9장

날씨 정보를 이용한 맛집 추천 프로젝트

학습순서

9.1 학습목표

점심 메뉴를 날씨에 따라 추천해주는 프로그램이 있다면 얼마나 좋을까요? 이번 프로젝트는 날씨 정보와 미세먼지 정보를 이용한 음식 추천 프로젝트입니다.

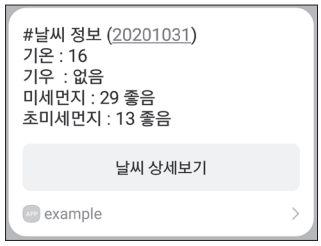
구현 순서

'날씨 정보를 이용한 맛집 추천 프로젝트'를 위한 구현 순서는 아래와 같습니다. 공공데이터 포털 OpenAPI를 통해서 날씨 데이터와 미세먼지 데이터를 가져옵니다. 날씨와 미세먼지 상황별 추천 음식을 모아 네이버 지역 검색으로 맛집을 검색합니다. 그리고 그 결과를 카카오톡 메시지로 전송합니다.

[그림 9-1] 날씨 정보를 이용한 맛집 추천 프로젝트 구현 순서

전체 흐름을 보면서 느꼈겠지만 '8장. 스마트 일정 관리 프로젝트'와 매우 유사합니다. 사용하는 데이터만 변경되었을 뿐입니다. '8장. 스마트 일정 관리 프로젝트'는 구글 캘린더 데이터를 사용했고, 이 장에서는 날씨와 미세먼지 데이터를 사용합니다. 이렇게 사용하는 데이터만 바꾸었을 뿐인데, 새로운 프로젝트로 느껴지죠? 여러분도 작은 변화를 주어 다양하고, 근사한 프로젝트를 만들 수 있답니다.

최종 결과



[그림 9-2] 날씨 및 미세먼지 정보가 담긴 카카오톡 메시지

[그림 9-3] 날씨 및 미세먼지 정보에 맞는 맛집 정보

언어 & 환경(IDE)

파이썬 3.7 & 주피터 노트북

프로젝트 리소스 및 소스코드

- food_recommender.ipynb: 구현된 소스코드
- kakao_utils.py: 카카오톡 메시지 전송을 위한 유틸리티 구현 코드(참고: 2장. 나에게 카카오톡 메시지 보내기 프로젝트)
- res\kakao_message\kakao_token.json: 카카오 access token과 refresh token이 관리되는 파일 (참고: 2장. 나에게 카카오톡 메시지 보내기 프로젝트)

※ 프로젝트에 필요한 리소스 및 소스코드의 경로 구성은 자유롭게 설정하면 됩니다. 그러나, 아래 설명되어 있는 코드 구현이 제시된 구성으로 되어 있으니, 동일한 경로와 파일명으로 구성해 둔다면 별다른 코드 수정 없이 실행해 볼 수 있습니다.

9.2 사전 준비하기

날씨 정보를 이용한 맛집 추천 프로젝트를 위해서 네 가지 사전 준비가 필요합니다.

- 9.2.1 2장에서 배운 '나에게 카카오톡 메시지 보내기'를 사전에 준비하기
- 9.2.2 공공 데이터 포털 – 날씨 서비스 인증키 발급받기
- 9.2.3 공공 데이터 포털 – 미세먼지 서비스 인증키 발급받기
- 9.2.4 카카오 애플리케이션에 사이트 도메인 등록하기

9.2.1 나에게 카카오톡 메시지 보내기

이 부분에 대한 내용은 2장에서 배웠던 내용을 참고하면 됩니다.

※ 9.2.2와 9.2.3에서 배우는 공공 데이터 포털의 OpenAPI는 인증키 발급 이후 한두 시간 후에 사용 가능합니다. 만약, 주말이라면 하루 이틀 정도 소요된다는 점을 고려해주세요.

9.2.2 공공 데이터 포털 – 날씨 서비스 인증키 발급받기

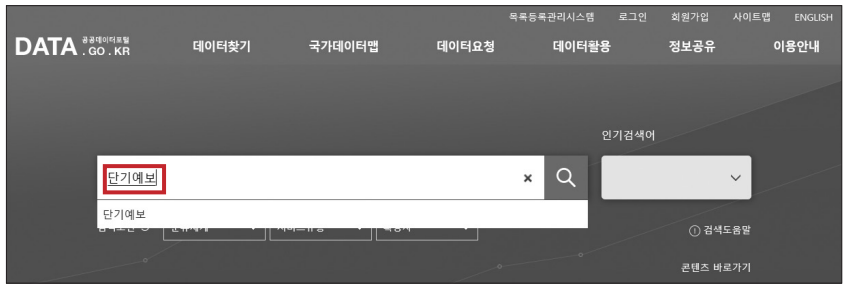
국가 공공데이터 포털에 접속해서 계정이 있다면 [로그인]을 하고, 계정이 없다면 [회원가입]을 합니다.

〉 접속 URL: www.data.go.kr

[그림 9–4] 공공데이터 포털 회원가입 화면

회원가입은 어렵지 않으니, 순서대로 진행한 후 로그인을 합니다. 이 장에서는 회원가입 설명을 별도로 작성하지 않았습니다.

검색 창에 '단기예보'를 입력한 후 검색합니다.



[그림 9–5] 단기예보 검색

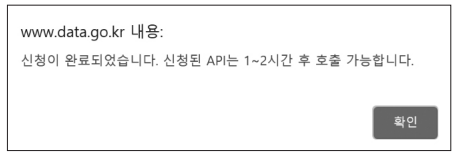
검색 결과 하단으로 내려오면 '오픈 API'가 있습니다. '단기예보 조회서비스' 항목의 [활용신청] 버튼을 눌러줍니다.

[그림 9–6] 오픈 API 항목 중 단기예보 조회서비스 활용 신청

'OpenAPI 개발계정 신청' 상세 페이지에서 '단기예보 조회'를 선택하고 라이선스에 '동의합니다'를 체크한 후 [활용신청] 버튼을 클릭합니다.

[그림 9-7] 단기예보 조회서비스 활용신청 상세 페이지

활용신청 후 아래와 같은 팝업창이 뜹니다. 신청된 후 1~2시간 후 호출이 가능합니다. 구현할 때 꼭 참고하세요!



[그림 9-8] 활용신청 완료 후 제약사항

마이페이지에서 활용 건수 및 '단기예보 조회서비스'가 추가되었는지 확인합니다. '단기예보 조회서비스'를 클릭하여 인증키를 확인합니다. 이 인증키를 잘 보관해주세요.

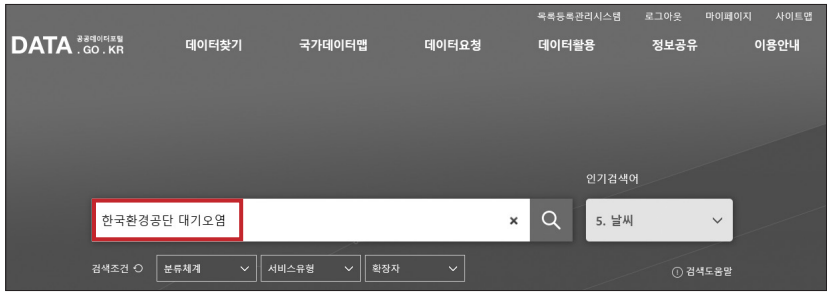
[그림 9-9] 마이페이지 → 단기예보 조회서비스 클릭

[그림 9-10] 단기예보 조회서비스 인증키 확인

다음은 미세먼지 서비스를 신청하겠습니다.

9.2.3 공공 데이터 포털 – 미세먼지 서비스 인증키 발급받기

인증키를 신청하는 방법은 날씨 서비스 인증키 발급과 매우 유사합니다. 검색 창에 '한국환경공단 대기오염'으로 검색합니다.

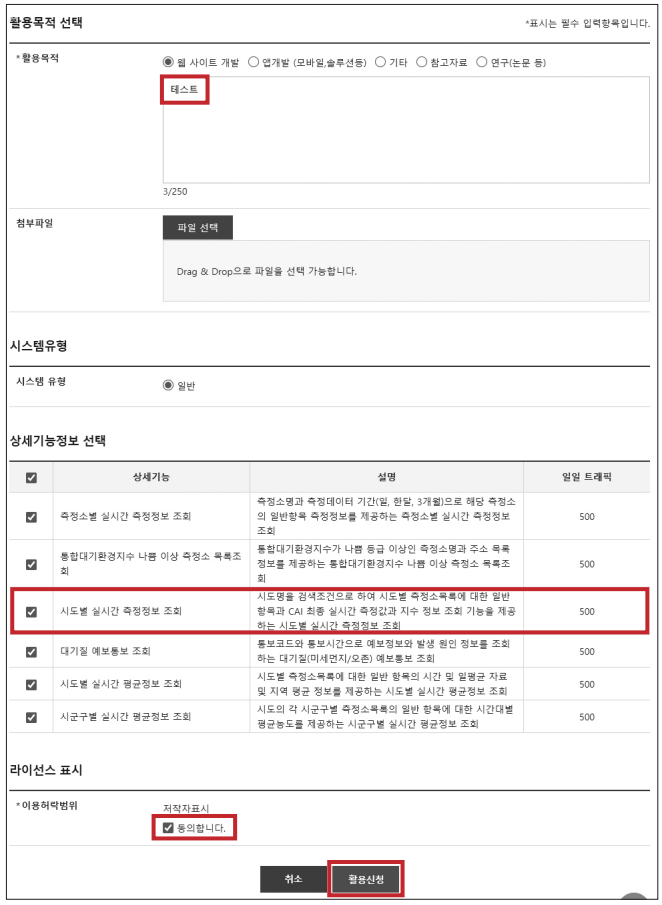


[그림 9–11] 한국환경공단 대기오염으로 검색

검색 결과 하단으로 내려오면 [오픈 API]가 있습니다. '한국환경공단_에어코리아_대기오염정보' 항목의 [활용신청]을 눌러줍니다.

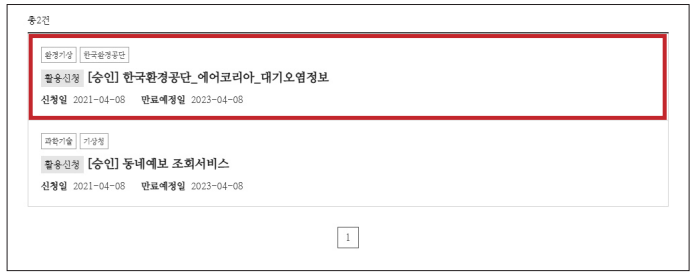
[그림 9–12] 오픈 API 항목 중 한국환경공단_에어코리아_대기오염정보 활용 신청

개발 계정 신청 상세페이지에서 '시도별 실시간 평균정보 조회'를 선택하고 라이선스 동의와 [활용신청] 버튼을 클릭합니다.



[그림 9-13] 한국환경공단 대기오염 서비스 활용 신청 상세 페이지

신청 API는 1~2시간 후 호출이 가능하다는 팝업이 뜹니다. 마이페이지로 들어가 인증키를 확인해봅니다.



[그림 9-14] 마이페이지 → 한국환경공단_에어코리아_대기오염정보 클릭

[그림 9–15] 한국환경공단_에어코리아_대기오염정보 인증키 확인

이 인증키를 잘 보관해주세요!

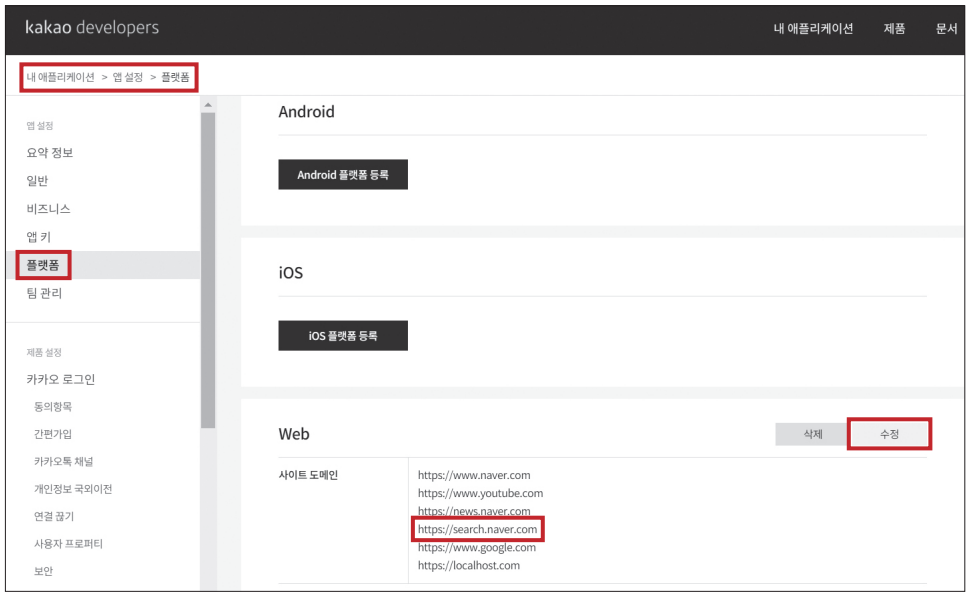
9.2.4 카카오 애플리케이션에 사이트 도메인 등록하기

카카오톡 메시지를 만들 때, 버튼을 만들 예정입니다. [날씨 상세보기] 버튼이 눌려지면, 네이버 '날씨' 검색으로 이동하고 싶습니다.

[그림 9–16] 네이버 날씨 검색

이를 위해 https://search.naver.com을 카카오 애플리케이션에 등록합니다. 만약 등록하지 않는다면, 버튼을 눌러도 해당 링크로 이동하지 않습니다.

순서: 카카오 개발자 사이트 접속 → '내 애플리케이션' 클릭 → 앱 설정 → 플랫폼 → Web에서 [수정] 버튼 클릭→ https://search.naver.com 추가



[그림 9-17] 카카오 애플리케이션에 사이트 도메인 등록

9.3 사전 지식 쌓기

날씨 정보를 이용한 맛집 추천 프로젝트를 시작하기에 앞서 학습하면 좋은 내용입니다.

- 9.3.1 날씨 정보 가져오기
- 9.3.2 미세먼지 정보 가져오기

9.3.1 날씨 정보 가져오기

날씨 정보로는 기온과 기상 상태 정보를 가져오겠습니다. 인증키를 확인했던 페이지로 이동합니다.

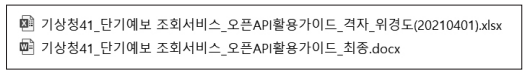
순서: 마이페이지 → 단기예보 조회서비스 클릭

[상세설명] 버튼을 클릭한 후, '참고문서'에 있는 '기상청41_단기예보 조회서비스_오픈API활용가이드_최종.zip' 파일을 다운받습니다.

[그림 9-18] 상세설명 클릭

[그림 9-19] 기상청41_단기예보 조회서비스_오픈API활용가이드_최종.zip 다운로드

zip 파일의 압축을 풀었습니다. 두 개의 파일이 있네요. 첫 번째 파일은 동네예보 조회서비스를 사용하기 위한 가이드 문서이며, 두 번째 파일은 파라미터 설정을 위한 위경도 좌표가 있는 파일입니다.



[그림 9-20] 압축을 푼 결과 내용

먼저 '기상청41_단기예보 조회서비스_오픈API활용가이드_최종.docx' 파일을 열어 봅시다. 문서에서 '[단기예보조회] 상세기능명세'를 찾아서 이동합니다. 대략 10페이지로 이동해보니 있네요. 이곳에 요청 URL 주소와 파라미터 정보가 적혀 있습니다.

3) [단기예보조회] 상세기능명세

a) 상세기능정보

상세기능 번호	3	상세기능 유형	조회 (상세)
상세기능명(국문)	단기예보조회		
상세기능 설명	단기예보 정보를 조회하기 위해 발표일자, 발표시각, 예보지점 X 좌표, 예보지점 Y 좌표의 조회 조건으로 발표일자, 발표시각, 자료구분문자, 예보값, 예보일자, 예보시각, 예보지점 X 좌표, 예보지점 Y 좌표의 정보를 조회하는 기능		
Call Back URL	http://apis.data.go.kr/1360000/VilageFcstInfoService_2.0/getVilageFcst		
최대 메시지 사이즈	[48,452] byte		
평균 응답 시간	[600] ms	초당 최대 트랙잭션	[30] tps

[그림 9-21] 날씨 정보 요청을 위한 URL 주소

b) 요청 메시지 명세

항목명(영문)	항목명(국문)	항목크기	항목구분	샘플데이터	항목설명
serviceKey	인증키	100	1	인증키 (URL Encode)	공공데이터포털에서 발급받은 인증키
numOfRows	한 페이지 결과 수	4	1	50	한 페이지 결과 수 Default: 10
pageNo	페이지 번호	4	1	1	페이지 번호 Default: 1
dataType	응답자료형식	4	0	XML	요청자료형식(XML/JSON) Default: XML
base_date	발표일자	8	1	20210628	'21년 6월 28일발표
base_time	발표시각	4	1	0500	05시 발표 *** 하단 참고자료 참조**
nx	예보지점 X 좌표	2	1	55	예보지점의 X 좌표값 ***별첨 엑셀 자료 참조**
ny	예보지점 Y 좌표	2	1	127	예보지점의 Y 좌표값 ***별첨 엑셀 자료 참조**

※ 항목구분 : 필수(1), 옵션(0), 1 건 이상 복수건(1..n), 0 건 또는 복수건(0..n)

[그림 9-22] 날씨 정보 요청을 위한 파라미터

파라미터 중 nx, ny의 경우는 엑셀 자료를 참고하라고 합니다. '기상청41_단기예보 조회서비스_오픈API활용가이드_격자_위경도(20210401).xlsx'을 열어 확인해봅니다. 해당 파일은 지역별 위경도를 나타낸 표입니다. 만약 문래동의 위경도를 알고 싶다면, 해당 주소를 검색하여 '격자 X'와 '격자 Y' 값을 사용하면 됩니다.

	A	B	C	D	E	F	G
1	구분	행정구역코드	1단계	2단계	3단계	격자 X	격자 Y
311	kor	1156055000	서울특별시	영등포구	당산제1동	58	126
312	kor	1156056000	서울특별시	영등포구	당산제2동	59	126
313	kor	1156058500	서울특별시	영등포구	도림동	59	126
314	kor	1156060500	서울특별시	영등포구	문래동	59	126
315	kor	1156061000	서울특별시	영등포구	양평제1동	58	126
316	kor	1156062000	서울특별시	영등포구	양평제2동	58	126
317	kor	1156063000	서울특별시	영등포구	신길제1동	59	125

[그림 9-23] 위경도 확인

예제는 문래동 주소로 X:59, Y:126로 입력하겠습니다.

코드

```
01 import requests
02 import json
03 import datetime
04
05 vilage_weather_url="http://apis.data.go.kr/1360000/ilageFcstInfoService_2.0
                        /getVilageFcst?"
06
07 service_key = "<단기예보조회의 인증키를 입력합니다.>"
08 base_date = datetime.datetime.today().strftime("%Y%m%d")  # "20200214" == 기준 날짜
09 base_time = "0800"  # 기준시간 값
10 nx = "59"
11 ny = "126"
12
13 payload = "serviceKey=" + service_key + "&" +\
14     "dataType=json" + "&" +\
15     "base_date=" + base_date + "&" +\
16     "base_time=" + base_time + "&" +\
17     "nx=" + nx + "&" +\
18     "ny=" + ny
19
20 # 값 요청
21 res = requests.get(vilage_weather_url + payload)
22 try:
23     items = res.json().get('response').get('body').get('items')
24     print(items)
25 except:
26     print("날씨 정보 요청 실패 : ", res.text)
```

실행 결과

```
{'item': [{'baseDate': '20201031',
   'baseTime': '0800',
   'category': 'POP',
   'fcstDate': '20201031',
   'fcstTime': '1200',
   'fcstValue': '20',
   'nx': 59,
   'ny': 126},
  {'baseDate': '20201031',
```

```
    'baseTime': '0800',
    'category': 'PTY',
    'fcstDate': '20201031',
    'fcstTime': '1200',
    'fcstValue': '0',
    'nx': 59,
    'ny': 126},

   ... (생략) ...

   {'baseDate': '20201031',
    'baseTime': '0800',
    'category': 'TMP',
    'fcstDate': '20201031',
    'fcstTime': '1200',
    'fcstValue': '16',
    'nx': 59,
    'ny': 126,

    ... (생략) ...
 ]}
```

결과로 출력된 category 값이 어떤 의미인지 확인할 수 있습니다. '기상청41_단기예보 조회서비스_오픈API활용가이드_최종.docx' 파일에 설명되어 있습니다. 이번 프로젝트에서는 두 개의 값을 사용합니다. 사용할 값은 강수형태와 기온을 의미하는 PTY와 TMP입니다.

예보구분	항목값	항목명	단위	압축bit수
단기예보	POP	강수확률	%	8
	PTY	강수형태	코드값	4
	PCP	1시간 강수량	범주 (1 mm)	8
	REH	습도	%	8
	SNO	1시간 신적설	범주(1 cm)	8
	SKY	하늘상태	코드값	4
	TMP	1시간 기온	℃	10
	TMN	일 최저기온	℃	10
	TMX	일 최고기온	℃	10
	UUU	풍속(동서성분)	m/s	12
	VVV	풍속(남북성분)	m/s	12
	WAV	파고	M	8
	VEC	풍향	deg	10
	WSD	풍속	m/s	10

[그림 9-24] 날씨 정보의 카테고리 의미

코드 설명

01~03행: 필요한 라이브러리를 import합니다.

05행: 날씨 정보를 얻기 위한 request url입니다.

07행: 발급받은 서비스 키입니다.

08행: 기준 날짜를 설정합니다. 오늘 날짜를 `datetime.datetime.today()`로 구하고, 날짜를 YYYYmmdd 형태로 만들기 위해 strftime("%Y%m%d")를 사용했습니다.

09행: 기준 시간을 설정합니다. 0800은 오전 8시를 의미합니다.

10~11행: '[그림 9-23] 위경도 확인'에서처럼, 문래동 위경도 좌표입니다.

13~18행: '[그림 9-22] 날씨 정보 요청을 위한 파라미터'와 같이 설정해야 할 값을 설정했습니다.

21행: 날씨 정보 요청을 합니다.

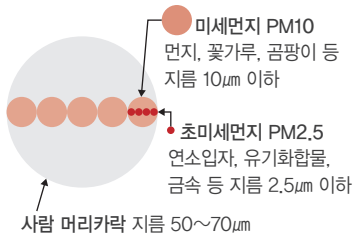
22~26행: 요청 성공 시 날씨 정보를 추출합니다. `res.json()`만 실행해보면, 아래와 같은 결과가 나옵니다.

```
{'response': {'header': {'resultCode': '00', 'resultMsg': 'NORMAL_SERVICE'},
  'body': {'dataType': 'JSON',
   'items': {'item': [{'baseDate': '20201031',
      'baseTime': '0800',
      'category': 'POP',
      'fcstDate': '20201031',
      'fcstTime': '1200',
      'fcstValue': '20',
      'nx': 59,
      'ny': 126},
```

날씨에 해당하는 정보들이 있는 값을 추출하기 위해 `res.json().get('response').get('body').get('items')`로 구현했습니다.

9.3.2 미세먼지 정보 가져오기

이번에는 미세먼지 정보를 가져오겠습니다. OpenAPI를 사용하기 전에 미세먼지 용어부터 확인해봅니다.



[그림 9-25] 미세먼지와 초미세먼지 비교[1]

1 출처: https://www.yna.co.kr/view/AKR20180320087100004

미세먼지를 크게 두 가지 종류로 나누고 있습니다. 미세먼지와 초미세먼지입니다. 사람의 머리카락과 비교한 그림인데요, 미세먼지는 PM10이라는 수치를 사용하고 초미세먼지는 PM2.5를 의미합니다. 이번 프로젝트에서는 두 치수를 모두 사용하겠습니다.

미세먼지의 PM10과 초미세먼지의 PM2.5를 요청해봅시다. 인증키를 확인했던 페이지로 이동합니다.

> **순서:** 마이페이지 → 한국환경공단_에어코리아_대기오염정보 클릭 → 상세설명 클릭

'참고문서'의 '에어코리아_대기오염정보 조회 서비스_기술문서_v1.0.docx 파일'을 다운받습니다. '데이터 포맷'은 json과 xml을 모두 지원하네요.

JSON+XML **한국환경공단_에어코리아_대기오염정보**

각 측정소별 대기오염정보를 조회하기 위한 서비스로 기간별, 시도별 대기오염 정보와 통합대기환경지수 나쁨 이상 측정소 내역, 대기질(미세먼지/오존) 예보 통보 내역 등을 조회할 수 있다.

1 2 관심

활용신청

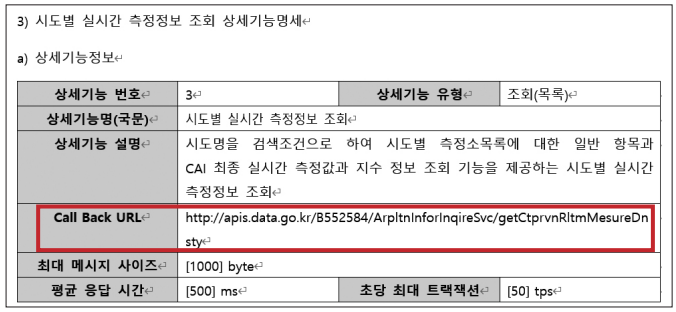
오류신고 및 담당자 문의

OpenAPI 정보

분류체계	환경 - 대기	제공기관	한국환경공단
관리부서명	대기환경처	관리부서 전화번호	032-590-3507
API 유형	REST	데이터포맷	JSON+XML
활용신청	497	키워드	한국환경공단,에어코리아,미세먼지
등록	2020-12-06	수정	2021-01-08
심의유형	개발단계 : 허용 / 운영단계 : 미허용		
비용부과유무	무료		
이용허락범위	OPEN 공공저작물_출처표시 + 변경금지		
참고문서	에어코리아_대기오염정보 조회 서비스_기술문서_v1.0.docx		

[그림 9-26] 한국환경공단_에어코리아_대기오염정보의 참고 문서 다운받기

다운받은 문서의 13페이지로 이동해보니 요청 URL 주소와 파라미터 정보가 적혀 있습니다.



3) 시도별 실시간 측정정보 조회 상세기능명세

a) 상세기능정보

상세기능 번호	3	**상세기능 유형**	조회(목록)
상세기능명(국문)	시도별 실시간 측정정보 조회		
상세기능 설명	시도명을 검색조건으로 하여 시도별 측정소목록에 대한 일반 항목과 CAI 최종 실시간 측정값과 지수 정보 조회 기능을 제공하는 시도별 실시간 측정정보 조회		
Call Back URL	http://apis.data.go.kr/B552584/ArpltnInforInqireSvc/getCtprvnRltmMesureDnsty		
최대 메시지 사이즈	[1000] byte		
평균 응답 시간	[500] ms	**초당 최대 트랜잭션**	[50] tps

[그림 9-27] 미세먼지 정보 요청을 위한 URL 주소

b) 요청 메시지 명세

항목명(영문)	항목명(국문)	항목크기	항목구분	샘플데이터	항목설명
serviceKey	서비스키	-	1	인증키(URL Encode)	서비스키
returnType	데이터표출방식	4	0	xml	ml 혹은 json
numOfRows	한 페이지 결과 수	4	0	100	한 페이지 결과 수
pageNo	페이지 번호	4	0	1	페이지 번호
sidoName	시도 명	10	1	서울	시도 이름 (전국, 서울, 부산, 대구, 인천, 광주, 대전, 울산, 경기, 강원, 충북, 충남, 전북, 전남, 경북, 경남, 제주, 세종)
ver	오퍼레이션 버전	4	0	1.0	버전별 상세 결과 아래쪽 참고

[그림 9-28] 미세먼지 정보 요청을 위한 파라미터

참고로 ver 파라미터에 대한 정보는 이렇게 설명되어 있네요.

※ 버전(ver) 항목설명

- 버전을 포함하지 않고 호출할 경우 : PM2.5 데이터가 포함되지 않은 원래 오퍼레이션 결과 표출
- 버전 1.0을 호출할 경우 : PM2.5 데이터가 포함된 결과 표출
- 버전 1.1을 호출할 경우 : PM10, PM2.5의 24시간 예측 이동 평균데이터가 포함된 결과 표출
- 버전 1.2를 호출할 경우 : 측정망 정보 데이터가 포함된 결과 표출
- 버전 1.3을 호출할 경우 : PM10, PM2.5의 1시간 등급 자료가 포함된 결과 표출

버전 1.0을 호출하여 PM10과 PM2.5의 정보를 모두 받아보겠습니다. 이런 정보들을 보니, 이제 코드를 구현할 수 있겠죠?

코드

```
01 import requests
02 import json
03 import datetime
04
05 dust_url = "http://apis.data.go.kr/B552584/ArpltnInforInqireSvc/
               getCtprvnRltmMesureDnsty?"
06
07 service_key = "<한국환경공단_에어코리아_대기오염정보의 인증키를 입력합니다.>"
08
09 payload = "serviceKey=" + service_key + "&" +\
10     "returnType=json" + "&" +\
11     "sidoName=서울" + "&" +\
12     "ver=1.0"
13
```

```
14  # pm10과 pm2.5 수치 가져오기
15  res = requests.get(dust_url + payload)
16  result = res.json()
17  dust = dict()
18  if (res.status_code == 200) & (result['response']['header']['resultCode']
                                   == '00'):
19      dust['PM10'] = {'value' : int(result['response']['body']['items']
                                      [0]['pm10Value'])}
20      dust['PM2.5'] = {'value' : int(result['response']['body']['items']
                                       [0]['pm25Value'])}
21  else:
22      print("미세먼지 가져오기 실패 : ", result['response']['header']['resultMsg'])
23
24  print(dust)
```

실행 결과

```
{'PM10': {'value': 29}, 'PM2.5': {'value': 13}}
```

코드 설명

01~3행: 필요한 라이브러리를 import합니다.

05행: 문서에 설명되어 있는 요청 url입니다.

09~12행: 요청 파라미터를 설정합니다. 발급받은 서비스 키는 serviceKey 변수에 설정합니다. returnType=json으로 설정하여 응답 데이터 포맷을 json으로 선택합니다. sidoName=서울로, 서울의 정보를 가져옵니다. 여러분이 살고 있는 지역으로 설정하면 더 재밌겠죠? pm2.5와 pm10 정보를 모두 받기 위해서 ver=1.0으로 설정했습니다.

15행: 미세먼지 정보를 얻기 위해 요청했습니다.

19~20행: 성공적으로 수치를 가지고 온 경우에 dust 값을 채워 넣습니다. result['response']['body']['items'][0]은 응답으로 온 결과에 제일 첫 번째 정보를 사용하기 위해서입니다. pm10과 2.5의 값이 string형으로 전달되므로, `int()`로 자료형을 변환했습니다.

9.4 구현하기

날씨 정보를 이용한 맛집 추천 프로젝트 구현 순서는 아래와 같습니다.

공공데이터 포털의 OpenAPI를 통해서 날씨 데이터와 미세먼지 데이터를 가지고 옵니다. 날씨와 미세먼지 상황에 맞춘 추천 음식을 모아 네이버 지역 검색으로 맛집을 검색합니다. 그리고 그 결과를 카카오톡 메시지로 전송합니다.

[그림 9-29] 날씨 정보를 이용한 맛집 추천 프로젝트 구현 순서

구현 순서

작은 기능 단위로 코딩하면서 점진적으로 프로젝트를 완성해봅시다.

- Step 1: 날씨 정보 얻기 및 정제하기
- Step 2: 미세먼지 정보 얻기 및 정제하기
- Step 3: 날씨에 따른 음식 데이터 구하기
- Step 4: 네이버 맛집 검색하기
- Step 5: 카카오톡 메시지를 보내기 위한 사전 준비하기
- Step 6: 텍스트 템플릿으로 날씨 및 미세먼지 정보 전송하기
- Step 7: 리스트 템플릿으로 맛집 정보 전송하기

STEP 1 날씨 정보 얻기 및 정제하기

공공데이터 포털의 OpenAPI를 사용하여 날씨 정보를 요청하고 필요한 정보만 저장합니다.

코드

```
01 import requests
02 import json
03 import datetime
04
05 vilage_weather_url="http://apis.data.go.kr/1360000/VilageFcstInfoService_2.0/
                        getVilageFcst?"
06
07 service_key =  "<단기예보조회의 인증키를 입력합니다.>"
08 base_date = datetime.datetime.today().strftime("%Y%m%d") # "20200214" == 기준 날짜
09 base_time = "0800"  # 기준 시간
10 nx = "59"
11 ny = "126"  # 문래동 위경도 좌표
12
13 payload = "serviceKey=" + service_key + "&" +\
14     "dataType=json" + "&" +\
15     "base_date=" + base_date + "&" +\
16     "base_time=" + base_time + "&" +\
17     "nx=" + nx + "&" +\
18     "ny=" + ny
19
20 pty_code = { "0": "없음", "1" : "비", "2" : "비/눈", "3":"눈", "4":"소나기",
              "5":"빗방울", "6":"빗방울/눈날림", "7":"눈날림"}
21
22 data = dict()
23 data['date'] = base_date
24 weather = dict()
25
26 # 값 요청
27 res = requests.get(vilage_weather_url + payload)
28 try:
29     items = res.json().get('response').get('body').get('items')
30     for item in items['item']:
31         # 기온
32         if item['category'] == 'TMP':
33             weather['tmp'] = item['fcstValue']
34
35         # 강수상태
36         if item['category'] == 'PTY':
37             weather['code'] = item['fcstValue']
38             weather['state'] = pty_code[item['fcstValue']]
```

```
except:
    print("날씨 정보 가져오기 실패 : ", res.text)

data['weather'] = weather
print(data['weather'])
```

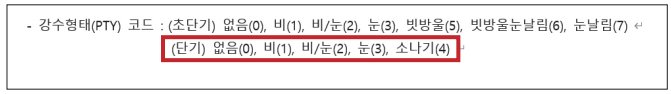
실행 결과

```
{'code': '0', 'state': '없음', 'tmp': '16'}
```

코드 설명

01~18행: '9.3.1 날씨 정보 가져오기'에서 설명한 내용입니다.

20행: 강수형태(PTY)일 경우, 각 값이 가지는 의미입니다. '기상청41_단기예보 조회서비스_오픈API활용가이드_최종.docx' 파일을 열어보면 다음과 같은 내용이 있습니다.


- 강수형태(PTY) 코드 : (초단기) 없음(0), 비(1), 비/눈(2), 눈(3), 빗방울(5), 빗방울눈날림(6), 눈날림(7)
(단기) 없음(0), 비(1), 비/눈(2), 눈(3), 소나기(4)

[그림 9-30] 강수형태 코드

22행: 카카오톡 메시지에 날씨와 미세먼지 데이터를 전송할 예정입니다. 해당 정보를 data 변수에 정제하여 저장합니다.

23행: 'date'를 key로, 오늘 날짜를 value로 저장합니다.

24행: 날씨 정보를 관리하기 위해서 weather dictionary 변수를 생성합니다.

27행: 날씨 정보를 얻기 위해 요청합니다.

29~38행: 날씨 정보 중에서 온도에 해당하는 TMP 값과 강수 상태에 해당하는 PTY 값만 사용합니다. 해당 요소들을 추출합니다.

STEP 2 미세먼지 정보 얻기 및 정제하기

이번에는 미세먼지 정보를 요청하고, 정제해보겠습니다.

코드

```
def get_pm10_state(pm10_value):
    if pm10_value < 30:
        pm10_state = "좋음"
    elif pm10_value < 80:
        pm10_state = "보통"
    elif pm10_value < 150:
        pm10_state = "나쁨"
```

```
    else:
        pm10_state = "매우 나쁨"

    return pm10_state

def get_pm25_state(pm25_value):
    if pm25_value < 15:
        pm25_state = "좋음"
    elif pm25_value < 35:
        pm25_state = "보통"
    elif pm25_value < 75:
        pm25_state = "나쁨"
    else:
        pm25_state = "매우 나쁨"

    return pm25_state

# 미세먼지 데이터 요청
dust_url = "http://apis.data.go.kr/B552584/ArpltnInforInqireSvc/
            getCtprvnRltmMesureDnsty?"

service_key = "<한국환경공단_에어코리아_대기오염정보의 인증키를 입력합니다.>"

payload = "serviceKey=" + service_key + "&" +\
    "returnType=json" + "&" +\
    "sidoName=서울" + "&" +\
    "ver=1.0"

# pm10 pm2.5 수치 가져오기
res = requests.get(dust_url + payload)
result = res.json()
dust = dict()
if (res.status_code == 200) & (result['response']['header']['resultCode'] == '00'):
    dust['PM10'] = {'value' : int(result['response']['body']['items']
                                  [0]['pm10Value'])}
    dust['PM2.5'] = {'value' : int(result['response']['body']['items']
                                   [0]['pm25Value'])}
    # PM10 미세먼지 30 80 150
    pm10_value = dust.get('PM10').get('value')
    pm10_state = get_pm10_state(pm10_value)

```

```
46      # PM2.5 초미세먼지 15 35 75
47      pm25_value = dust.get('PM2.5').get('value')
48      pm25_state = get_pm25_state(pm25_value)
49
50      dust.get('PM10')['state'] = pm10_state
51      dust.get('PM2.5')['state'] = pm25_state
52  else:
53      print("미세먼지 가져오기 실패 : ", result['response']['header']['resultMsg'])
54
55  data['dust'] = dust
56  print(data['dust'])
```

실행 결과

```
{'PM10': {'value': 29}, 'PM2.5': {'value': 13}}
{'PM10': {'value': 29, 'state': '좋음'}, 'PM2.5': {'value': 13, 'state': '좋음'}}
```

코드 설명

미세먼지와 초미세먼지의 기준표입니다. 각 값들은 기준 범위가 다릅니다. 미세먼지는 80~150 범위가 나쁨이지만, 초미세먼지는 35~75가 나쁨이네요. 각 범위별 좋음/보통/나쁨/매우나쁨 단계를 표현하는 코드 구현이 필요합니다.

[그림 9-31] 미세먼지 기준표

[그림 9-32] 초미세먼지 기준표

01~11행: 미세먼지의 단계를 표현하는 함수입니다. pm10 값을 입력으로 받고, 단계에 맞춰 상태값을 전달하는 함수를 정의합니다.

13~23행: 초미세먼지의 단계를 표현하는 함수입니다. pm2.5 값을 입력으로 받고, 단계에 맞춰 상태값을 전달하는 함수를 정의합니다.

26~38행: '9.3.2 미세먼지 정보 가져오기'를 참고하세요.

40~51행: 미세먼지 및 초미세먼지의 수치(value)와 상태값(state)을 구하고, dust 변수에 각 정보를 추가합니다.

55행: 카카오톡 메시지를 보낼 때 참고할 data 변수에 미세먼지 정보를 설정합니다.

STEP 3 날씨에 따른 음식 데이터 구하기

'날씨에 따른 음식 데이터를 어떻게 구해야 할까?' 고민을 많이 했는데요. 구글에 검색하니 정보가 나오더라고요. 구글에 비 오는 날, 미세먼지 많은 날을 검색하여 추천 음식들을 모아봤습니다. 추후 여러분이 원하는 음식으로 변경해보세요.

[그림 9-33] 비 오는날 먹으면 좋은 음식 검색

- 비/눈 오는 날: 부대찌개, 아구찜, 해물탕, 칼국수, 수제비, 짬뽕, 우동, 치킨, 맥주, 국밥, 김치부침개, 두부김치, 파전
- 미세먼지: 콩나물국밥, 고등어, 굴, 쌀국수, 마라탕

코드

```
import random

rain_foods = "부대찌개,아구찜,해물탕,칼국수,수제비,짬뽕,우동,치킨,국밥,김치부침개,두부김치,
              파전".split(',')
pmhigh_foods = "콩나물국밥,고등어,굴,쌀국수,마라탕".split(',')

def get_foods_list(weather,dust_pm10,  dust_pm20):
    if weather != '0':
        recommand_state = 'Case1'
        # random.sample(x, k=len(x)) 무작위로 리스트 섞기
        foods_list = random.sample(rain_foods, k=len(rain_foods))
```

```
11      elif dust_pm10  == '매우나쁨' or dust_pm20 == '매우나쁨' :
12          recommand_state = 'Case2'
13          foods_list = random.sample(pmhigh_foods, k=len(pmhigh_foods))
14      else:
15          recommand_state = 'Case3'
16          foods_list = ['']
17
18      return recommand_state, foods_list
```

코드 설명

01행: 필요한 라이브러리를 import합니다.

03행: 비/눈 오는 날의 추천음식 목록입니다.

04행: 미세먼지가 많은 날의 추천음식 목록입니다.

06~18행: 날씨, 미세먼지, 초미세먼지를 입력으로 받아, 추천 상태와 추천 음식 목록을 전달하는 함수입니다. 추천 상태는 임의로 정했습니다.

[표 9-1] 음식 추천 조건

기상 : 비/눈/소나기 여부	미세먼지 : 높음 이상 여부	추천 결과
O	O	Case 1: 비 오는 날 음식 세 개 추천
O	X	
X	O	Case 2: 미세먼지 음식 세 개 추천
X	X	Case 3: 리뷰 순 맛집 음식 세 개 추천

- Case1. 비나 눈이 온다면, rain_foods에 저장된 음식을 추천합니다.
- Case2. 맑지만 미세먼지 수치가 높은 날에는 pmhigh_foods에 저장된 음식을 추천합니다.
- Case3. 맑고 미세먼지도 없다면, 리뷰 순 맛집을 추천합니다.

10행에서 `random.sample()`를 호출하였는데, 만약 random.sample(rain_foods, k=3)에 사용한다면 rain_foods에서 무작위로 세 개만 추출하라는 의미입니다. 코드에서는 random.sample(rain_foods, k=len(rain_foods))으로 전체 길이만큼을 추출하는 것이므로, 순서만 바뀐다고 생각하면 됩니다.

STEP 4 네이버 맛집 검색하기

이제 네이버 지역 검색 OpenAPI를 사용하여 '문래동'의 맛집을 찾아봅니다.

코드

```
01  def naver_local_search(query, display):
02      # 네이버 애플리케이션의 client_id와 client_secret 키 설정
03      headers = {
04          "X-Naver-Client-Id" :  "<네이버 애플리케이션의 Client ID를 입력하세요>",
05          "X-Naver-Client-Secret" : "<네이버 애플리케이션의 Client Secret를 입력하세요>"
```

```
    }

    # 지역 검색 요청 파라미터 설정
    params = {
                "sort"  : "comment",
                "query" : query,
                "display" : display
             }

    # 지역 검색 URL과 요청 파라미터
    naver_local_url = "https://openapi.naver.com/v1/search/local.json"

    # 지역 검색 요청
    res = requests.get(naver_local_url, headers=headers, params=params)

    # 지역 검색 결과 확인
    places = res.json().get('items')

    return places

# 경우 1 : 비/눈/소나기              => 비오는날 음식 세 개 추천
# 경우 2 : 초/미세먼지 나쁨 이상      => 미세먼지에 좋은 음식 세 개 추천
# 경우 3 : 정상                     => 블로그 리뷰 순 맛집 추천
weather = data.get('weather').get('code')
dust_pm10 = data.get('dust').get('PM10').get("state")
dust_pm20 = data.get('dust').get('PM2.5').get("state")

# 날씨 상태와 음식 종류 선정
weather_state, foods_list = get_foods_list(weather, dust_pm10, dust_pm20)

# 위치는 사용자가 사용할 지역으로 변경 가능
location = "문래동"

# 추천된 맛집을 담을 리스트
recommands = []
for food in foods_list:
    # 지역 검색 요청 파라미터 설정
    # 만약, 날씨가 맑은 경우, food가 ''이므로, '문래동 맛집'이 된다.
    query= location + " " + food + " 맛집"

    # 맛집 검색 결과
    result_list = naver_local_search(query, 3)
```

```
48
49      if len(result_list) > 0:
50          if weather_state == 'Case3':
            # Case3 처리 로직 :  맛집 검색 결과에서 가장 상위 세 개를 가져옴
51              recommands = result_list
52              break
53          else:  # Case1, Case2 처리 로직 : 해당 음식 검색 결과에서 가장 상위를 가져옴
54              recommands.append(result_list[0])
55      else:
56          print("검색 결과 없음")  # 메뉴에 해당하는 맛집이 없을 수 있음
57
58      if len(recommands) == 3:
59              break
60
61  print(recommands)
```

실행 결과

```
[{'title': '올드문래',
  'link': 'http://oldmullae.itrocks.kr/',
  'category': '술집>맥주,호프',
  'description': '',
  'telephone': '',
  'address': '서울특별시 영등포구 문래동2가 14-28',
  'roadAddress': '서울특별시 영등포구 도림로 433-6',
  'mapx': '302448',
  'mapy': '546330'},
 {'title': '러스트 베이커리',
  'link': 'https://rustbakery.modoo.at',
  'category': '카페,디저트>베이커리',
  'description': '',
  'telephone': '',
  'address': '서울특별시 영등포구 문래동2가 42-19',
  'roadAddress': '서울특별시 영등포구 경인로79길 15',
  'mapx': '302397',
  'mapy': '546248'},
 {'title': '호텔707',
  'link': 'http://www.instagram.com/hotel707',
  'category': '술집>바(BAR)',
  'description': '',
  'telephone': '',
  'address': '서울특별시 영등포구 문래동2가 60-4 호텔707',
  'roadAddress': '서울특별시 영등포구 도림로139길 2-2',
  'mapx': '302369',
  'mapy': '546414'}]
```

코드 설명

01~24행: 각 행의 의미는 '8장. 스마트 일정 관리 프로젝트'의 'Step 3) 네이버 지역 검색으로 맛집 검색하기'를 참고하세요. 동일한 코드지만, 함수로 만들었다는 차이점이 있습니다. 함수로 만들면, 재사용이 용이하고 가독성이 높아집니다.

29~34행: 날씨(weather), 미세먼지(dust_pm10), 초미세먼지(dust_pm20)의 상태를 입력으로 주고 `get_food_list()`를 호출합니다. 이를 통해 날씨 상태와 추천하는 음식 종류를 얻어옵니다.

47행: 네이버 지역 검색으로 맛집을 검색합니다.

49~59행: 맛집 검색 결과에서 recommands 변수를 채워 줄 때, Case 3인 경우는 검색 결과를 대입한 후 for문을 빠져나옵니다. 그 이유는 Case 3의 경우는 날씨나 미세먼지 상태가 아닌 리뷰 순으로 세 개를 받아오도록 되어 있기 때문입니다. Case 3이 아닌 경우는 54행처럼 한 개의 값만 가지고 옵니다. 만약 foods_list=['수제비', '짬뽕', '우동' ...(생략)...]이라면, '수제비' 맛집 한 개, '짬뽕' 맛집 한 개, '우동' 맛집 한 개로 설정됩니다. 58행에서는 추천 음식이 총 세 개가 되었다면, 반복문을 종료합니다.

STEP 5 카카오톡 메시지를 보내기 위한 사전 준비하기

코드

```
01 import json
02 import kakao_utils
03
04 KAKAO_TOKEN_FILENAME = "res/kakao_message/kakao_token.json"
05 KAKAO_APP_KEY = "<REST_API 앱키를 입력하세요>"
06 kakao_utils.update_tokens(KAKAO_APP_KEY, KAKAO_TOKEN_FILENAME)
```

코드 설명

'2장. 나에게 카카오톡 메시지 보내기 프로젝트'의 '보충자료'를 참고합니다.

STEP 6 텍스트 템플릿으로 날씨 및 미세먼지 정보 전송하기

텍스트 템플릿으로 날씨와 미세 먼저 정보를 전송하겠습니다.

코드

```
01 # 날씨 상세 정보 URL
02 weather_url = "https://search.naver.com/search.naver
                 ?sm=top_hty&fbm=0&ie=utf8&query=%EB%82%A0%EC%94%A8"
03
04 # 날씨 정보 만들기
```

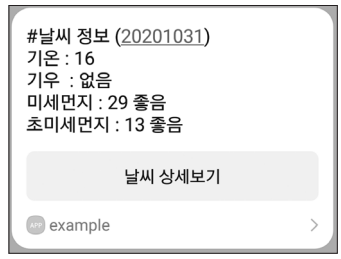
```
text = f"""\
# 날씨 정보 ({data['date']})
기온 : {data['weather']['tmp']}
기우  : {data['weather']['state']}
미세먼지 : {data['dust']['PM10']['value']} {data['dust']['PM10']['state']}
초미세먼지 : {data['dust']['PM2.5']['value']} {data['dust']['PM2.5']['state']}
"""

# 텍스트 템플릿 형식 만들기
template = {
  "object_type": "text",
  "text": text,
  "link": {
    "web_url": weather_url,
    "mobile_web_url": weather_url
  },
  "button_title": "날씨 상세보기"
}

# 카카오톡 메시지 전송
res = kakao_utils.send_message(KAKAO_TOKEN_FILENAME, template)
if res.json().get('result_code') == 0:
    print('날씨 및 미세먼지 정보 성공적으로 보냈습니다.')
else:
    print('날씨 및 미세먼지 정보 성공적으로 보내지 못했습니다. 오류메시지 : ', res.json())
```

문래동의 날씨와 미세먼지 정보를 담아 카카오톡으로 전송된 메시지입니다.

실행 결과



[그림 9-34] 날씨 및 미세먼지 정보가 담긴 카카오톡 메시지

코드 설명

05~11행: 텍스트 메시지로 보내질 정보를 적습니다. 날짜, 기온, 기우, 미세먼지, 초미세먼지에 대한 정보입니다.

14~22행: 텍스트 템플릿을 구성합니다.

25~29행: 카카오톡 메시지를 전송합니다.

STEP 7 리스트 템플릿으로 맛집 정보 전송하기

카카오톡의 리스트 템플릿을 이용하여 맛집 정보를 전송하겠습니다.

코드

```
# 리스트 템플릿 형식 만들기
contents = []
template = {
    "object_type" : "list",
    "header_title" : "현재 날씨에 따른 음식 추천",
    "header_link" : {
        "web_url": weather_url,
        "mobile_web_url" : weather_url
    },
    "contents" : contents,
    "buttons" : [
        {
            "title" : "날씨 정보 상세보기",
            "link" : {
                "web_url": weather_url,
                "mobile_web_url" : weather_url
            }
        }
    ],
}

# contents 만들기
for place in recommands:
    title = place.get('title')  # 장소 이름
    # title : 태극쿵푸<b>마라탕</b>
    # html 태그 제거
    title = title.replace('<b>','').replace('</b>','')

    category = place.get('category')  # 장소 카테고리
```

```
    telephone = place.get('telephone')  # 장소 전화번호
    address = place.get('address')  # 장소 지번 주소

    # 각 장소를 클릭할 때 네이버 검색으로 연결해주기 위해 작성된 코드
    enc_address = address + ' ' + title
    query = "query=" + enc_address

    # 장소 카테고리가 카페이면 카페 이미지
    # 이외에는 음식 이미지
    if '카페' in category:
        image_url = "https://freesvg.org/img/pitr_Coffee_cup_icon.png"
    else:
        image_url = "https://freesvg.org/img/bentolunch.png?w=150&h=150&fit=fill"

    # 전화번호가 있다면 제목과 함께 넣어줍니다.
    if telephone:
        title = title + "\ntel) " + telephone

    # 카카오톡 리스트 템플릿 형식에 맞춰줍니다.
    content = {
        "title": "[" + category + "] " + title,
        "description": ' '.join(address.split()[1:]),
        "image_url": image_url,
        "image_width": 50, "image_height": 50,
        "link": {
            "web_url": "https://search.naver.com/search.naver?" + query,
            "mobile_web_url": "https://search.naver.com/search.naver?" + query
        }
    }

    contents.append(content)


# 카카오톡 메시지 전송
res = kakao_utils.send_message(KAKAO_TOKEN_FILENAME, template)
if res.json().get('result_code') == 0:
    print('맛집 정보 성공적으로 보냈습니다.')
else:
    print('맛집 정보 성공적으로 보내지 못했습니다. 오류메시지 : ', res.json())
```

날씨와 미세먼지를 고려한 문래동의 맛집 정보입니다.

실행 결과

[그림 9-35] 날씨 및 미세먼지에 맞는 맛집 정보가 담긴 카카오톡 메시지

코드 설명

'8장. 스마트 일정 관리 프로젝트'의 'Step 5) 카카오톡 메시지 전송하기'를 참고하세요. 동일한 템플릿에 정보만 다르다고 생각해도 무관합니다.

사는 동네로 위경도 좌표를 바꿔보거나, 미세먼지 상태도 지금과 다르게 변경해보세요. 우리 동네의 날씨 데이터와 미세먼지를 확인하는 것만으로 재미난 프로젝트가 될 것입니다.

9.5 요약과 정리하기

이 장에서는 동네의 온도, 미세먼지 데이터를 고려하여 맛집을 추천해주는 프로젝트를 진행했습니다. 온도와 미세먼지 데이터는 공공데이터 포털(www.data.go.kr)에서 제공하는 OpenAPI를 사용했습니다. 그 후 이어지는 네이버 지역 검색 OpenAPI로 맛집 찾기와 카카오톡 메시지 보내기는 '8장. 스마트 일정 관리 프로젝트'와 유사합니다. 구현을 하면서 느꼈겠지만, 데이터만 변경했을 뿐인데, 서로 다른 콘셉트의 프로젝트가 탄생했습니다. 여러분은 어떤 데이터로 변경해서 다른 프로젝트를 탄생시키고 싶은가요?