": "korean",
        "ttf": "malgun.ttf",
        "web": "http://fonts.googleapis.com/css?family=Nobile"
    },
    {
        "name": "Nobile",
        "ttf": "nobile.ttf",
                        [아래 생략]
```

4) __main__.py 작성

__main__.py 에 다음 코드를 추가한다.

```
# 워드크라우드
    visualizer.wordcloud(dictWords, pagename)
```