# 파이썬 프로그래밍

## 4주차: 웹 데이터 수집과 API 활용

# 목차

# 1. requests로 웹 데이터 가져오기

# HTTP 기본 개념

### HTTP 메서드

- GET: 데이터 조회
- POST: 데이터 전송
- PUT/PATCH: 데이터 수정
- DELETE: 데이터 삭제

### 상태 코드

- 2xx: 성공 (200 OK)
- 3xx: 리다이렉션
- 4xx: 클라이언트 오류 (404 Not Found)
- 5xx: 서버 오류 (500 Internal Server Error)

# requests 라이브러리 기본

```python
import requests

# GET 요청
response = requests.get('https://api.github.com')
print(response.status_code)  # 200
print(response.json())  # JSON 응답을 딕셔너리로

# 파라미터 전달
params = {'q': 'python', 'sort': 'stars'}
response = requests.get('https://api.github.com/search/repositories',
                        params=params)

# POST 요청
data = {'username': 'user', 'password': 'pass'}
response = requests.post('https://httpbin.org/post', data=data)
```

# 헤더와 예외 처리

```
# 사용자 정의 헤더
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)'
}
response = requests.get('https://example.com', headers=headers)

# 타임아웃과 예외 처리
try:
    response = requests.get('https://api.example.com', timeout=5)
    response.raise_for_status()  # 4xx, 5xx 에러시 예외 발생
except requests.exceptions.Timeout:
    print("요청 시간 초과")
except requests.exceptions.RequestException as e:
    print(f"요청 오류: {e}")
```

# 2. BeautifulSoup으로 HTML 파싱

# BeautifulSoup 기초

```
from bs4 import BeautifulSoup
import requests

# HTML 가져오기
response = requests.get('https://example.com')
soup = BeautifulSoup(response.text, 'html.parser')

# 요소 찾기
title = soup.find('title').text
first_p = soup.find('p')  # 첫 번째 <p> 태그
all_links = soup.find_all('a')  # 모든 <a> 태그

# 속성 접근
for link in all_links:
    print(link.get('href'))  # href 속성값
```

# CSS 선택자 사용

```
# CSS 선택자로 요소 찾기
# 클래스 선택
articles = soup.select('.article')

# ID 선택
header = soup.select_one('#header')

# 복합 선택자
links_in_nav = soup.select('nav a')

# 속성 선택자
external_links = soup.select('a[target="_blank"]')

# 예시: 뉴스 제목 추출
headlines = soup.select('h2.headline')
for headline in headlines:
    print(headline.get_text(strip=True))
```

# 데이터 추출과 정제

```
# 텍스트 추출
text = element.get_text()  # 모든 텍스트
text = element.get_text(strip=True)  # 공백 제거
text = element.string  # 직접 포함된 텍스트만

# 태그 네비게이션
parent = element.parent
children = list(element.children)
siblings = element.find_next_siblings()

# 데이터 정제 예시
def clean_price(price_text):
    # "₩1,234,567" -> 1234567
    import re
    return int(re.sub(r'[^\d]', '', price_text))
```

# 3. API 활용하기

# REST API 이해

### REST (Representational State Transfer)

- 자원(Resource): URI로 표현
- 행위(Verb): HTTP 메서드로 표현
- 표현(Representation): JSON, XML 등

### 엔드포인트 예시

```
GET    /users          # 사용자 목록
GET    /users/123      # 특정 사용자
POST   /users          # 사용자 생성
PUT    /users/123      # 사용자 수정
DELETE /users/123      # 사용자 삭제
```

# API 인증 방식

```
# API 키 인증 (쿼리 파라미터)
params = {'apikey': 'YOUR_API_KEY'}
response = requests.get('https://api.example.com/data', params=params)

# API 키 인증 (헤더)
headers = {'X-API-Key': 'YOUR_API_KEY'}
response = requests.get('https://api.example.com/data', headers=headers)

# Bearer 토큰 인증
headers = {'Authorization': 'Bearer YOUR_TOKEN'}
response = requests.get('https://api.example.com/data', headers=headers)
```

# JSON 응답 처리

```python
import requests
import json

# API 호출
response = requests.get('https://api.github.com/users/python')

# JSON 파싱
data = response.json()

# 데이터 접근
print(data['name'])  # Python
print(data['public_repos'])  # 공개 저장소 수

# 에러 처리
if response.status_code == 200:
    data = response.json()
else:
    print(f"API 오류: {response.status_code}")
    error_data = response.json()
    print(error_data.get('message', '알 수 없는 오류'))
```

# 4. 실습 프로젝트

# 날씨 정보 수집기

```python
import requests
import json
from datetime import datetime

class WeatherCollector:
    def __init__(self, api_key):
        self.api_key = api_key
        self.base_url = "http://api.openweathermap.org/data/2.5"

    def get_current_weather(self, city):
        endpoint = f"{self.base_url}/weather"
        params = {
            'q': city,
            'appid': self.api_key,
            'units': 'metric',
            'lang': 'kr'
        }

        response = requests.get(endpoint, params=params)
        if response.status_code == 200:
            return response.json()
        return None
```

# 날씨 데이터 처리

```
def parse_weather_data(self, data):
    if not data:
        return None

    return {
        '도시': data['name'],
        '온도': data['main']['temp'],
        '체감온도': data['main']['feels_like'],
        '습도': data['main']['humidity'],
        '날씨': data['weather'][0]['description'],
        '시간': datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    }

# 사용 예시
collector = WeatherCollector('YOUR_API_KEY')
weather = collector.get_current_weather('Seoul')
parsed_data = collector.parse_weather_data(weather)
print(parsed_data)
```

# 뉴스 헤드라인 스크래퍼

```
class NewsScaper:
    def __init__(self, base_url):
        self.base_url = base_url
        self.session = requests.Session()
        self.session.headers.update({
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)'
        })

    def get_headlines(self):
        response = self.session.get(self.base_url)
        soup = BeautifulSoup(response.text, 'html.parser')

        headlines = []
        articles = soup.select('article.news-item')

        for article in articles:
            headline = {
                'title': article.select_one('h2').text.strip(),
                'summary': article.select_one('p.summary').text.strip(),
                'url': article.select_one('a')['href']
            }
            headlines.append(headline)

        return headlines
```

# 데이터 저장

```python
import csv
import json

def save_to_csv(data, filename):
    """데이터를 CSV 파일로 저장"""
    if not data:
        return

    keys = data[0].keys()
    with open(filename, 'w', encoding='utf-8', newline='') as f:
        writer = csv.DictWriter(f, fieldnames=keys)
        writer.writeheader()
        writer.writerows(data)

def save_to_json(data, filename):
    """데이터를 JSON 파일로 저장"""
    with open(filename, 'w', encoding='utf-8') as f:
        json.dump(data, f, ensure_ascii=False, indent=2)

# 사용 예시
headlines = scraper.get_headlines()
save_to_json(headlines, 'news_headlines.json')
save_to_csv(headlines, 'news_headlines.csv')
```

# 마무리

## 학습한 내용

- **4주차**: 웹 데이터 수집 기술
  - HTTP 요청과 응답 처리
  - HTML 파싱과 데이터 추출
  - API 활용과 JSON 처리

# Q&A

## 질문 있으신가요?