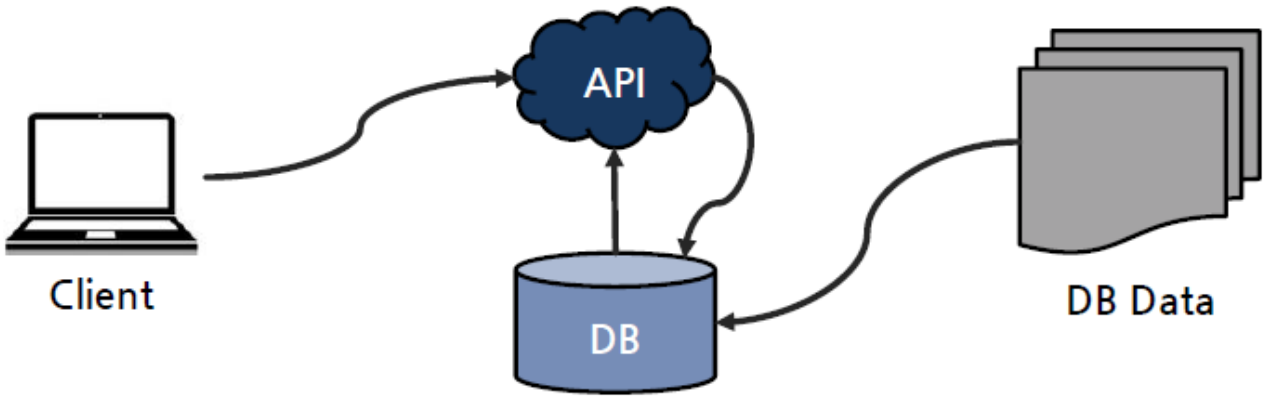
05. 공공DB API와 SNS 수집

차례

1. Open API와 Rest API

- Open API vs Rest API

API	▪ Application Programming Interface의 약자 ▪ 특정 프로그램을 만들기 위해 제공되는 모듈(함수 등)



1. Open API와 Rest API

- Open API vs Rest API

Open API

- 누구나 사용할 수 있도록 공개된 API
- 주로 Rest API 기술을 많이 사용

Rest API

- Representational State Transfer API의 약자
- HTTP 프로토콜을 통해서 정보를 제공하는 함수
- 실질적인 API 사용은 정해진 구조의 URL 문자열 사용

일반적으로 XML, JSON의 형태로 응답 출력

1. Open API와 Rest API

- 웹 API
  - 웹 API는 일반적으로 HTTP 통신을 사용하는데 사용
  - 지도, 검색, 주가, 환율 등 다양한 정보를 가지고 있는 웹 사이트의 기능을 외부에서 쉽게 사용할 수 있도록 사용 절차와 규약을 정의한 것

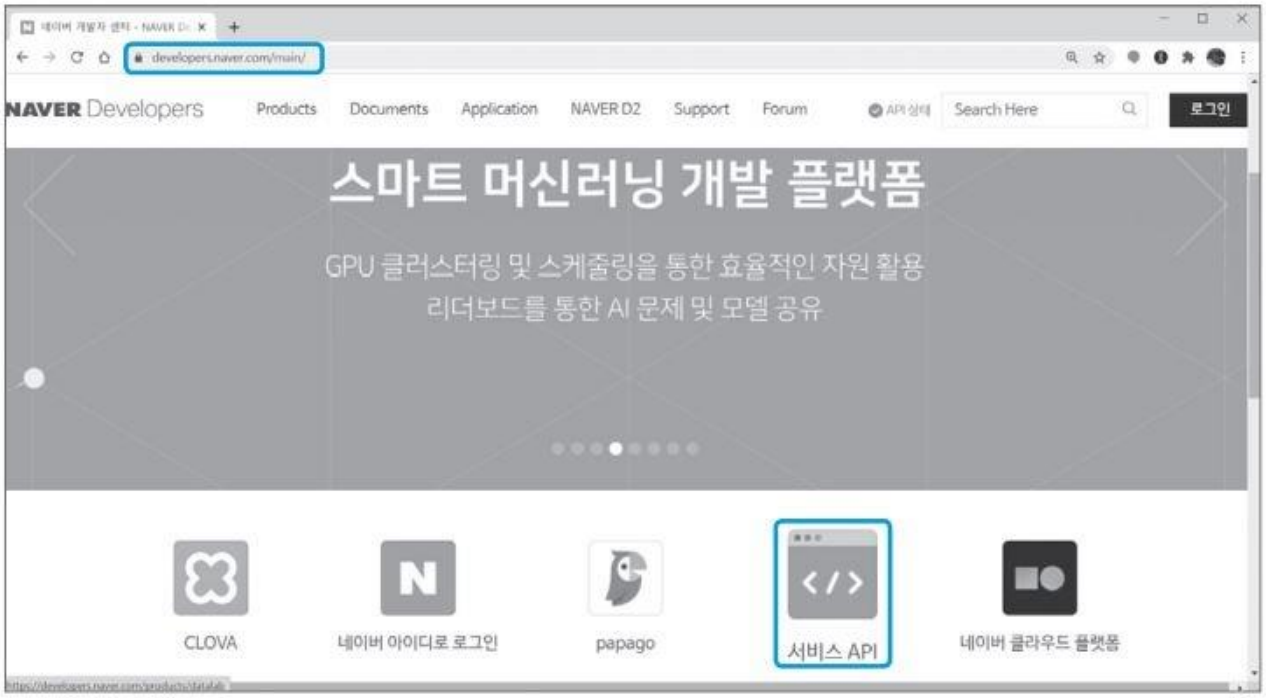
종류	주소
네이버 개발자 센터	https://developers.naver.com
카카오 앱 개발 플랫폼 서비스	https://developers.kakao.com
페이스북 개발자 센터	https://developers.facebook.com
트위터 개발자 센터	https://developer.twitter.com
공공데이터포털	https://www.data.go.kr
세계 날씨	http://openweathermap.org
유료/무료 API 스토어	http://mashup.or.kr http://www.apistore.co.kr/api/apiList.do

02. 네이버 API를 이용한 크롤링

- 네이버 개발자 가입

1. 네이버 개발자 센터 접속하기

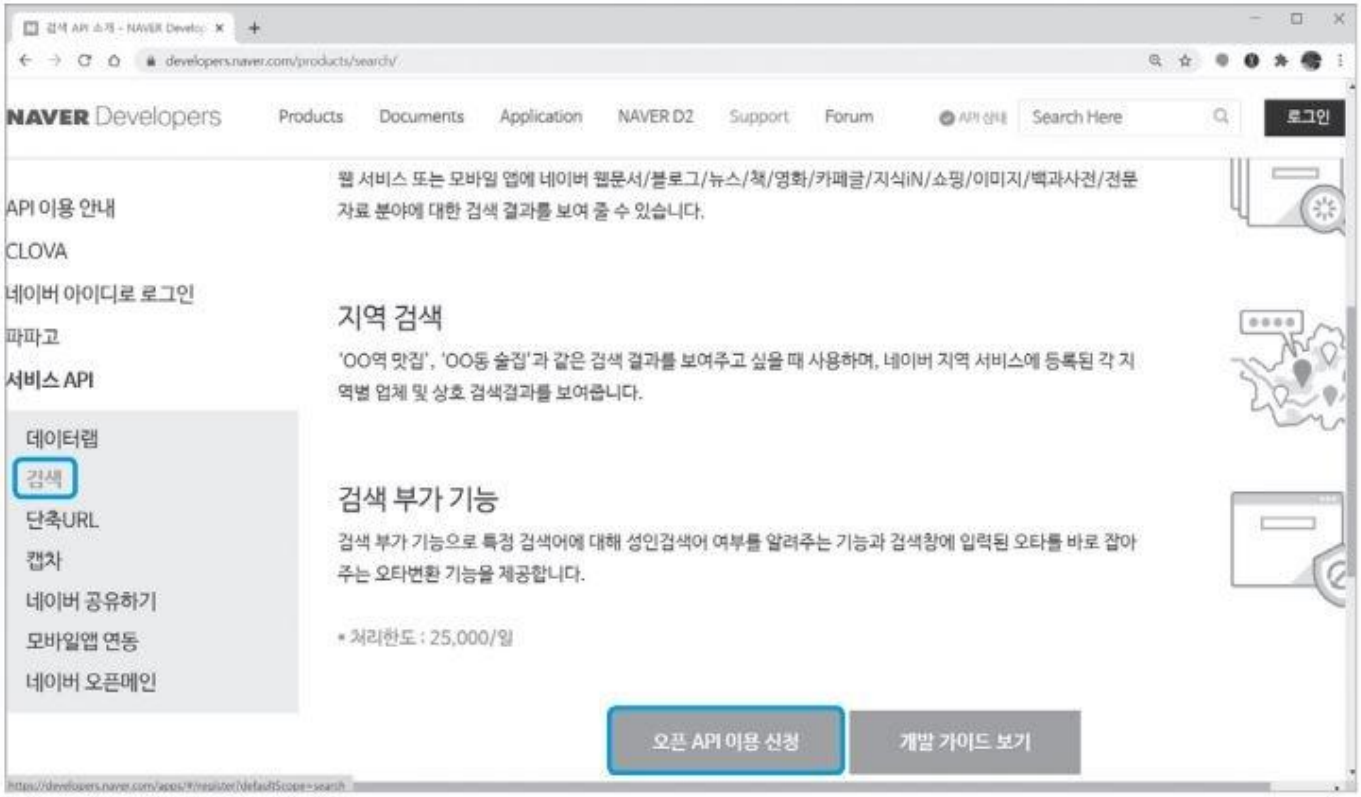
– 네이버 개발자 센터(https:// developers.naver.com)에 접속



02. 네이버 API를 이용한 크롤링

- 네이버 개발자 가입

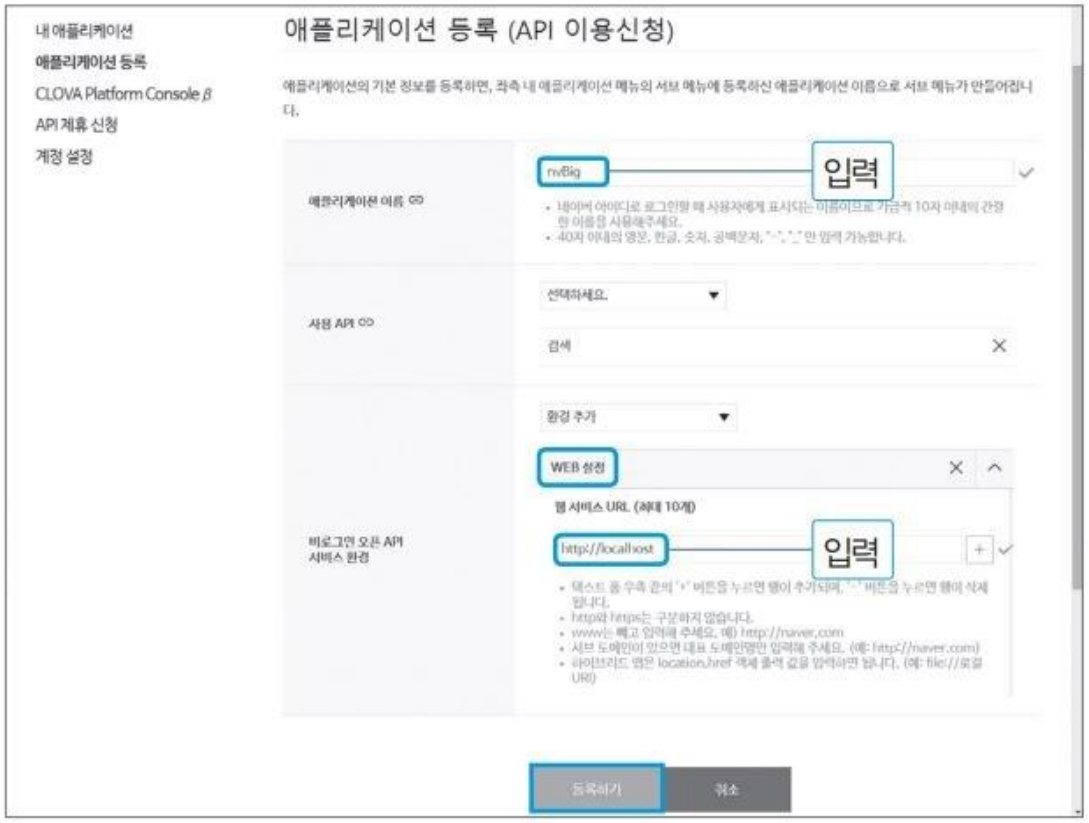
2. 오픈 API 이용 신청하기



02. 네이버 API를 이용한 크롤링

- 네이버 개발자 가입

3. 애플리케이션 등록하기



02. 네이버 API를 이용한 크롤링

- 네이버 개발자 가입

4. 애플리케이션 정보 확인하기

02. 네이버 API를 이용한 크롤링

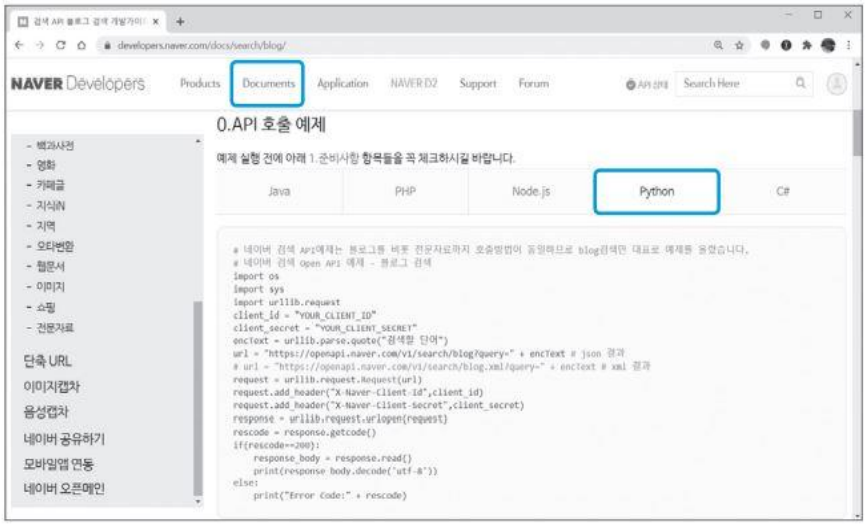
- 네이버 개발자 가입

4. 애플리케이션 정보 확인하기

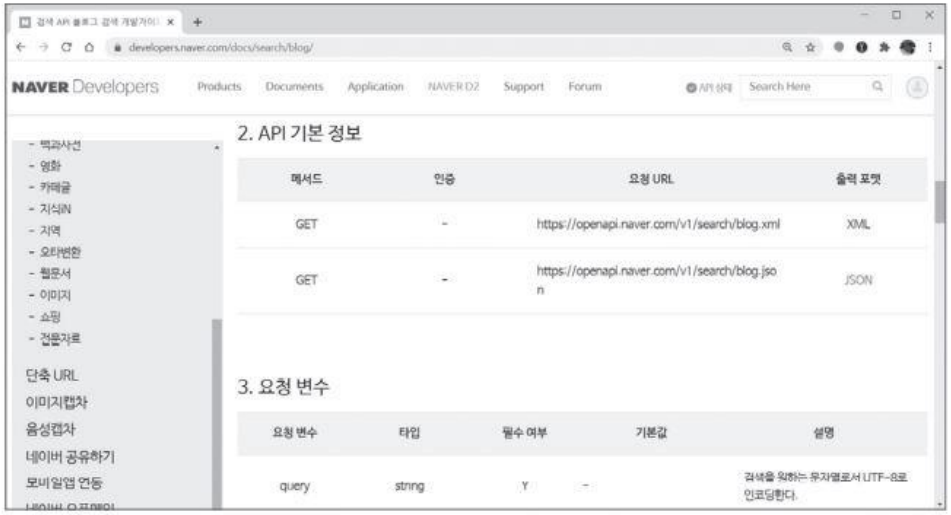
02. 네이버 API를 이용한 크롤링

- 네이버 개발자 가입

5. 검색 API 이용 안내 페이지 확인하기



(a) 파이썬 코드로 확인



(b) API 기본 정보와 요청 변수 확인

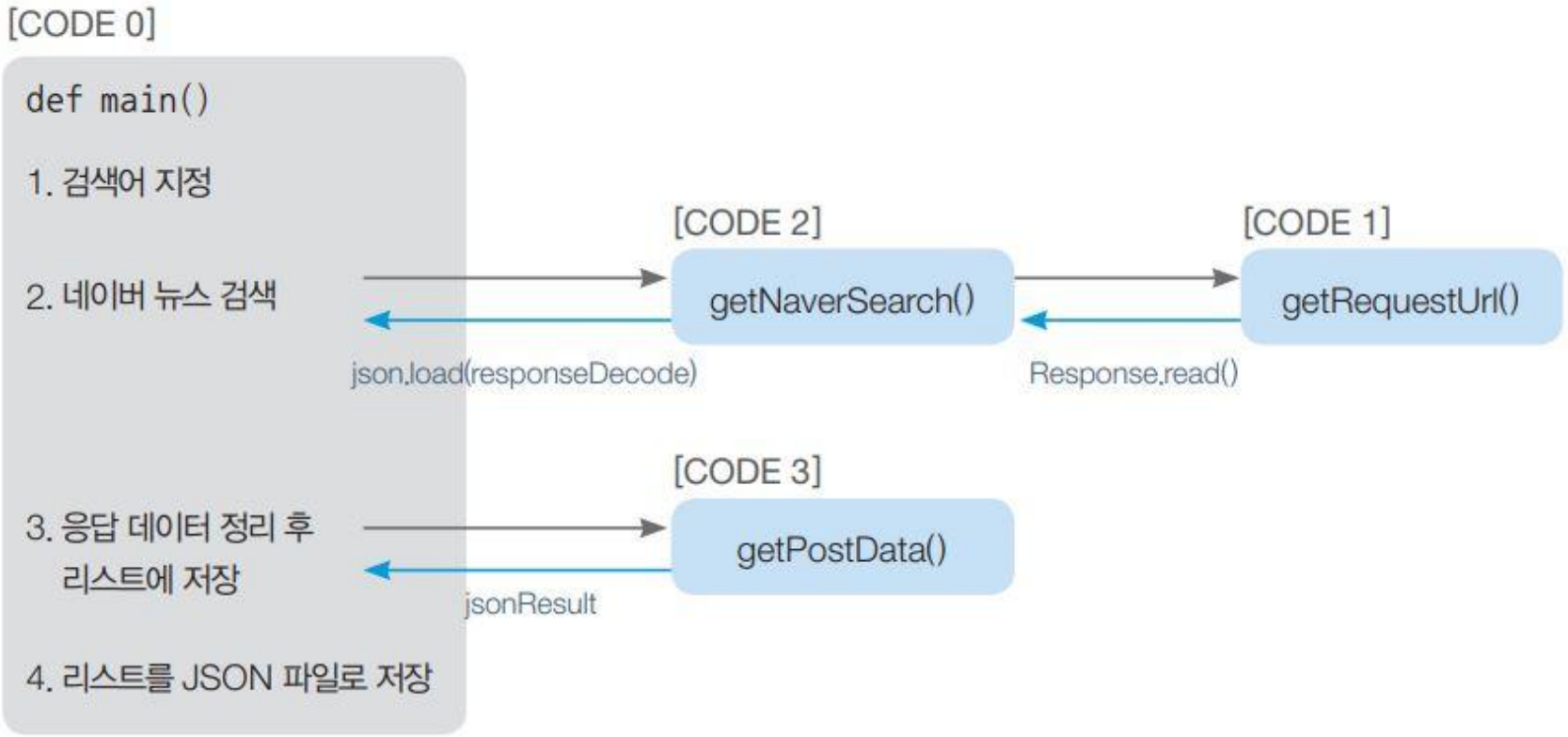
02. 네이버 API를 이용한 크롤링

- 네이버 뉴스 크롤링 : 1. 전체 작업 설계하기

작업 설계	사용할 코드
1. 검색어 지정하기	srcText = '월드컵'
2. 네이버 뉴스 검색하기	**getNaverSearch()**
2.1 url 구성하기	url = base + node + srcText
2.2 url 접속과 검색 요청하기	urllib.request.urlopen()
2.3 요청 결과를 응답 JSON으로 받기	json.load()
3. 응답 데이터를 정리하여 리스트에 저장하기	**getPostData()**
4. 리스트를 JSON 파일로 저장하기	json.dumps()

02. 네이버 API를 이용한 크롤링

- 네이버 뉴스 크롤링 : 2. 프로그램 구성 설계하기



02. 네이버 API를 이용한 크롤링

네이버 뉴스 크롤링 : 3. 함수 설계하기

- [CODE 0] 전체 작업 스토리를 설계

- 지역 변수
  - node: 네이버 검색 API에서 검색할 대상 노드([표 5-2] 참고)
  - srcText: 사용자 입력으로 받은 검색어 저장
  - cnt: 검색 결과 카운트
  - jsonResult: 검색 결과를 정리하여 저장할 리스트 객체
  - jsonResponse: 네이버 뉴스 검색에 대한 응답을 저장하는 객체
  - total: 전체 검색 결과 개수
  - post: 응답받은 검색 결과 중에서 한 개를 저장한 객체
  - items: 전체 응답 검색 결과로 내부 항목은 title, originallink, link, description, pubDate
  - jsonFile: JSON 파일에 저장할 데이터를 담은 객체

- 메서드
  - input('검색어를 입력하세요: '): 사용자로부터 입력을 받는다.
  - getNaverSearch(node, srcText, 1, 100): 1부터 100개의 검색 결과를 처리한다([CODE 2]).
  - getPostData(): 검색 결과 한 개를 처리한다([CODE 2]).
  - json.dumps(): 객체를 JSON 형식으로 변환

02. 네이버 API를 이용한 크롤링

▪ 네이버 뉴스 크롤링 : 3. 함수 설계하기

```
01 def main():
02     node = 'news' #크롤링할 대상
03     srcText = input('검색어를 입력하세요: ')
04     cnt = 0
05     jsonResult = []
06
07     jsonResponse = getNaverSearch(node, srcText, 1, 100) #[CODE 2]
08     total = jsonResponse['total']
09
10     while ((jsonResponse != None) and (jsonResponse['display'] != 0)):
11         for post in jsonResponse['items']:
12             cnt += 1
13             getPostData(post, jsonResult, cnt) #[CODE 3]
14
15         start = jsonResponse['start'] + jsonResponse['display']
16         jsonResponse = getNaverSearch(node, srcText, start, 100) #[CODE 2]
17
18     print('전체 검색 : %d 건' %total)
19
20     with open('%s_naver_%s.json' % (srcText, node), 'w', encoding = 'utf8') as outfile:
21         jsonFile = json.dumps(jsonResult, indent = 4, sort_keys = True, ensure_ascii = False)
22           outfile.write(jsonFile)
23
24     print("가져온 데이터 : %d 건" %(cnt))
25     print('%s_naver_%s.json SAVED' % (srcText, node))
```

02행 : 네이버 뉴스 검색을 위해 검색 API 대상을 'news'로 설정

03행 : 파이썬 셸 창에서 검색어를 입력받아 srcText에 저장

07행 : getNaverSearch() 함수를 호출하여 start = 1, display= 100에 대한 검색 결과를 반환받아 jsonResponse에 저장

10~16행 : 검색 결과jsonResponse에 데이터가 있는 동안 for문(11~13행) 으로 검색 결과를 한 개씩 처리하는 작업getPostData( )을 반복
반복 작업이 끝나면 다음 검색 결과 100개를 가져오기 위해 start 위치를 변경

15행 : getNaverSearch() 함수를 호출하여 새로운 검색 결과를
jsonResponse에 저장하고16행 for문11~13행을 다시 반복

20~23행 : 파일 객체를 생성하여 정리된 데이터를 JSON 파일에 저장

01. 네이버 API를 이용한 크롤링

- 네이버 뉴스 크롤링 : 3. 함수 설계하기

표 5-2 네이버 검색 API 개발자 가이드(https://developers.naver.com/docs/search/news)

구분	내용 및 설명	
URL	뉴스	https://openapi.naver.com/v1/search/**news**.json
	블로그	https://openapi.naver.com/v1/search/**blog**.json
	카페	https://openapi.naver.com/v1/search/**cafearticle**.json
	영화	https://openapi.naver.com/v1/search/**movie**.json
	쇼핑	https://openapi.naver.com/v1/search/**shop**.json
요청 변수	query	검색을 원하는 문자열이며 UTF-8로 인코딩한다.
	start	검색 시작 위치로 최대 1000까지 가능하다. 1(기본값)~1000(최대값)
	display	검색 결과 출력 건수를 지정한다. 10(기본값)~100(최대값)
응답 변수	items	검색 결과로 title, originallink, link, description, pubDate를 포함한다.
	title	검색 결과 문서의 제목이다.
	link	검색 결과 문서를 제공하는 네이버의 하이퍼텍스트 link다.
	originallink	검색 결과 문서를 제공하는 언론사의 하이퍼텍스트 link다.
	description	검색 결과 문서의 내용을 요약한 정보다.
	pubDate	검색 결과 문서가 네이버에 제공된 시간이다.

01. 네이버 API를 이용한 크롤링

네이버 뉴스 크롤링 : 3. 함수 설계하기

2. [CODE 1] url 접속을 요청하고 응답을 받아서 반환하는 부분을 작성

- 매개변수

  url: 네이버 뉴스 검색(' 올림픽')에 대한 url

- 지역 변수

  req: url 접속 요청(request) 객체

  app_id: 네이버 개발자로 등록하고 받은 Client ID

  app_secret: 네이버 개발자로 등록하고 받은 Client Secret

  response: 네이버 서버에서 받은 응답을 저장하는 객체

- 메서드

  urllib.request.Request(): urllib 패키지의 request 모듈에 있는 Request() 함수로 네이버 서버에 보낼 요청(request) 객체를 생성

  Request.add_header(): 서버에 보내는 요청 객체에 헤더 정보를 추가

  urllib.request.urlopen(): 서버에서 받은 응답을 변수에 저장하기 위해 메모리로 가져오는 urllib 패키지의 request 모듈에 있는 함수

  response.getcode(): 요청 처리에 대한 응답 상태를 확인하는 response 객체의 멤버 함수로 상태 코드가 200이면 요청 처리 성공을 나타냄

  datetime.datetime.now(): 현재 시간을 구하는 함수

  response.read().decode('utf-8'): utf-8 형식으로 문자열을 디코드

01. 네이버 API를 이용한 크롤링

- 네이버 뉴스 크롤링

3. 함수 설계하기

```
01  def getRequestUrl(url):
02      req = urllib.request.Request(url)
03      req.add_header("X-Naver-Client-Id", client_id)
04      req.add_header("X-Naver-Client-Secret", client_secret)
05
06      try:
07          response = urllib.request.urlopen(req)
08          if response.getcode() == 200:
09              print("[%s] Url Request Success" % datetime.datetime.now())
10              return response.read().decode('utf-8')
11      except Exception as e:
12          print(e)
13          print("[%s] Error for URL : %s" % (datetime.datetime.now(), url))
14          return None
```

02행 : 매개변수로 받은 url에 대한 요청을 보낼 객체를 생성

03~04행 : API를 사용하기 위한 Client ID와 Client Secret 코드를 요청 객체 헤드에 추가

07행 : 요청 객체를 보내고 그에 대한 응답을 받아 response 객체에 저장

08~10행 : getcode()로 response 객체에 저장된 코드를 확인
200이면 요청이 정상처리된 것이므로 성공 메시지를 파이썬 셸 창에 출력하고 응답을 utf-8 형식으로 디코딩하여 반환

11~14행
요청이 처리되지 않은 예외 사항 exception이 발생하면 에러 메시지를 파이썬 셸 창에 출력

01. 네이버 API를 이용한 크롤링

▪ 네이버 뉴스 크롤링 : 함수 설계하기

3. [CODE 2] 네이버 뉴스 검색 url을 만들고 [CODE 1]의 getRequestUrl(url)을 호출하여 반환받은 응답 데이터를 파이썬 json 형식으로 반환하는 부분

- 매개변수

  node: 네이버 검색 API를 이용하여 검색할 대상 노드(news, blog, cafearticle, movie, shop 등)

  srcText: 검색어

  page_start: 검색 시작 위치(1~1000)

  display: 출력 건수(10~100)

- 지역 변수

  base: 검색 url의 기본 주소

  node: 검색 대상에 따른 json 파일 이름

  parameter: url에 추가할 검색어와 검색 시작 위치, 출력 건수 등의 매개변수

  responseDecode: getRequestUrl(url)을 호출하여 반환받은 응답 객체(utf-8로 디코드)

- 메서드

  getRequestUrl(url): [CODE1]을 호출하여 url 요청에 대한 응답을 받음

  json.loads(responseDecode): 응답 객체를 파이썬이 처리할 수 있는 JSON 형식으로 변환

01. 네이버 API를 이용한 크롤링

▪ 네이버 뉴스 크롤링 : 3. 함수 설계하기

```
01  def getNaverSearch(node, srcText, page_start, display):
02      base = "https://openapi.naver.com/v1/search"
03      node = "/%s.json" % node
04      parameters = "?query=%s&start=%s&display=%s" % (urllib.parse.quote(srcText), start, display)
05
06      url = base + node + parameters
07       responseDecode = getRequestUrl(url) #[CODE 1]
08
09      if (responseDecode == None):
10          return None
11      else:
12          return json.loads(responseDecode)
```

02~06행 : [표 5-2]의 네이버 검색 API 정보에 따라 요청 URL을 구성

07행 : 완성한 url을 이용하여 getRequestUrl() 함수를 호출하여 받은 utf-8 디코드 응답을responseDecode에 저장

12행 : 서버에서 받은 JSON 형태의 응답 객체를 파이썬 객체로 로드하여 반환

01. 네이버 API를 이용한 크롤링

네이버 뉴스 크롤링 : 3. 함수 설계하기

4. [CODE 3] JSON 형식의 응답 데이터를 필요한 항목만 정리하여 딕셔너리 리스트인 jsonResult를 구성하고 반환하도록 작성

- 매개변수

  post: 응답으로 받은 검색 결과 데이터 중에서 결과 한 개를 저장한 객체

  jsonResult: 필요한 부분만 저장하여 반환할 리스트 객체

  cnt: 현재 작업 중인 검색 결과의 번호

- 지역 변수

  post['title']: post 객체의 title 항목에 저장된 값

  post['description']: post 객체의 description 항목에 저장된 값

  post['originallink']: post 객체의 originallink 항목에 저장된 값

  post['link']: post 객체의 link 항목에 저장된 값

- 메서드

  datetime.datetime.strptime(): 문자열을 날짜 객체 형식으로 변환

  pDate.strftime(): 날짜 객체의 표시 형식을 지정

  jsonResult.append(): 리스트 객체인 jsonResult에 원소를 추가

01. 네이버 API를 이용한 크롤링

- 네이버 뉴스 크롤링 : 3. 함수 설계하기

```
01  def getPostData(post, jsonResult, cnt):
02      title = post['title']
03      description = post['description']
04      org_link = post['originallink']
05      link = post['link']
06
07      pDate = datetime.datetime.strptime(post['pubDate'], '%a, %d %b %Y %H:%M:%S +0900')
08      pDate = pDate.strftime('%Y-%m-%d %H:%M:%S')
09
10  jsonResult.append({'cnt':cnt, 'title':title, 'description': description, 'org_link':org_link,
11                          'link': org_link, 'pDate':pDate})
11      return
```

02~05행 : 검색 결과가 들어 있는 post 객체에서 필요한 데이터 항목을 추출하여 변수에 저장

07행 : 네이버에서 제공하는 시간인 pubDate는 문자열 형태이므로 날짜 객체로 변환
pubDate는 그리니치 평균시 형식을 사용하는데 한국 표준시보다 9시간 느리므로 +0900 을 사용해 한국 표준시로 맞춤

08행 : 수정된 날짜를 '연-월-일 시:분:초' 형식으로 나타냄

10행 : 2~5행에서 저장한 데이터를 딕셔너리 형태인 {'키':값}으로 구성하여 리스트 객체인 jsonResult에 추가

01. 네이버 API를 이용한 크롤링

- 네이버 뉴스 크롤링 : 4. 전체 프로그램 작성하기
  - 파이썬 셸 창에서 [File]-[New File]을 클릭해 새 파일 창을 열고 다음의 파이썬 프로그램을 작성

```
import os
import sys
import urllib.request
import datetime
import time
import json
client_id = '본인이 발급받은 네이버 Client ID'
client_secret = '본인이 발급받은 네이버 Client Secret'

#[CODE 1]
def getRequestUrl(url):
    req = urllib.request.Request(url)
    req.add_header("X-Naver-Client-Id", client_id)
    req.add_header("X-Naver-Client-Secret", client_secret)

    try:
        response = urllib.request.urlopen(req)
        if response.getcode() == 200:
            print("[%s] Url Request Success" % datetime.datetime.now())
            return response.read().decode('utf-8')
     except Exception as e:
         print(e)
         print("[%s] Error for URL : %s" % (datetime.datetime.now(), url))
         return None
```

```
#[CODE 2]
def getNaverSearch(node, srcText, start, display):
    base = "https://openapi.naver.com/v1/search"
    node = "/%s.json" % node
    parameters = "?query = %s&start = %s&display = %s" %
                 (urllib.parse.quote(srcText), start, display)

    url = base + node + parameters
    responseDecode = getRequestUrl(url) #[CODE 1]

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

01. 네이버 API를 이용한 크롤링

- 네이버 뉴스 크롤링: 4. 전체 프로그램 작성하기

```
#[CODE 3]
def getPostData(post, jsonResult, cnt):
    title = post['title']
    description = post['description']
    org_link = post['originallink']
    link = post['link']

    pDate = datetime.datetime.strptime(post['pubDate'], '%a,
            %d %b %Y %H:%M:%S+0900')
    pDate = pDate.strftime('%Y-%m-%d %H:%M:%S')

    jsonResult.append({'cnt':cnt, 'title':title, 'description': description,
                       'org_link':org_link, 'link': org_link, 'pDate':pDate})
    return
```

네이버 뉴스 크롤링: 4. 전체 프로그램 작성하기

```
#[CODE 0]
def main():
    node = 'news' #크롤링할 대상
    srcText = input('검색어를 입력하세요: ')
    cnt = 0
    jsonResult = []
    jsonResponse = getNaverSearch(node, srcText, 1, 100) #[CODE 2]
    total = jsonResponse['total']

    while ((jsonResponse != None) and (jsonResponse['display'] != 0)):
        for post in jsonResponse['items']:
            cnt += 1
            getPostData(post, jsonResult, cnt) #[CODE 3]

        start = jsonResponse['start'] + jsonResponse['display']
        jsonResponse = getNaverSearch(node, srcText, start, 100) #[CODE 2]

    print('전체 검색 : %d 건' %total)

    with open('%s_naver_%s.json' % (srcText, node), 'w', encoding='utf8') as outfile:
        jsonFile = json.dumps(jsonResult, indent = 4, sort_keys = True, ensure_ascii = False)
        outfile.write(jsonFile)

    print("가져온 데이터 : %d 건" %(cnt))
    print('%s_naver_%s.json SAVED' % (srcText, node))
```

```
if __name__ == '__main__':
    main()
```

실습1(네이버 블로그에서 검색)

```
import urllib.request
import json

client_key = '부여받은 Client ID'
client_secret = '부여받은 Client Secret'

query = '치맥'
encText = urllib.parse.quote_plus(query)

num = 100
naver_url = 'https://openapi.naver.com/v1/search/blog.json?query=' + encText
+ '&display=' + str(num)

request = urllib.request.Request(naver_url)
request.add_header("X-Naver-Client-Id",client_key)
request.add_header("X-Naver-Client-Secret",client_secret)
response = urllib.request.urlopen(request)

rescode = response.getcode()
```

실습1-2

```
if(rescode == 200):
    response_body = response.read()
    dataList = json.loads(response_body)
    count = 1
    print('[' + query + '에 대한 네이버 블로그 글 ]')
    for data in dataList['items'] :
        print (str(count) + ' : ' + data['title'])
        print ('[' + data['description'] + ']')
        count += 1
else:
    print('오류 코드 : ' + rescode)
```

네이버 API(파파고-번역)

- Papago NMT API
  - 네이버 Papago에 적용된 번역 REST API, 입력된 텍스트를 다른 나라 언어(영어, 중국어)로 번역한 텍스트로 출력해주는 REST API
- Papago NMT API 예제 코드
  - https://developers.naver.com/docs/nmt/examples/
- Papago NMT API Reference
  - https://developers.naver.com/docs/nmt/reference/

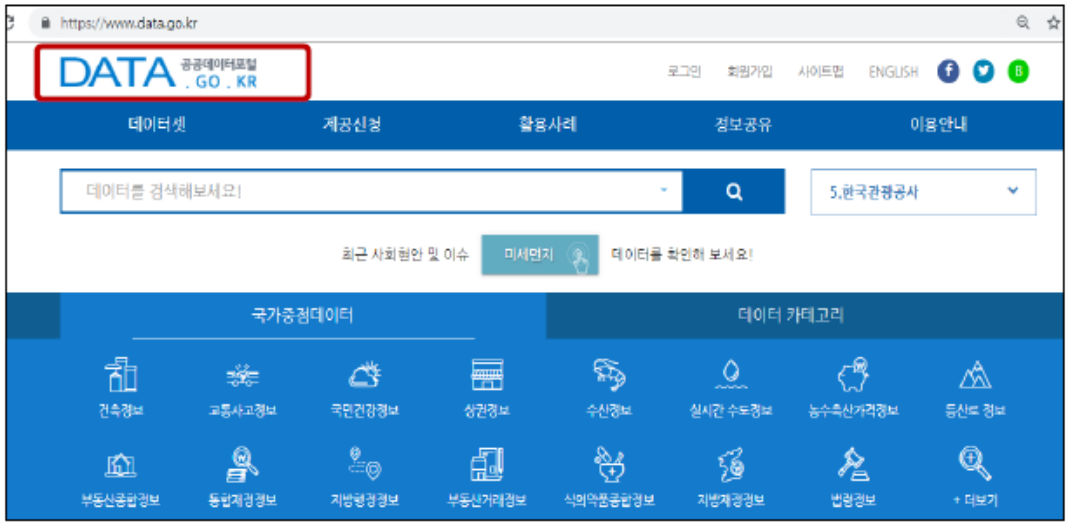
네이버 API(파파고-번역)

```
import os
import sys
import urllib.request
client_id = 'AdWH0R70Yez_uPyfJvYD'
client_secret = 'NQgA3Og0kE'

def papago_func(encText):
    encText = urllib.parse.quote(encText)
    data = "source=ko&target=en&text=" + encText
    url = "https://openapi.naver.com/v1/papago/n2mt"
    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, data=data.encode("utf-8"))
    rescode = response.getcode()
    if(rescode==200):
        response_body = response.read()
        print(response_body.decode('utf-8'))
    else:
        print("Error Code:" + rescode)

encText=input("번역할 문장을 입력하세요")
papago_func(encText)
```

3. Open API를 활용한 공공 DB 수집

- 서울시 버스 정보와 버스 위치 정보
  - 공공데이터포털사이트URL

http://www.data.go.kr/



3. Open API를 활용한 공공 DB 수집

- 서울시 버스 정보와 버스 위치 정보
  - **공공 데이터 포털 사이트에 로그인**

3. Open API를 활용한 공공 DB 수집

- 서울시 버스 노선 정보와 버스 정류소 위치 정보
  - **Open API 활용 가이드 제공 => 이 문서를 참조하여 구현**

3. Open API를 활용한 공공 DB 수집

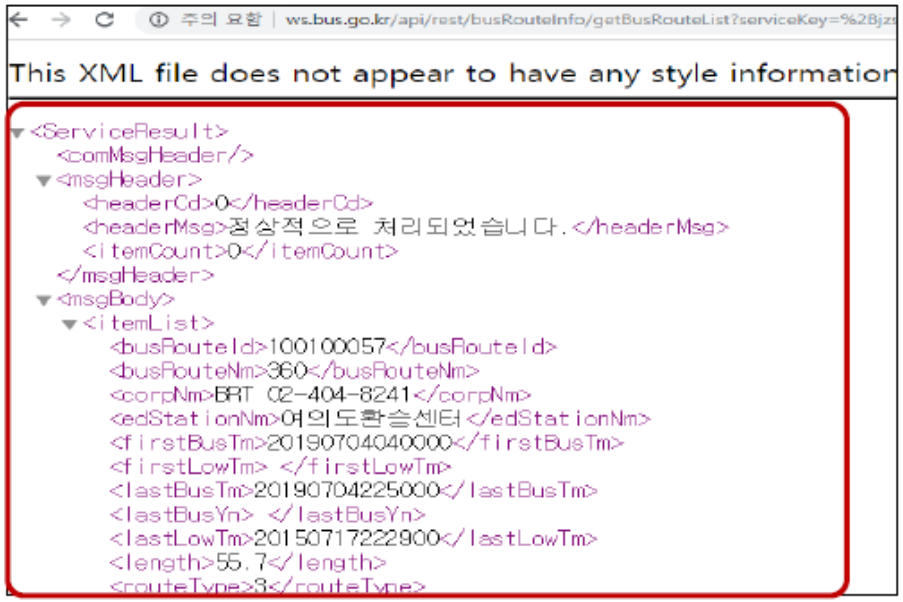
- 서울시 버스 정류소 위치 정보
  - **버스의 위치 정보를 받아 오기**

    - 버스의 노선 정보 요청
      ➡ 버스의 라우트아이디<busRouteId>를 알아야 함

    - 부여받은 인증키와 정보를 얻고자 하는 버스 번호를 입력하여 URL 구성

    http://ws.bus.go.kr/api/rest/busRouteInfo/getBusRouteList?serviceKey=인증키&strSrch=버스번호

    - 버스 번호에 대한 노선 정보 요청

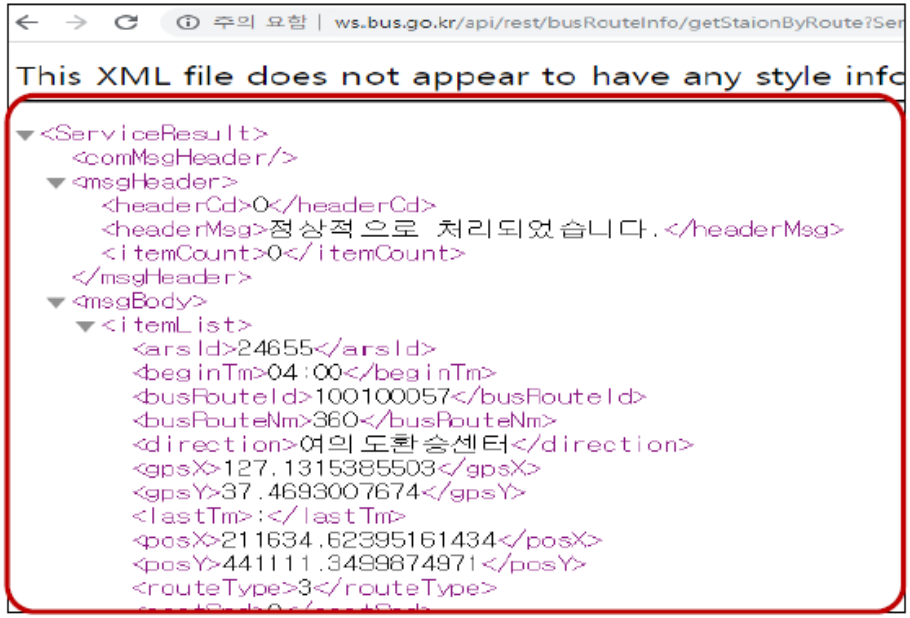


3. Open API를 활용한 공공 DB 수집

- 서울시 버스 정보와 버스 위치 정보
  - **버스 정류장 위치 정보를 받아 오기**

    - busRouteId에는 버스 노선 정보를 요청하고 받아온 XML 문서를 스크래핑하여 추출한 busRouteId를 사용

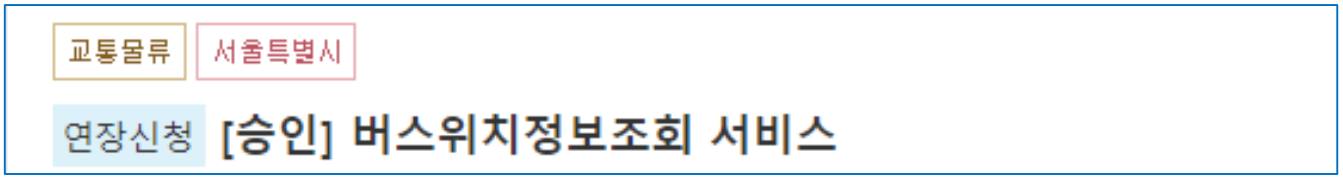
    http://ws.bus.go.kr/api/rest/busRouteInfo/getStaionByRoute?ServiceKey=인증키&busRouteId=100100057



. 3. Open API를 활용한 공공 DB 수집

▪ 서울시 버스 위치 정보

- 버스 노선 ID를 파라미터로 버스 위치 정보 받아오기



- busRoutId로 차량들의 위치정보를 조회한다.

http://ws.bus.go.kr/api/rest/buspos/getBusPosByRtid?ServiceKey=인증기&busRoutId=버스노선Id

3. Open API를 활용한 공공 DB 수집
- 버스 위치 정보 얻어 오기 실습

```
url2 ='http://ws.bus.go.kr/api/rest/buspos/getBusPosByRtid'
params = '?ServiceKey='+ key+'&busRouteId='+busRouteId

request = (url2 + params)
response_body = req.urlopen(request).read().decode('utf-8')
soup = BeautifulSoup(response_body, 'xml')
#print(response_body)
item_list=soup.find_all('itemList')
for item in item_list:
    gpsX=item.find('gpsX').text
    gpsY=itme.find('gpsY').text
    print(gpsX,gpsY)
```

출입국 관광통계 조회

- 출입국 관광통계 조회

공공행정 한국문화관광연구원

활용신청 [승인] 출입국관광통계서비스

- http://openapi.tour.go.kr/openapi/service/EdrcntTourismStatsService/getEdrcntTourismStatsList

요청변수(Request Parameter)

항목명	샘플데이터	설명
YM	201201	연월
NAT_CD	112	국가코드
ED_CD	D	출입국구분코드

실습-1(서울특별시 버스위치 정보 조회)

```
from bs4 import BeautifulSoup
import urllib.request as req

busNum = '360'
key = '인증키'
url1 = 'http://ws.bus.go.kr/api/rest/busRouteInfo/getBusRouteList?serviceKey='+key+'&strSrch='+busNum
savename = 'businfo.xml'
req.urlretrieve(url1, savename)

xml = open(savename, 'r', encoding='utf-8').read()
soup = BeautifulSoup(xml, 'xml')

busRouteId = None
for itemList in soup.find_all('itemList') :
    busRouteId = itemList.find('busRouteId').string
    busRouteNm = itemList.find('busRouteNm').string
    if busRouteNm == busNum :
        break

url2='http://ws.bus.go.kr/api/rest/buspos/getBusPosByRtid?ServiceKey='+key+'&busRouteId='+busRouteId
savename = 'buspos.xml'
req.urlretrieve(url2, savename)
```

실습1

```
xml = open(savename, 'r', encoding='utf-8').read()
soup = BeautifulSoup(xml, 'xml')

busPos = []
for itemList in soup.find_all('itemList') :
    gpsY = itemList.find('gpsY').string
    gpsX = itemList.find('gpsX').string

    busPos.append((gpsY, gpsX))

print('[ ' + busNum + '번 버스의 운행 위치 ]')
if len(busPos) != 0 :
    print(busNum + '번 버스 ' + str(len(busPos)) + '대 운행중...')
    for lat,lng in busPos :
        print(lat+','+lng)
else :
    print('현재 운행중인 ' + busNum + '번 버스가 없어요...')
```

실습-2(출입국관광통계서비스)

```
import urllib.request
import datetime
import time
import json
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import font_manager, rc

def get_request_url(url):
    req = urllib.request.Request(url)

    try:
        response = urllib.request.urlopen(req)
        if response.getcode() == 200:
            print ("[%s] Url Request Success" % datetime.datetime.now())
            return response.read().decode('utf-8')
    except Exception as e:
        print(e)
        print("[%s] Error for URL : %s" % (datetime.datetime.now(), url))
        return None
```

```
#[CODE 1]
def getNatVisitor(yyyymm, nat_cd, ed_cd):
    access_key=“인증키”
    end_point =
"http://openapi.tour.go.kr/openapi/service/EdrcntTourismStatsService/getEdrcntTo
urismStatsList"

    parameters = "?_type=json&serviceKey=" + access_key
    parameters += "&YM=" + yyyymm
    parameters += "&NAT_CD=" + nat_cd
    parameters += "&ED_CD=" + ed_cd

    url = end_point + parameters

    retData = get_request_url(url)

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

```
def main():
    jsonResult = []
    #중국: 112 / 일본: 130 / 미국: 275
    national_code = "275"
    ed_cd = "E"

    nStartYear = 2011
    nEndYear = 2017

    for year in range(nStartYear, nEndYear):
        for month in range(1, 13):
            yyyymm = "{0}{1:0>2}".format(str(year), str(month))
            jsonData = getNatVisitor(yyyymm, national_code, ed_cd)

            print (json.dumps(jsonData,indent=4, sort_keys=True,ensure_ascii=False))

            if (jsonData['response']['header']['resultMsg'] == 'OK'):
                krName = jsonData['response']['body']['items']['item']["natKorNm"]
                krName = krName.replace(' ', '')
                iTotalVisit = jsonData['response']['body']['items']['item']["num"]
                print('%s_%s : %s' %(krName, yyyymm, iTotalVisit))
                jsonResult.append({'nat_name': krName, 'nat_cd': national_code,
                                   'yyyymm': yyyymm, 'visit_cnt': iTotalVisit})
```

```
    cnVisit = []
    VisitYM = []
    index = []
    i = 0
    for item in jsonResult:
        index.append(i)
        cnVisit.append(item['visit_cnt'])
        VisitYM.append(item['yyyymm'])
        i = i + 1

    with open('%s(%s)_해외방문객정보_%d_%d.json'
              % (krName, national_code, nStartYear, nEndYear-1), 'w', encoding='utf8') as outfile:
        retJson = json.dumps(jsonResult,indent=4, sort_keys=True, ensure_ascii=False)
        outfile.write(retJson)


if __name__=='__main__':
    main()
```