

PJT명	파이썬을 활용한 API 데이터 수집	
단계	01 PJT	
진행일자	2025.07.25	
예상 구현 시간	필수기능	5H
	심화기능	3H

## 1. 목표

- Python 기초 문법을 이해하고 활용할 수 있다.
- 외부 API(JSON) 사용법을 이해하고 활용할 수 있다.
- JSON 데이터를 분석하고, 원하는 형태로 가공할 수 있다.

## 2. 준비사항

### 1) 사용 데이터

- OpenWeatherMap API: <https://openweathermap.org/api>
- API 종류: Current Weather Data

### Current & Forecast weather data collection

#### Current Weather Data

API doc

Subscribe

- Access current weather data for any location
- We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations
- JSON, XML, and HTML formats
- Included in both free and paid subscriptions

#### Hourly Forecast 4 days

API doc

Subscribe

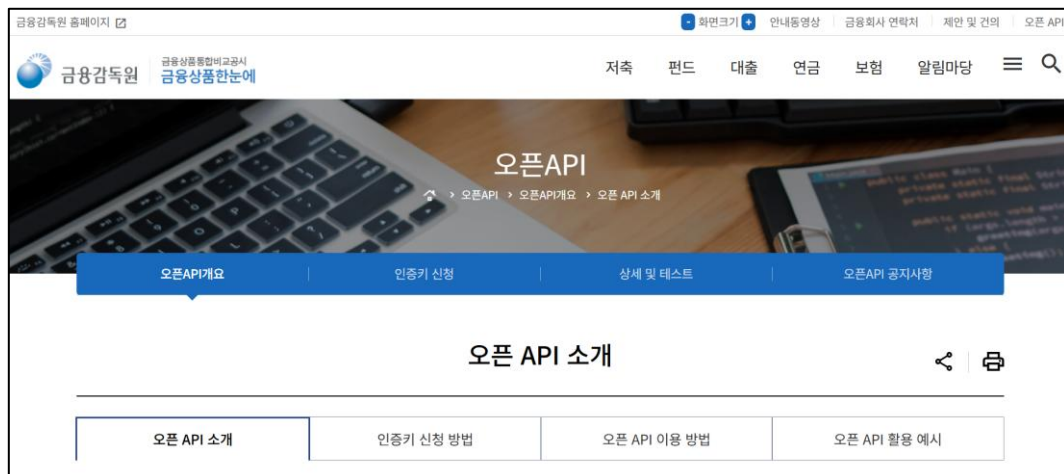
- Hourly forecast is available for 4 days
- Forecast weather data for 96 timestamps
- JSON and XML formats
- Included in the Developer, Professional and Enterprise subscription plans

## JSON

JSON format API response example

```
{
  "coord": {
    "lon": 7.367,
    "lat": 45.133
  },
  "weather": [
    {
      "id": 501,
      "main": "Rain",
      "description": "moderate rain",
      "icon": "10d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 284.2,
    "feels_like": 282.93,
    "temp_min": 283.06,
    "temp_max": 286.82,
```

- 금융상품통합비교공시 API (심화)
- <https://finlife.fss.or.kr/finlife/main/contents.do?menuNo=700029>
- API 종류: 정기예금 API



### 예제 URL(JSON)

[http://finlife.fss.or.kr/finlifeapi/depositProductsSearch.json?auth=\[발급받은 인증키\]&topFinGrpNo=020000&pageNo=1](http://finlife.fss.or.kr/finlifeapi/depositProductsSearch.json?auth=[발급받은 인증키]&topFinGrpNo=020000&pageNo=1)

### 예제 요청결과(JSON)

```
{
  "result": {
    "prdt_div": "D",
    "total_count": "79",
    "max_page_no": "1",
    "now_page_no": "1",
    "err_cd": "000",
    "err_msg": "정상",
    "baseList": [
      {
        "dcls_month": "201609",
        "fin_co_no": "0010001",
        "fin_prdt_cd": "WR0001A",
        "kor_co_nm": "우리은행",
        "fin_prdt_nm": "우리웰리치 주거래예금",
        "join_way": "영업점, 인터넷, 스마트폰",
        "mtrt_int": "만기 후\n- 1개월이내: 만기시점약정이율×50%\n- 1개월초과 6개월이내: 만기시점약정이율×30%\n- 6개월초과: 만기시점약정이율×20%\n※ 만기시점 약정이율: 일반정기예금 금리",
        "spcl_cnd": "다음 중 하나 충족한 입금건에 대해 최고 연0.2%p\n1. 순신규고객\n2. 가계대출이용고객\n3. 입금일 전월 주거래우대조건 2가지이상\n4. 건별3천만원이상\n5. 건별 만기 자동재예치",
        "join_member": "실명의 개인",
        "etc_note": "- 추가입금은 신규가입 시 선택한 예치기간을 각 입금건별 입금일로부터 적용\n- 재예치는 입금건별 최초 입금일로부터 최장 10년간 가능",
        "max_limit": null,
        "dcls_strt_day": "20160920",
        "dcls_end_day": null,
        "fin_co_subm_day": "201609201028",
        "dcls_month": "201609",
        "fin_co_no": "0010001",
        "fin_prdt_cd": "WR0001B",
        "kor_co_nm": "우리은행",
        "fin_prdt_nm": "우리웰리치100예금(회전형)",
        "join_way": "영업점, 인터넷,
```

2) 필수 준비사항

- OpenWeatherMap API KEY
- 금융감독원 API KEY

3) 개발언어 및 툴

- Python (VSCode)

4) 필수 라이브러리 / 오픈소스

- requests (파이썬 패키지)

### 3. 작업 순서

- 1) 팀원과 같이 요구사항(기본/추가/심화)을 확인하고, GitLab에 프로젝트를 생성한다.
  - 프로젝트 이름은 01-pjt로 지정한다.
  - 각 반 담당 강사님을 Maintainer로 설정한다.
- 2) 페어와 함께 요구사항을 분석하고 필요한 코드를 파악한다.
- 3) 필수 요구사항을 구현한다.
- 4) [선택] 심화 과제에 대해 페어와 함께 정리 후 구현한다.
- 5) README를 작성한다.
- 6) 제출 기한에 맞춰 모든 산출물이 GitLab에 업로드한다.

#### 4. 요구사항

본 프로젝트는 외부 API(OpenWeatherMap, 금융상품통합비교공시 API)를 통해 데이터를 수집하고, 해당 데이터를 분석 및 가공하여 사용자 맞춤형 날씨 서비스로 확장할 수 있는 기반을 마련하는 것을 목표로 한다.

단순한 API 연동을 넘어서 데이터의 구조를 이해하고, 필요한 정보를 추출하며, 사용 목적에 맞게 가공하는 일련의 과정을 수행한다.

또한, 팀원과 협업하여 아래 요구사항을 구현하고, 향후 실사용 서비스를 위한 기술을 익힌다.

이번 관통 프로젝트는 금융 상품 비교 애플리케이션 프로젝트의 필수 기능 중 하나인 외부 API를 통한 데이터 수집 및 가공 기능을 중점적으로 구현한다.

이를 통해 현실 데이터 기반의 서비스 설계 경험을 쌓고, 다양한 외부 데이터를 목적에 맞게 정제 및 가공하여 활용할 수 있는 데이터 처리 역량을 강화하는 것을 목표로 한다.

##### ● 요구사항 명세

- 아래에 제시된 기능 명세에 맞는 기능을 구현한다.
- 필수 기능은 반드시 구현해야 하며, 심화 기능은 팀 내 기획에 따라 선택적으로 구현한다.

번호	분류	요구사항명	요구사항 상세	우선순위
기능적 요구사항				
F01	날씨	Key 값 출력	날씨 데이터의 응답을 JSON 형태로 변환 후 Key 값만 출력하도록 구성	필수
F02	날씨	원하는 값 추출	main, weather 키 값을 딕셔너리로 추출	필수
F03	날씨	Key 한글화	추출한 키 값을 한글 키로 변환한 새로운 딕셔너리 생성	필수
F04	날씨	섭씨 온도 추가	F03의 결과 딕셔너리에 온도 관련 값을 이용해 섭씨 데이터 필드 추가	필수

F05	날씨	생성형 AI 활용	OpenWeatherMap API에 대한 프롬프트 구성 및 실행	필수
F06	금융	Key 값 출력	전체 정기예금의 응답을 JSON 형태로 변환 후 Key 값만 출력하도록 구성	심화
F07	금융	정기예금 리스트 추출	전체 응답 중 정기예금 상품 리스트 정보만 출력	심화
F08	금융	옵션 정보 가공	전체 응답 중 정기예금 상품들의 옵션 리스트를 출력	심화
F09	금융	상품+옵션 통합	금융 상품 + 옵션 정보 딕셔너리로 가공	심화
F10	금융	생성형 AI 활용	금융 API 기반 프롬프트 작성 및 결과 출력	심화
...	...	...	...	...
비 기능적 요구사항				
NF01	환경설정	API Key 발급	API 사용을 위해 공식 사이트에서 API Key를 발급받고, 코드로 활용할 수 있도록 구성	필수
NF02	UX	코드 사용성	코드가 직관적으로 작성되어야 하며, 주요 로직에 주석이 포함되어야 함	필수
NF03	효율성	함수 구조화	기능별로 함수가 분리되어 있어야 하며, 코드 중복을 최소화해야 함	필수
NF04	문서화	README 작성	README.md에 구현 설명, 학습 내용, 느낀 점 등을 상세히 기록해야 함	필수
...	...	...	...	...

### 1) 기본(필수) 기능

이번 프로젝트는 OpenWeatherMap API를 활용하여 현재 날씨 데이터를 수집하고, 해당 데이터를 정제 및 가공하여 다양한 형태로 활용 가능한 데이터 처리 로직을 구현한다.

사용자는 응답 받은 JSON 데이터에서 원하는 키를 추출하고, 한글화 및 추가 가공을 통해 정보를 읽기 쉽게 정리할 수 있어야 한다.

## A. 준비하기

- OpenWeatherMap API 구조 및 사용법 문서 확인

## B. 데이터 수집 및 분석

- 날씨 데이터 수집 및 JSON 형태 응답 확인
  - API 로부터 불러온 날씨 데이터를 모두 출력한다

## C. 데이터 파싱

- 응답 데이터에서 Key 값들만 따로 추출하여 출력 (요구사항 F01)
  - Key 값만 따로 출력하여 내부 데이터의 상세 구조를 파악한다.

- 결과물 예시

```
dict_keys(['coord', 'weather', 'base', 'main', 'visibility', 'wind', 'clouds', 'dt', 'sys', 'timezone', 'id', 'name', 'cod'])
```

- 특정 데이터만 추출하여 딕셔너리로 구성하여 출력 (요구사항 F02)

- Key 값이 (main, weather) 인 데이터만 따로 구성

- 결과물 예시

```
{'main': {'feels_like': 299.05,
          'grnd_level': 998,
          'humidity': 100,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'pressure': 1007,
          'sea_level': 1007,
          'temp': 297.91,
          'temp_max': 298.93,
          'temp_min': 297.91},
 'weather': [{'description': 'overcast clouds',
               'icon': '04d',
               'id': 804,
               'main': 'Clouds'}]}
```

- Key 값들을 모두 한글로 변환 (요구사항 F03)

- Key 값들을 모두 한글로 변환한 새로운 딕셔너리를 구성하여 출력한다.

- 결과물 예시

```
{'기본': {'None': 998,
          '기압': 1007,
          '습도': 100,
          '온도': 297.91,
          '체감온도': 299.05,
          '최고온도': 298.93,
          '최저온도': 297.91},
 '날씨': [{'식별자': 804, '아이콘': '04d', '요약': 'overcast clouds',
          '핵심': 'Clouds'}]}
```

#### D. 데이터 가공

- 한글로 변환된 데이터에 섭씨 온도 데이터를 추가 (요구사항 F04)

- 결과물 예시

```
{'기본': {'None': 998,
          '기압': 1007,
          '습도': 100,
          '온도': 297.91,
          '온도(섭씨)': 24.76,
          '체감온도': 299.05,
          '체감온도(섭씨)': 25.9,
          '최고온도': 298.93,
          '최고온도(섭씨)': 25.78,
          '최저온도': 297.91,
          '최저온도(섭씨)': 24.76},
 '날씨': [{'식별자': 804, '아이콘': '04d', '요약': 'overcast clouds',
          '핵심': 'Clouds'}]}
```

#### E. 생성형 AI 활용하기 (요구사항 F05)

- 생성형 AI가 OpenWeatherMap API를 이해하고 설명할 수 있도록 프롬프트를 구성
- 날씨와 관련된 아이디어를 구성하고, 생성형 AI를 통해 해당 내용의 정답을 얻을 수 있도록 프롬프트 구성
  - 질문 예시: 서울의 3일 후 날씨를 알기 위해서 어떻게 요청을 보내야 해?

생성형 AI를 제외한 전체 기능은 함수 단위로 구현하고, 사용자 입력 없이 실행 가능한 구조로 작성 (생성형 AI는 생성형 AI 서비스가 제공하는 화면에서 진행)

## 2) 심화 기능

추후 금융 상품 비교 애플리케이션 구현을 위해 금융감독원 금융상품통합비교공시 API를 활용하여 정기예금 상품 정보 및 옵션 정보를 수집하고, 해당 데이터를 정제 및 가공하여 활용 가능한 데이터 형태로 변환하는 로직을 구현.

사용자는 응답 받은 JSON 데이터에서 원하는 키를 추출하고, 한글화 및 추가 가공을 통해 정보를 읽기 쉽게 정리할 수 있어야 한다.

### A. 준비하기

- 금융상품통합비교공시 API 구조 및 사용법 문서 확인

### B. 데이터 수집 및 분석

- 정기예금 상품 데이터 수집 및 JSON 응답 확인
  - API로부터 불러온 예금 데이터를 모두 출력한다.

### C. 데이터 파싱

- 응답 데이터에서 Key 값들만 따로 추출하여 출력 (요구사항 F06)
  - Key 값만 따로 출력하여 내부 데이터의 상세 구조를 파악한다.
  - 결과물 예시

- ```
dict_keys(['prdt_div', 'total_count', 'max_page_no', 'now_page_no', 'err_cd', 'err_msg', 'baselist', 'optionList'])
```



- 정기예금 상품 리스트 정보만 추출하여 출력 (요구사항 F07)
  - 새로운 딕셔너리에 정기예금 상품 리스트 정보만 구성하여 출력
  - 결과물 예시

```
[{'dcls_end_day': None,
  'dcls_month': '202506',
  'dcls_strt_day': '20250709',
  'etc_note': '- 가입기간: 1~36개월\n'
              '- 최소가입금액: 1만원 이상\n'
              '- 만기일을 일,월 단위로 자유롭게 선택 가능\n'
              '- 만기해지 시 신규일 당시 영업점과 인터넷 홈페이지에 고시된 계약기간별
금리 적용',
  'fin_co_no': '0010001',
  'fin_co_subm_day': '202507091007',
  'fin_prdt_cd': 'WR0001B',
  'fin_prdt_nm': 'WON플러스예금',
  'join_deny': '1',
  'join_member': '실명의 개인',
  'join_way': '인터넷,스마트폰,전화(텔레뱅킹)',
  'kor_co_nm': '우리은행',
  'max_limit': None,
  'mtrt_int': '만기 후\n'
              '- 1개월이내 : 만기시점약정이율×50%\n'
              '- 1개월초과 6개월이내: 만기시점약정이율×30%\n'
              '- 6개월초과 : 만기시점약정이율×20%\n'
              '\n'
              '※만기시점 약정이율 : 일반정기예금 금리',
  'spcl_cnd': '해당사항 없음'},
 {'dcls_end_day': '99991231',
  'dcls_month': '202506',
  'dcls_strt_day': '20250620',
```

- 각 상품의 옵션 정보만 따로 추출하여 리스트로 가공 (요구사항 F08)
  - 옵션 정보: (금리유형, 금리, 기간 등)
  - 결과물 예시

```
[{'금융상품코드': 'WR0001B',
  '저축 금리': 2.5,
  '저축 기간': '1',
  '저축금리유형': 'S',
  '저축금리유형명': '단리',
  '최고 우대금리': 2.5},
 {'금융상품코드': 'WR0001B',
  '저축 금리': 2.5,
  '저축 기간': '3',
  '저축금리유형': 'S',
  '저축금리유형명': '단리',
  '최고 우대금리': 2.5},
 {'금융상품코드': 'WR0001B',
  '저축 금리': 2.5,
  '저축 기간': '6',
  '저축금리유형': 'S',
```

#### D. 데이터 가공

- 하나의 금융 상품에 여러 옵션 정보를 포함하는 구조로 가공하여 딕셔너리를 생성한다 (요구사항 F09)

■ 예: {금융회사명, 상품명, 옵션: [옵션1, 옵션2, ...]}

- 결과물 예시

```
[{'금리정보': [{'저축금리': 2.5,
                '저축기간': '1',
                '저축금리유형': 'S',
                '저축금리유형명': '단리',
                '최고우대금리': 2.5},
               {'저축금리': 2.5,
                '저축기간': '3',
                '저축금리유형': 'S',
                '저축금리유형명': '단리',
                '최고우대금리': 2.5},
               {'저축금리': 2.5,
                '저축기간': '6',
                '저축금리유형': 'S',
                '저축금리유형명': '단리',
                '최고우대금리': 2.5},
               {'저축금리': 2.5,
                '저축기간': '12',
                '저축금리유형': 'S',
                '저축금리유형명': '단리',
                '최고우대금리': 2.5},
               {'저축금리': 2.45,
                '저축기간': '24',
                '저축금리유형': 'S',
                '저축금리유형명': '단리',
                '최고우대금리': 2.45},
               {'저축금리': 2.45,
                '저축기간': '36',
                '저축금리유형': 'S',
                '저축금리유형명': '단리',
                '최고우대금리': 2.45}],
 '금융상품명': 'WON플러스예금',
 '금융회사명': '우리은행'}
```

#### E. 생성형 AI 활용하기

- 생성형 AI가 금융상품통합비교공시 API를 이해하고 설명할 수 있도록 프롬프트를 구성
- 적금, 예금 등 은행상품과 관련된 아이디어를 구성하고, 생성형 AI를 통해 해당 내용의 정답을 얻을 수 있도록 프롬프트 구성
  - 질문 예시: 금리가 가장 높은 적금 상품을 가져오려면 어떻게 해야 해? 예상되는 결과와 함께 알려줘

생성형 AI를 제외한 전체 기능은 함수 단위로 구현하고, 사용자 입력 없이 실행 가능한 구조로 작성 (생성형 AI는 생성형 AI 서비스가 제공하는 화면에서 진행)

#### 3) 분석/설계 예시

- 메인 코드 흐름

| 단계 | 설명 (특정 Python 파일 실행)                   |
|----|----------------------------------------|
| 1  | API 요청                                 |
| 2  | JSON 응답                                |
| 3  | 데이터 추출 → 요구 사항에 맞는 변환 (한글 변환, 온도 추가 등) |
| 4  | 콘솔 출력                                  |

## 5. 참고자료

- Requests 패키지 활용법 참고 문서 – MDN
  - <https://developer.mozilla.org/ko/docs/Web/API/Request>
- OpenWeatherMap API 공식 문서 (Current weather data)
  - <https://openweathermap.org/current>
- 금융감독원 금융상품통합비교공시 API (정기예금 data)
  - <https://finlife.fss.or.kr/finlife/api/fdrmDpstApi/list.do?menuNo=700052>

## 6. 결과

최종적으로 제출해야 할 항목은 아래와 같다

### 1. 구현 소스 코드

- 완성된 각 문제 별 소스코드 및 실행화면 캡처본
  - 콘솔 출력 결과 등

### 2. README.md

- 단계별로 구현 과정 중 학습한 내용, 어려웠던 부분, 새로 배운 것들 및 느낌 점을 상세히 기록

위 내용들을 모두 포함하여 GitLab에 업로드 한다.

GitLab 프로젝트 이름은 프로젝트 번호 + pjt 형식으로 지정한다 (01\_pjt)

제출 기한 - 18:00