네이버 서비스API 수정

#1 모듈 입력

In [ ]:
```
import re
import json
import math
import datetime
import requests
import urllib.request
import urllib.error
import urllib.parse
import lxml
from bs4 import BeautifulSoup

#https://developers.naver.com/main/ 사이트에서 애플리케이션 등록하여 Naver Development ID/SECRET 필
naver_client_id = "cjqqQMWYQo3a4nuJ9AEE"
naver_client_secret = "Z1ysOpQY7A"
```

#2 시작 프로그램

```
# 프로그램 시작
if __name__ == '__main__':
    no = 0              # 몇 개의 포스트를 지정하였는지 카운트
    query = "컴퓨터"    # 검색 단어
    display = 10        # 검색 건수 단위
    start = 1           # 검색 시작
    sort = "date"       # 정렬 옵션: sim(유사도, 기본값), date(날짜순)

    #검색 데이터 파일로 저장
    fs = open(query + ".txt", 'a', encoding='utf-8')

    # 검색 결과 1000개로 제한 함수 정의: get_blog_count(query, display)
    blog_count = get_blog_count(query, display)

    fs.close()
```

#3 get_blog_count 함수 정의: 블로그 카운트

```python
### get_blog_count(query, display) 함수 정의
def get_blog_count(query, display):

    # encode_query <- query 파싱
    encode_query = urllib.parse.quote(query)

    # search_url <- OpenApi주소 + encode_query
    search_url = "https://openapi.naver.com/v1/search/blog?query=" + encode_query

    # search_url로 요청
    request = urllib.request.Request(search_url)

    # request <- ID/SECRET
    request.add_header("X-Naver-Client-Id",naver_client_id)
    request.add_header("X-Naver-Client-Secret",naver_client_secret)

    # response <- request로 요청한 자료
    response = urllib.request.urlopen(request)

    # response_code <- response.getcode(접속상황 코드)
    response_code = response.getcode()

    if response_code == 200:
        # response_body <- response.read(검색자료)
        response_body = response.read()

        # response_body_dict(딕셔너리 변수) <- json형식 자료
        response_body_dict = json.loads(response_body.decode('utf-8'))

        print("lastBuildDate:" + str(response_body_dict['lastBuildDate'])) # 마지막 검색일자(lastBu
        print("total:" + str(response_body_dict['total']))                 # 검색 전체 개수(total)
        print("start:" + str(response_body_dict['start']))                 # 검색 문서 시작(start)
        print("display:" + str(response_body_dict['display']))             # 검색 개수(display)

        if response_body_dict['total'] == 0:
            # 검색한 결과가 하나도 없으면 blog_count <- 0
            blog_count = 0
        else:
            # 제한된 개수로 계산된 결과를 정수로 반환하는 모듈(math.ceil)
            blog_total = math.ceil(response_body_dict['total'] / int(display))
            if blog_total >= 1000:
                # 1000개 제한 결과 제공
                blog_count = 1000
            else:
                blog_count = blog_total

            print("블로그 전체수:" + str(blog_total))
            print("블로그 갯수:" + str(blog_count))

    return blog_count
```

#4 get_blog_post 함수 정의: 블로그 게시물 출력

```
### get_blog_post(query, display, start_index, sort) 함수 정의
def get_blog_post(query, display, start_index, sort):
    # no, fs 전역변수 선언
    global no, fs

    # encode_query <- query 파싱(-get_blog_count() 함수 동일-)
    encode_query = urllib.parse.quote(query)

    # search_url <- OpenApi주소 + encode_query + 추가 부분
    search_url = "https://openapi.naver.com/v1/search/blog?query=" + encode_query + ₩
                 "&display=" + str(display) + "&start=" + str(start_index) + "&sort=" + sort

    # search_url로 요청
    request = urllib.request.Request(search_url)

    # request <- ID/SECRET
    request.add_header("X-Naver-Client-Id",naver_client_id)
    request.add_header("X-Naver-Client-Secret",naver_client_secret)

    # response <- request로 요청한 자료
    response = urllib.request.urlopen(request)

    # response_code <- response.getcode(정상호출 코드)
    response_code = response.getcode()

    if response_code == 200:
        # response_body <- response.read(검색자료)
        response_body = response.read()

        # response_body_dict(딕션어리 변수) <- json형식 자료
        response_body_dict = json.loads(response_body.decode('utf-8'))

        #JSON의 items 속성 분해 items:= title, link, description, bloggername, bloggerlink, post_date
        for item_index in range(0, len(response_body_dict['items'])):
            try:
                #json파일의 불필요한 tag 삭제
                remove_html_tag = re.compile('<.*?>')
                #item_index=0: title, remove_html_tag
                title = re.sub(remove_html_tag, '', response_body_dict['items'][item_index]['title'] )
                #item_index=1: link, replace('amp;','')
                link = response_body_dict['items'][item_index]['link'].replace("amp;", "")
                #item_index=2: description, remove_html_tag
                description = re.sub(remove_html_tag, '', response_body_dict['items'][item_index]['description'] )
                #item_index=3: blogger_name
                blogger_name = response_body_dict['items'][item_index]['bloggername']
                #item_index=4: blogger_link
                blogger_link = response_body_dict['items'][item_index]['bloggerlink']
                #item_index=5: post_date
                post_date = response_body_dict['items'][item_index]['postdate']
                no += 1
                fs.write(str(no) + " 건" + title + "₩n" + link + "₩n"+ description + "₩n"+ blogger_name + "₩n" ₩
                         + blogger_link + "₩n" + post_date + "₩n" + "----------------------------------------" + '₩n')

                #print('#' + str(no))
                #print('제목: ' + title)
                #print('링크: ' + link)
                #print('설명: ' + description)
                #print('작성자: ' + blogger_name)
                #print('블로거 링크: ' + blogger_link)
                #print('포스트 일자: ' + post_date)
            except:
                item_index += 1
```

#5 검색어 입력

```
# 프로그램 시작
if __name__ == '__main__':
    query = input("검색 질의: ")
    #query = "컴퓨터"    # 검색 단어
    no = 0              # 몇 개의 포스트를 지정하였는지 카운트
    display = 10        # 검색 건수 단위
    start = 1           # 검색 시작
    sort = "date"       # 정렬 옵션: sim(유사도, 기본값), date(날짜순)

    #검색 데이터 파일로 저장
    fs = open(query + ".txt", 'a', encoding='utf-8')

    # 검색 결과 1000개로 제한 함수 정의: get_blog_count(query, display)
    blog_count = get_blog_count(query, display)
    for start_index in range(start, blog_count + 1, display):
        get_blog_post(query, display, start_index, sort)

    fs.close()
```