1. 데이터 수집과 크롤링 개념 소개

1.1 데이터 수집의 개념

1.2 웹 크롤링의 개념

1.3 데이터 수집과 웹 크롤링의 윤리적, 법적 쟁점

데이터 수집의 개념

◆ 데이터 수집이란?

- 데이터 수집은 정보를 수집하고 기록하는 과정

◆ 데이터의 중요성

- 의사결정 지원
  - 데이터는 정확한 정보를 기반으로 한 의사 결정을 가능하게 함
  - 예측분석, 마케팅 전략 수립, 재무 계획 등 다양한 분야에서 중요한 역할을 지원
    - 예) 기업의 예측 분석을 통해 다음 분기의 매출을 예상하고, 이에 따른 생산 계획을 수립

- 경쟁력 확보
  - 데이터 분석을 통해 시장 동향을 파악하고 경쟁력을 유지할 수 있게 함
    - 예) 소셜 미디어 데이터를 분석하여 제품의 인지도를 높이고 타겟 마케팅을 통해 새로운 고객을 확보

데이터 수집의 개념

◆ 활용 예시

- 비즈니스
  - 고객 행동 분석을 통한 개인화된 마케팅 전략 수립
    - 예) 고객의 구매 이력, 웹사이트 방문 기록, 소셜 미디어 활동 데이터를 분석하여 각 고객에게 맞춤형 광고 메시지 제공
- 연구
  - 실험 결과 데이터를 분석하여 연구 성과를 평가
    - 예) 의약품 개발 과정에서 실험 데이터를 통해 효능과 부작용을 평가하고, 이를 토대로 향후 연구 방향을 설정
- 정부
  - 공공 데이터를 분석하여 정책 결정에 반영
    - 예) 인구 통계 데이터를 기반으로 한 지역 별 교육 인프라 투자 계획 수립하여 교육의 질을 향상

웹 크롤링의 개념

◆ 웹 크롤링이란?

- 인터넷 상에 존재하는 다양한 웹 페이지를 자동으로 탐색하여 정보를 수집하는 과정
  - 웹 페이지의 데이터를 분석하거나 데이터베이스를 구축하기 위해 사용됨

◆ 크롤러와 스크래퍼의 차이점

- 크롤러(Crawler)
  - 웹 크롤링을 실행하는 프로그램 또는 스크립트를 의미
  - 웹 페이지를 자동으로 방문하고 링크를 따라가며 데이터를 수집함
    - 예) 네이버나 구글 검색 엔진의 크롤러는 웹 페이지를 수집하고, 이를 인덱싱하여 사용자에게 검색결과를 제공
- 스크래퍼(Scraper)
  - 특정 웹 페이지에서 필요한 데이터를 추출하는 도구를 의미
  - HTML 구조를 분석하고 원하는 데이터를 추출하여 저장하거나 분석에 활용
    - 예) 가격 비교 사이트에서는 각 상품 페이지에서 가격과 설명을 추출하여 사용자에게 제공

웹 크롤링의 개념

◆ 웹 크롤링의 작동 과정

- 시작 URL에서 부터 시작하여 HTML 문서를 다운로드
- 링크를 추출하고, 다음으로 이동할 URL을 결정하여 저장구조(큐)에 저장
- 각 URL을 방문하고 해당 웹 페이지의 내용을 추출
- 추출된 데이터는 필터링이나 전처리 후 저장하거나 분석에 활용됨

◆ 적용 사례

- 뉴스 집계 사이트
  - 다양한 뉴스 사이트에서 최신 기사를 수집하여 사용자에게 제공함
- 부동산 정보 수집
  - 부동산 웹 사이트에서 매매 및 임대 정보를 수집하여 시세 추이를 분석하고 비교
- SNS 데이터 분석
  - 소셜 미디어에서 특정 키워드의 사용량 및 트렌드를 분석하여 마케팅 전략에 활용

데이터 수집과 크롤링의 윤리적, 법적 쟁점

◆ 데이터 수집 시 고려해야 할 윤리적 문제

- 개인정보 보호
  - 개인 식별 정보를 포함한 데이터를 수집 시 개인정보 보호법에 따른 규제를 준수해야 함
    - 예) 사용자 동의 없이 개인 식별 가능 정보를 수집하는 것은 불법일 수 있음

- 데이터 사용 목적의 투명성
  - 수집한 데이터를 어떻게 사용하고, 누구와 공유하는지 명확히 고지해야 함
    - 예) 수집된 데이터가 마케팅 목적이나 연구에 사용될 경우, 목적을 명확히 하고 사용 동의를 받아야 함

- 데이터 정확성과 신뢰성
  - 수집한 데이터가 정확하고 신뢰할 수 있는지 확인해야 함
    - 예) 잘못된 정보를 공개할 경우 사용자에게 피해를 줄 수 있으므로 정확성 검토가 필요함

데이터 수집과 크롤링의 윤리적, 법적 쟁점

◆ 로봇 배제 표준(Robots.txt)에 대한 이해

- 웹 사이트 소유자가 웹 크롤러가 특정 페이지나 디렉토리에 방문하지 않도록 제어하는 표준
- 구조 및 작동 방식
  - Robots.txt 파일
    - 웹사이트 루트 디렉토리에 위치한 텍스트 파일로, 크롤러에게 허용되는 경로와 제한되는 경로를 명시함
  - 방문 권한 지정
    - 특정 경로에 대해 크롤러에게 허용 또는 거부 권한을 부여하여 웹 사이트 로딩 속도를 향상시키고 서버 부하를 줄임

◆ 고려사항

- 법적 규제 준수
  - 데이터 수집과 관련된 법적 요구사항과 규제를 준수
    - 예) 개인정보 보호법 시행으로 개인정보 보호에 관한 엄격한 규제
- 프로토콜 존중
  - 웹 사이트가 웹 크롤러에게 명시한 제한을 존중하고 준수해야 함
  - 웹 사이트 소유자의 의도와 서버 부하 관리를 위한 필수 조치

2. 실습 환경 구축

2.1 Visual Studio Code 설치

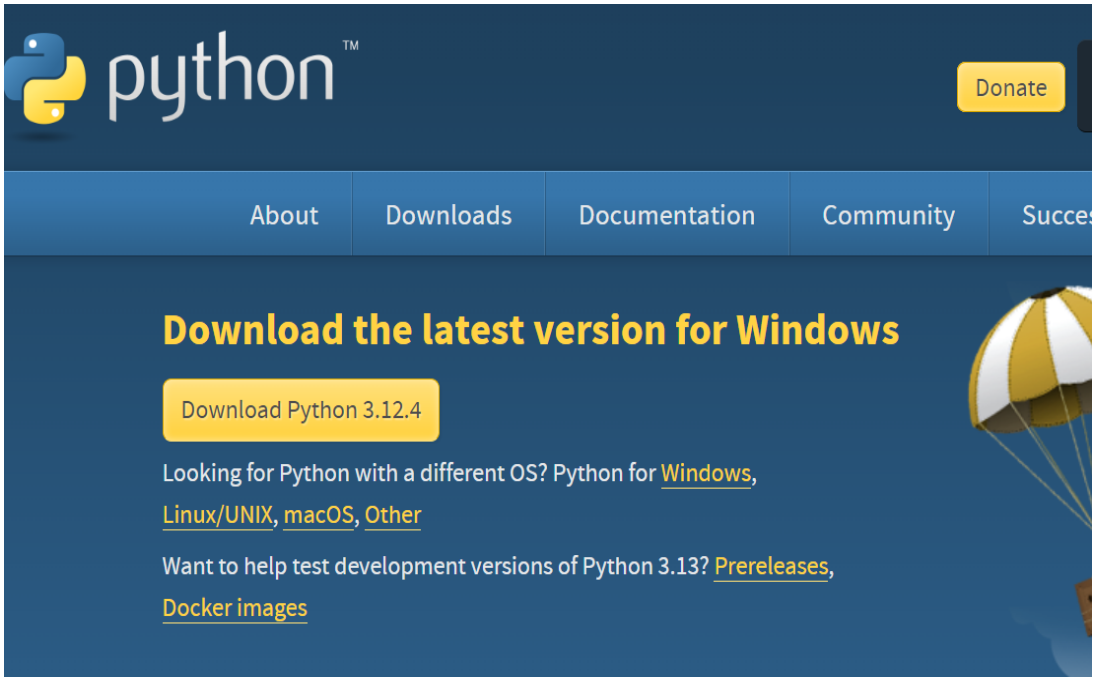
2.2 사용법 소개

Python 다운로드 및 설치

◆ Python 다운로드

- 공식 웹사이트에서 다운로드
  - https://www.python.org/downloads/
- 설치 파일 실행 후 설치 완료
- 콘솔창에서 파이썬 버전 확인

```
python --version
```



Visual Studio Code (VS Code) 설치

◆ VS Code란?

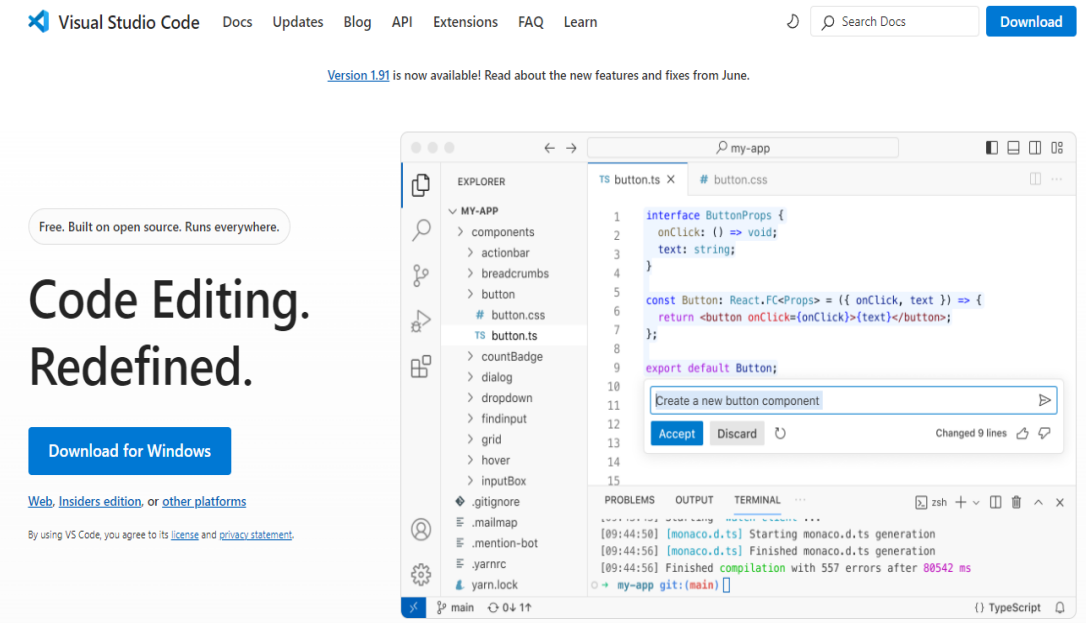
- 마이크로소프트에서 개발한 통합 개발 환경(IDE)을 제공하는 도구
  - 다양한 언어를 지원하며 빠르고 효율적인 코딩 환경을 제공
  - Window, macOS, Linux에서 모두 사용 가능
  - 수많은 확장 기능을 통해 기능 확장 가능

◆ 설치 방법

- VS Code 공식 웹사이트에서 다운로드

  https://code.visualstudio.com/

- 설치 파일 실행 후 설치 완료



VS Code 확장 기능

◆ 확장 기능 설치 

- Python extension for Visual Studio Code
- Jupyter
  - Jupyter Notebook 지원을 위한 기능
  - 기존 jupyter notebook 사용 방식 그대로 이용가능 함
- Korean Language Pack for Visual Studio Code (한국어 지원)

◆ 설정

- Auto save 설정

3. API를 이용한 데이터 수집

3.1 API 기본 개념 소개

3.2 JSON 데이터 처리

3.3 Requests 라이브러리 소개 및 설치

3.4 XML 데이터 처리

3.1 API 기본 개념 소개

API의 기본 개념 소개

◆ API란 무엇인가?

- API (Application Programming Interface)의 약자로, 소프트웨어 애플리케이션 간에 서로 통신할 수 있도록 만든 인터페이스를 의미함
  - 다른 애플리케이션이나 서비스에서 제공하는 기능을 외부에서 활용할 수 있게 함

◆ 특징

- 명세화
  - 특정한 규칙과 명세에 따라 정의되어야 함
- 재사용성
  - 기존의 기능을 다른 애플리케이션에서 재사용할 수 있도록 하는 기능
- 추상화
  - 내부 구현을 숨기고 필요한 기능에 집중할 수 있도록 도와 주는 기능

Open API의 기본 개념 소개

◆ Open API란?

- Open API는 누구나 접근하고 사용할 수 있도록 공개된 API를 의미함
- 주로 기업이나 공공기관에서 데이터를 외부에 제공하기 위해 사용함

◆ 특징

- 공개 접근
  - 누구나 접근할 수 있으며, 별도의 인증 절차를 거쳐 사용할 수 있음
- 표준화된 규격
  - REST, SOAP 등의 표준화된 규격을 따름
- 문서화
  - API 사용 방법과 예제 코드가 포함된 문서가 제공되어야 함

Open API의 기본 개념 소개

◆ Open API의 활용

- 공공 데이터
  - 정부 및 공공기관에서 제공하는 데이터(API)를 활용하여 새로운 서비스 개발
    - 예) 기상청 API를 이용한 날씨 정보 제공 앱
- 소셜 미디어
  - 트위터, 페이스북 등의 소셜 미디어 플랫폼에서 제공하는 API를 통해 사용자 데이터 수집 및 분석
- 금융 서비스
  - 금융기관에서 제공하는 API를 통해 실시간 금융 데이터 제공
    - 예) 주식 시세 API를 이용한 주식 거래 앱

◆ Open API 사용시 유의사항

- 대부분의 Open API는 인증을 필요로 함
- API 호출에는 제한이 있을 수 있으며, 준수하지 않으면 사용이 제한 될 수 있음
- API 제공자는 데이터 사용에 대한 정책을 명시하고 있으며, 이를 준수해야 함

공공 데이터 포털 API 이용하기

◆ 공공데이터 포털

- url : https://www.data.go.kr
- 회원가입 및 로그인

◆ 식품의약품 안전처_식품 영양성분 정보

- 활용신청 ➔ OpenAPI 개발계정 신청 ➔ 신청하면 승인

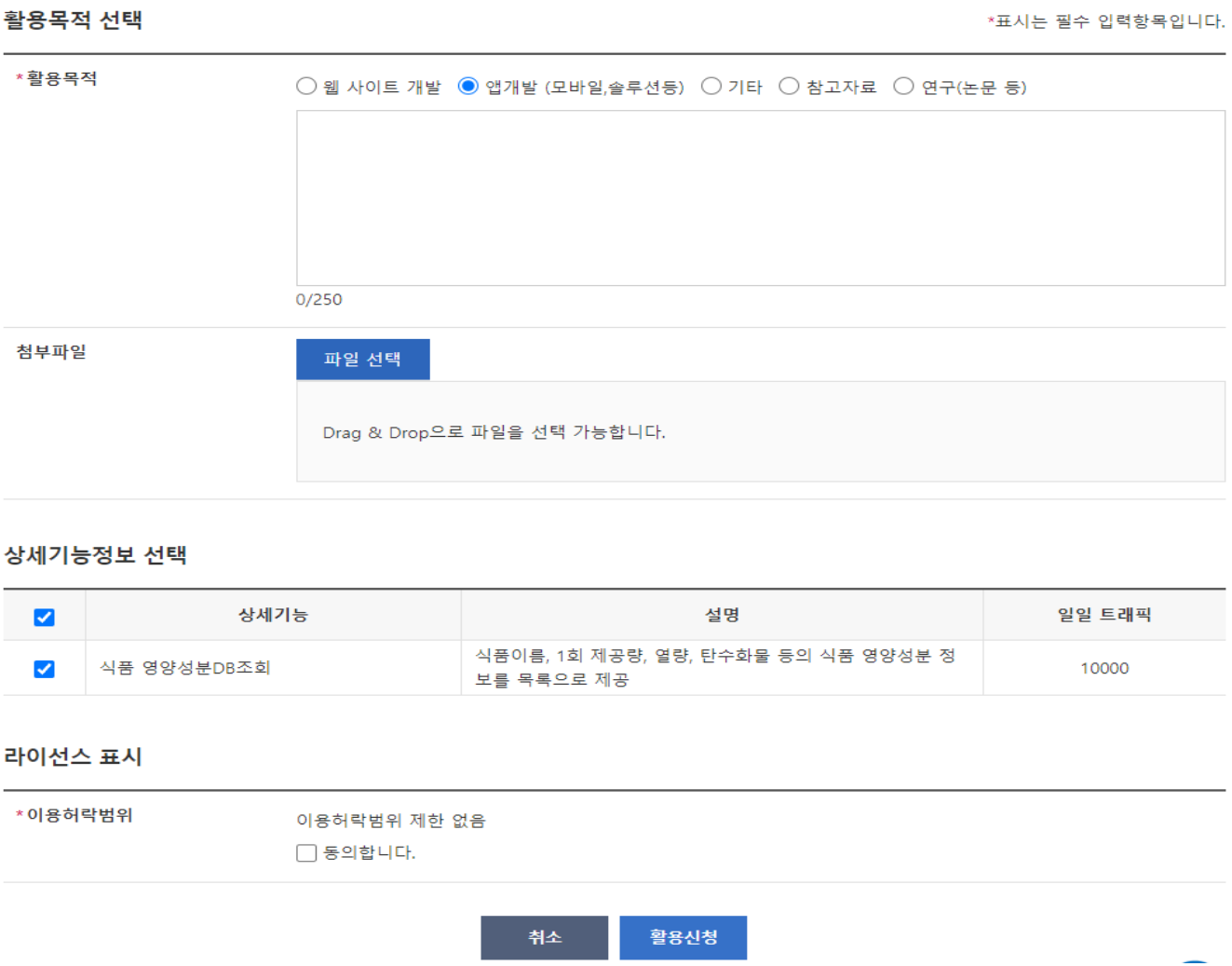
OpenAPI 개발계정 신청

JSON+XML 식품의약품안전처_식품 영양성분 정보

제공기관	식품의약품안전처	서비스유형	REST
심의여부	자동승인	신청유형	개발계정 | 활용신청
처리상태	신청	활용기간	승인일로부터 24개월 간 활용가능

공공 데이터 포털 API 이용하기

◆ 식품의약품 안전처_식품 영양성분 정보



3.2 JSON 데이터 처리

JSON 데이터

◆ JSON이란 무엇인가?

- JavaScript Object Notation 의 약자
- 데이터를 저장하고 전송하기 위해 사용하는 경량 데이터 교환 포맷
- 텍스트 기반으로 사람이 읽고 쓸 수 있으며, 기계가 분석하고 생성할 수 있음

◆ JSON의 특징

- 간결성
  - 가독성이 좋고, 구조가 간결하여 데이터를 쉽게 이해할 수 있음
- 가독성
  - 사람과 기계 모두가 읽고 쓰기 쉽게 설계되어 있음
- 유연성
  - 객체와 배열을 사용하여 복잡한 데이터 구조를 쉽게 표현할 수 있음
- 호환성
  - 대부분의 프로그래밍 언어에서 JSON을 쉽게 파싱하고 생성할 수 있는 라이브러리를 제공함

JSON의 기본 문법

◆ 키-값 쌍으로 데이터를 표현하며, 중첩된 구조를 가질 수 있음

◆ 주요 요소

- JSON 객체
  - 중괄호 '{}'로 감싸고, 키-값 쌍으로 구성

```
{
    "name": "John",
    "age": 30
}
```

- JSON 배열
  - 대괄호 '[]'로 감싸고, 쉼표로 구분된 값들의 리스트
  - 배열의 인덱스는 0부터 시작

```
[ "apple", "banana", "cherry" ]
```

JSON의 기본 문법

◆ JSON 데이터 타입

- 기본자료형
  - 객체(사전), 배열(리스트), 문자열, 숫자, Boolean, null
- Null은 빈 값을 나타냄
- 리스트 마지막에 쉼표가 있으면 안됨
- 객체의 키 값은 반드시 문자열

```
obj = """
{"name": "Wes",
 "places_lived": ["United States", "Spain", "Germany"],
 "pet": null,
 "siblings": [{"name": "Scott", "age": 25, "pet": "Zuko"},
              {"name": "Katie", "age": 33, "pet": "Cisco"}]
}
"""
```

JSON 데이터 읽고 쓰기

◆ 파이썬 표준 라이브러리 json

- JSON 데이터를 파이썬 객체로 변환하거나, 파이썬 객체를 JSON 데이터로 변환하는 기능 제공
- 주요 기능
  - json.loads() : JSON 문자열을 파이썬 사전 객체로 디코딩

```
import json
result = json.loads(obj)
result
```

```
{'name': 'Wes',
 'places_lived': ['United States', 'Spain', 'Germany'],
 'pet': None,
 'siblings': [{'name': 'Scott', 'age': 25, 'pet': 'Zuko'},
  {'name': 'Katie', 'age': 33, 'pet': 'Cisco'}]}
```

result['name']

result['siblings']

result['siblings'][0]

result['siblings'][0]['name']

  - json.load() : 파일에서 읽은 JSON 데이터를 파이썬 사전 객체로 디코딩

```
json_file_path = "sample.json"
with open(json_file_path, 'r', encoding='utf-8') as file:
    result = json.load(file)
result
```

JSON 데이터 읽고 쓰기

◆ 파이썬 표준 라이브러리 json

- 주요 기능
  - json.dumps() : 파이썬 사전 객체를 JSON 문자열로 인코딩

```
asjson = json.dumps(result)
asjson
```

```
'{"name": "Wes", "places_lived": ["United States", "Spain", "Germany"], "pet": null, "siblings":
[{"name": "Scott", "age": 25, "pet": "Zuko"}, {"name": "Katie", "age": 33, "pet": "Cisco"}]}'
```

  - json.dump() : 파이썬 사전 객체를 JSON 파일로 인코딩

```
with open('newsample.json', 'w', encoding='utf-8') as file:
    json.dump(result, file, indent=4)
```

JSON 데이터 읽고 쓰기

◆ pandas.read_json() 함수

- 단순한 데이터 구조의 JSON 데이터 셋은 DataFrame으로 변환할 수 있음

```
In [68]: !cat examples/example.json
[{"a": 1, "b": 2, "c": 3},
 {"a": 4, "b": 5, "c": 6},
 {"a": 7, "b": 8, "c": 9}]
```

```
In [69]: data = pd.read_json('examples/example.json')

In [70]: data
Out[70]:
   a  b  c
0  1  2  3
1  4  5  6
2  7  8  9
```

◆ DataFrame.to_json() 함수

- DataFrame에 저장된 데이터를 JSON으로 저장

```
In [71]: print(data.to_json())
{"a":{"0":1,"1":4,"2":7},"b":{"0":2,"1":5,"2":8},"c":{"0":3,"1":6,"2":9}}

In [72]: print(data.to_json(orient='records'))
[{"a":1,"b":2,"c":3},{"a":4,"b":5,"c":6},{"a":7,"b":8,"c":9}]
```

JSON 데이터 읽고 쓰기

◆ 복잡한 구조의 JSON 데이터 셋의 DataFrame으로 변환

- pandas.json_normalize() 함수는 특정 키 값에 해당하는 DataFrame 생성
  - json의 load()/loads() 함수로 읽어 사전 데이터 생성 후 적용

```
import pandas as pd
import json

with open('sample.json', 'r') as file:
    data_dict = json.load(file)

df = pd.json_normalize(data_dict)

print(df)
```

```
  name                     place_lived   pet  \
0  Wes  [United States, Spain, Germany]  None

                                            siblings
0  [{'name': 'Scott', 'age': 25, 'pet': 'Zuko'}, ...
```

* JSON 구조가 복잡하지 않은 경우 직접 DataFrame 생성 가능

```
df = pd.DataFrame(data_dict)
```

JSON 파일에 저장

```
df.to_json("newexample.json", orient='records')
```

3.3 Requests 라이브러리 소개 및 설치

웹 API 사용하여 JSON 데이터 수신

◆ Requests 라이브러리

- 파이썬에서 HTTP 요청을 보내기 위해 널리 사용되는 라이브러리
- 간단하고 직관적인 인터페이스 제공하여 GET, POST, PUT, DELETE 등의 HTTP 메소드 사용이 용이함
- 주요 특징
  - 간단한 HTTP 요청/응답 처리
  - 다양한 HTTP 메소드 지원 (GET/POST/PUT/DELETE 등)
  - 요청 매개변수와 헤더 설정
  - JSON 데이터 자동 파싱
  - 타임아웃 및 예외처리 기능

- 라이브러리 설치

```
pip install requests
```

웹 API 사용하여 JSON 데이터 수신

◆ Requests 기본 사용법

- GET 요청 보내기
  - 서버에 데이터를 요청할 때 get() 함수 사용

```
import requests
url = 'https://api.github.com/repos/pydata/pandas/milestones/28/labels'
response = requests.get(url)
if response.status_code == 200:
    print(response.text)
else:
    print(response.status_code)
```

```
[{"id":76811,"node_id":"MDU6TGFiZWw3NjgxMQ==","url":"https://api.gi
thub.com/repos/pandas-
dev/pandas/labels/Bug","name":"Bug","color":"e10c02","default":fals
e,"description":null},
{"id":76812,"node_id":"MDU6TGFiZWw3NjgxMg==","url":"https://api.git
hub.com/repos/pandas-
dev/pandas/labels/Enhancement","name":"Enhancement","color":"4E9A06
","default":false,"description":null},
```

- json 데이터 변환시 json() 함수

```
result = response.json()
print(result)
```

```
[{'id': 76811, 'node_id': 'MDU6TGFiZWw3NjgxMQ==', 'url':
'https://api.github.com/repos/pandas-dev/pandas/labels/Bug',
'name': 'Bug', 'color': 'e10c02', 'default': False, 'description':
None}, {'id': 76812, 'node_id': 'MDU6TGFiZWw3NjgxMg==', 'url':
'https://api.github.com/repos/pandas-
dev/pandas/labels/Enhancement', 'name': 'Enhancement', 'color':
```

웹 API 사용하여 JSON 데이터 수신

◆ Requests 기본 사용법

- DataFrame 생성

```
import pandas as pd

issue_labels = pd.DataFrame(result)
issue_labels
```

	id	node_id	url	name	color	default	description
0	76811	MDU6TGFiZWw3NjgxMQ==	https://api.github.com/repos/pandas-dev/pandas...	Bug	e10c02	False	None
1	2301354	MDU6TGFiZWwyMzAxMzU0	https://api.github.com/repos/pandas-dev/pandas...	IO Data	06909A	False	IO issues that don't fit into a more specific ...
2	31404521	MDU6TGFiZWwzMTQwNDUyMQ==	https://api.github.com/repos/pandas-dev/pandas...	Dtype Conversions	e102d8	False	Unexpected or buggy dtype conversions
3	47232590	MDU6TGFiZWw0NzIzMjU5MA==	https://api.github.com/repos/pandas-dev/pandas...	IO SQL	5319e7	False	to_sql, read_sql, read_sql_query
4	76812	MDU6TGFiZWw3NjgxMg==	https://api.github.com/repos/pandas-dev/pandas...	Enhancement	4E9A06	False	None
5	127681	MDU6TGFiZWwxMjc2ODE=	https://api.github.com/repos/pandas-dev/pandas...	Refactor	FCE94F	False	Internal refactoring of code
6	2822098	MDU6TGFiZWwyODIyMDk4	https://api.github.com/repos/pandas-dev/pandas...	Indexing	0b02e1	False	Related to indexing on series/frames, not to i...

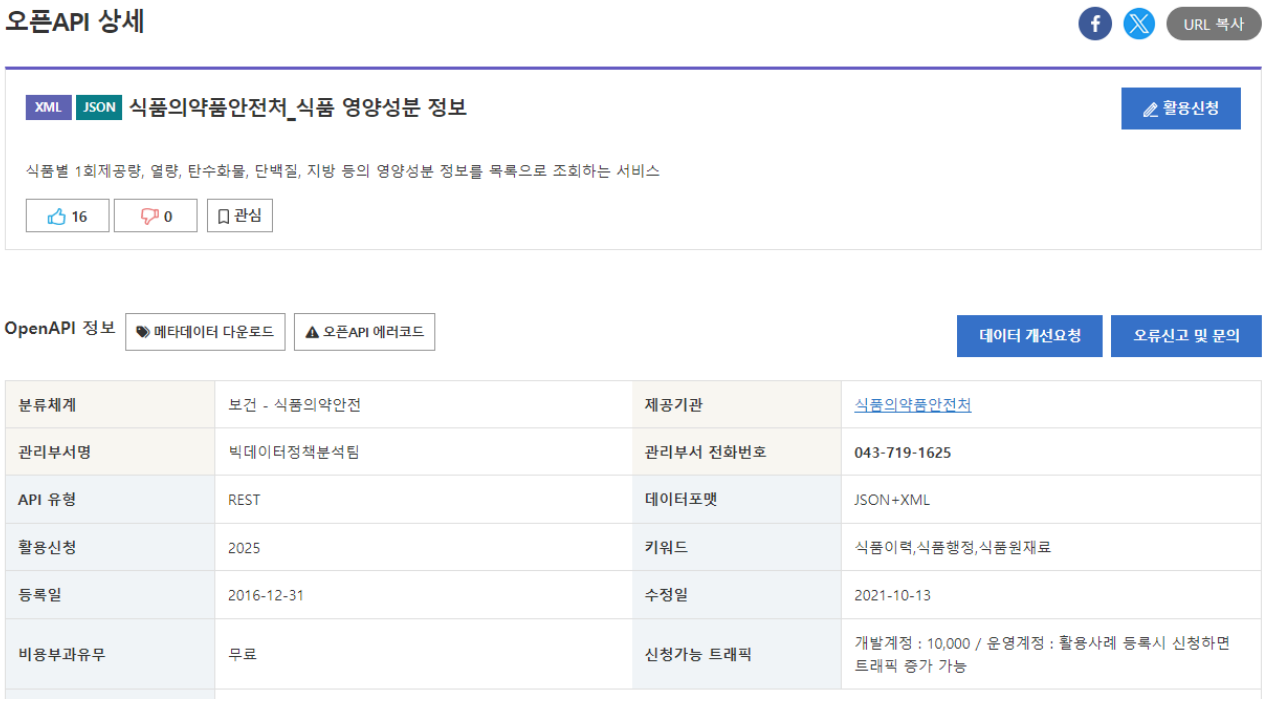
공공 데이터 API를 통한 JSON 데이터 처리

◆ 공공데이터 포털

- url : https://www.data.go.kr

◆ 식품의약품 안전처_식품 영양성분 정보

- 오픈API 상세 페이지



오픈API 상세

URL 복사

XML JSON 식품의약품안전처_식품 영양성분 정보

활용신청

식품별 1회제공량, 열량, 탄수화물, 단백질, 지방 등의 영양성분 정보를 목록으로 조회하는 서비스

16 0 관심

OpenAPI 정보 메타데이터 다운로드 오픈API 에러코드

데이터 개선요청 오류신고 및 문의

분류체계	보건 - 식품의약안전	제공기관	식품의약품안전처
관리부서명	빅데이터정책분석팀	관리부서 전화번호	043-719-1625
API 유형	REST	데이터포맷	JSON+XML
활용신청	2025	키워드	식품이력,식품행정,식품원재료
등록일	2016-12-31	수정일	2021-10-13
비용부과유무	무료	신청가능 트래픽	개발계정 : 10,000 / 운영계정 : 활용사례 등록시 신청하면 트래픽 증가 가능

공공 데이터 API를 통한 JSON 데이터 처리

◆ 데이터 요청 예시

- API key와 요청하는 데이터에 대한 정보를 인자(params)로 전달

```
apikey = "llj[redacted]
url = 'http://apis.data.go.kr/1471000/FoodNtrIrdntInfoService1/getFoodNtrItdntList1'
params ={
    'serviceKey' : apikey,
    'desc_kor' : '바나나칩',
    'pageNo' : '1',
    'numOfRows' : '3',
    'bgn_year' : '2017',
    'animal_plant' : '(유)돌코리아',
    'type' : 'json'
    }

response = requests.get(url, params=params)

if response.status_code == 200:
    result = response.json()
    print(result)
else:
    print('실패 : ',response.status_code)
```

3.4 XML 데이터 처리

XML 데이터

◆ XML이란?

- XML(eXtensible Markup Language)는 데이터를 구조화하고 저장하기 위한 마크업 언어
  - 데이터 교환에 대한 표준
  - HTML과 유사하지만, 데이터의 표현에 중점을 두고 있음

◆ 특징

- 유연성
  - 사용자 정의 태그를 사용하여 데이터를 표현할 수 있음
- 가독성
  - 사람이 읽고 쓸 수 있는 형태로 데이터를 표현
- 데이터 교환
  - 다양한 시스템 간에 데이터 교환을 용이하게 함

XML의 구조와 문법

◆ XML의 주요 요소

- 루트 엘리먼트 : 모든 XML 문서에는 단 하나의 루트 엘리먼트가 있어야 함
- 자식 엘리먼트 : 루트 엘리먼트 내에 포함된 엘리먼트
- 텍스트 노드 : 엘리먼트 내의 텍스트

◆ sample.xml

```
<?xml version="1.0" encoding="utf-8"?>
<customers>
    <customer>
        <name>홍길동</name>
        <address>서울 강남구</address>
    </customer>
    <customer>
        <name>고길동</name>
        <address>서울 강북구</address>
    </customer>
    <customer>
        <name>김길동</name>
        <address>서울 서초구</address>
    </customer>
</customers>
```

XML 데이터 읽기

◆ 파이썬 표준 라이브러리 xml 모듈

- import xml.etree.ElementTree as ET
- String으로 부터 XML 데이터 읽기

```
root = ET.fromstring(xml_data)

print(ET.tostring(root, encoding='utf-8').decode('utf-8'))
```

엘리먼트를 바이트 문자열로 변환한 후 문자열로 변환

```
<customers>
    <customer>
        <name>홍길동</name>
        <address>서울 강남구</address>
    </customer>
    <customer>
        <name>고길동</name>
        <address>서울 강북구</address>
    </customer>
    <customer>
        <name>김길동</name>
        <address>서울 서초구</address>
    </customer>
</customers>
```

```
xml_data = '''
<customers>
    <customer>
        <name>홍길동</name>
        <address>서울 강남구</address>
    </customer>
    <customer>
        <name>고길동</name>
        <address>서울 강북구</address>
    </customer>
    <customer>
        <name>김길동</name>
        <address>서울 서초구</address>
    </customer>
</customers>
'''
```

XML 데이터 읽기

◆ 파이썬 표준 라이브러리 xml 모듈

- import xml.etree.ElementTree as ET
- String으로 부터 데이터 읽기

```
root = ET.fromstring(xml_data)

print(f'루트 엘리먼트: {root.tag}')

for child in root:
    print(child.tag)
    for item in child:
        print(f'{item.tag}: {item.text}')
```

```
루트 엘리먼트: customers
customer
name: 홍길동
address: 서울 강남구
customer
name: 고길동
address: 서울 강북구
customer
name: 김길동
address: 서울 서초구
```

```
xml_data = '''
<customers>
   <customer>
      <name>홍길동</name>
      <address>서울 강남구</address>
   </customer>
   <customer>
      <name>고길동</name>
      <address>서울 강북구</address>
   </customer>
   <customer>
      <name>김길동</name>
      <address>서울 서초구</address>
   </customer>
</customers>
'''
```

XML 데이터 읽기

◆ 파이썬 표준 라이브러리 xml 모듈

- import xml.etree.ElementTree as ET
- XML 파일으로 부터 데이터 읽기

```
tree = ET.parse('sample.xml')

root = tree.getroot()
print(f'루트 엘리먼트: {root.tag}')

for child in root:
    print(child.tag)
    for item in child:
        print(f'{item.tag}: {item.text}')
```

```
루트 엘리먼트: customers
customer
name: 홍길동
address: 서울 강남구
customer
name: 고길동
address: 서울 강북구
customer
name: 김길동
address: 서울 서초구
```

sample.xml

```
<?xml version="1.0" encoding="utf-8"?>
<customers>
   <customer>
      <name>홍길동</name>
      <address>서울 강남구</address>
   </customer>
   <customer>
      <name>고길동</name>
      <address>서울 강북구</address>
   </customer>
   <customer>
      <name>김길동</name>
      <address>서울 서초구</address>
   </customer>
</customers>
```

XML 데이터 읽기

- 파이썬 표준 라이브러리 xml 모듈
  - 검색하기
    - find(path) : 첫번째 요소 반환 / findall(path) : 모든 요소 반환
    - findtext(path) : 첫번째 요소의 텍스트 반환
    - path 스트링
      - './name' : 현재 노드의 바로 아래 자식 중에서 'name' 검색
      - './/name' : 현재 노드의 모든 하위 경로에서 'name' 검색

```
tree = ET.parse('sample.xml')

root = tree.getroot()

result = root.findall('.//name')
for name in result:
    print(name.text)
```

```
<?xml version="1.0" encoding="utf-8"?>
<customers>
   <customer>
      <name>홍길동</name>
      <address>서울 강남구</address>
   </customer>
   <customer>
      <name>고길동</name>
      <address>서울 강북구</address>
   </customer>
   <customer>
      <name>김길동</name>
      <address>서울 서초구</address>
   </customer>
</customers>
```

XML 데이터 파일 저장하기

- 파이썬 표준 라이브러리 xml 모듈 이용하여 파일 저장하기

```
root = ET.Element("customers")
customer = ET.SubElement(root, "customer")
name = ET.SubElement(customer, "name")
name.text = "홍길동"
address = ET.SubElement(customer, "address")
address.text = "서울 강남구"
customer = ET.SubElement(root, "customer")
name = ET.SubElement(customer, "name")
name.text = "고길동"
address = ET.SubElement(customer, "address")
address.text = "서울 강북구"

tree = ET.ElementTree(root)
tree.write("output.xml", encoding="utf-8", xml_declaration=True)
```

XML 데이터 읽기

- lxml 라이브러리
  - 매우 빠른 파싱 속도를 제공하고, 대규모 XML 문서 처리에 적합
  - 설치 : pip install lxml

  - 문자열로 부터 데이터 읽기

```
from lxml import etree

root = etree.fromstring(xml_data)
print(f'루트 엘리먼트: {root.tag}')

for child in root:
    print(child.tag)
    for item in child:
        print(f'{item.tag}: {item.text}')
```

```
루트 엘리먼트: customers
customer
name: 홍길동
address: 서울 강남구
customer
name: 고길동
address: 서울 강북구
customer
name: 김길동
address: 서울 서초구
```

XML 데이터 읽기

- lxml 라이브러리
  - 파일로부터 XML 파일 읽기

```
tree = etree.parse('sample.xml')

root = tree.getroot()
print(f'루트 엘리먼트: {root.tag}')

for child in root:
    print(child.tag)
    for item in child:
        print(f'{item.tag}: {item.text}')
```

```
루트 엘리먼트: customers
customer
name: 홍길동
address: 서울 강남구
customer
name: 고길동
address: 서울 강북구
customer
name: 김길동
address: 서울 서초구
```

XML 데이터 쓰기

- lxml 라이브러리
  - XML 파일에 쓰기

```
new_customer = etree.Element('customer')
name = etree.SubElement(new_customer, "name")
name.text = "고길동"
address = etree.SubElement(new_customer, "address")
address.text = "서울 강북구"

root.append(new_customer)

tree.write('modified_sample.xml', encoding='UTF-8', xml_declaration=True)
```

웹 API를 사용하여 XML 데이터 수신

◆ RSS (Really Simple Syndication)

- Rich Site Summary라고도 함
- 뉴스나 블로그와 같이 컨텐츠 업데이트가 자주 일어나는 웹사이트에서, 업데이트된 정보를 자동으로 쉽게 사용자에게 제공하기 위한 서비스
- XML 데이터 형식

◆ 사이트 예

https://fs.jtbc.co.kr/RSS/culture.xml

```
response = requests.get(url)
root = etree.fromstring(response.content)
```

공공 데이터 API를 통한 XML 데이터 처리

◆ 식품 영양정보 – 바나나칩 열량 출력 예시

```
import requests
from lxml import etree

apikey = "[redacted]
url = 'http://apis.data.go.kr/1471000/FoodNtrIrdntInfoService1/getFoodNtrItdntList1'
params ={
    'serviceKey' : apikey,
    'desc_kor' : '바나나칩',
    'pageNo' : '1',
    'numOfRows' : '3',
    'bgn_year' : '2017',
    'animal_plant' : '(유)돌코리아',
    'type' : 'xml'
    }

response = requests.get(url, params=params)
if response.status_code == 200:
    root = etree.fromstring(response.content)
    kcal = root.find('.//NUTR_CONT1')
    print(kcal.text)
else:
    print('실패 : ', response.status_code)
```

감사합니다

아이리포