파이썬 프로그래밍

22. Python 활용 심화 (1)

00 Contents

❖ 수업 목표

- os, shutil 모듈을 사용하여 파일 및 디렉터리를 관리할 수 있다.
- pathlib 모듈을 사용하여 파일 및 디렉터리를 관리할 수 있다.
- openpyxl 모듈을 활용하여 Excel 데이터를 읽고 쓰고 수정할 수 있다.
- pandas 라이브러리를 활용하여 Excel 데이터를 다룰 수 있다.
- requests, BeautifulSoup 을 활용하여 웹 스크래핑을 할 수 있다.

❖ 세부 목표

1. 파일관리 기본(os, shutil 활용)

❖ os 모듈

▪ os 모듈 개요

- os 모듈은 운영 체제와 상호 작용하는 기능을 제공
- 내장 모듈

▪ 불러오기

- 라이브러리 임포트
  - import os

▪ 주요 기능

- 현재 작업 디렉토리 확인 및 변경
  - # 현재 작업 디렉토리 확인
  - import os
  - print("현재 작업 디렉토리:", os.getcwd())

  - # 작업 디렉토리 변경
  - os.chdir("./ch22/os_module")

00

1. 파일관리 기본(os, shutil 활용)

❖ os 모듈

▪ 주요 기능

- 디렉토리 및 파일 목록 조회
  - # 현재 디렉토리의 파일 및 폴더 목록 출력
  - print("디렉토리 목록:", os.listdir("."))

- 디렉토리 생성 및 삭제
  - # 디렉토리 생성
  - os.mkdir("test_dir")

  - # 디렉토리 삭제
  - os.rmdir("test_dir")

- 파일 존재 여부 확인
  - if os.path.exists("file.txt"):
  - print("파일이 존재합니다.")

00

1. 파일관리 기본(os, shutil 활용)

❖ os 모듈

▪ 주요 기능

- 파일 및 디렉토리 경로 다루기

```
folder = os.getcwd()

# 경로 합치기
print(os.path.join(folder, "file.txt"))

# 파일명만 추출
print(os.path.basename(f"{folder}/file.txt"))

# 디렉토리 경로 추출
print(os.path.dirname(f"{folder}/file.txt"))
```

1. 파일관리 기본(os, shutil 활용)

❖ shutil 모듈

▪ shutil 모듈 개요

- 파일 및 디렉토리 관리
- shutil 모듈은 고수준 파일 조작 기능을 제공

▪ 불러오기

- 라이브러리 임포트
  - import shutil

▪ 주요 기능

- 파일 복사
  - import shutil
  - shutil.copy("source.txt", "destination.txt") # 파일 복사
  - shutil.copy2("source.txt", "destination.txt") # 메타데이터 유지하며 복사

00

1. 파일관리 기본(os, shutil 활용)

❖ shutil 모듈

▪ 주요 기능

- 디렉토리 복사
  - # 디렉토리 전체 복사
  - shutil.copytree("source_dir", "destination_dir")

- 파일 및 디렉토리 이동
  - shutil.move("old_location", "new_location")

- 파일 및 디렉토리 삭제
  - # 디렉토리 및 하위 파일 모두 삭제
  - shutil.rmtree("directory_to_delete")

1. 파일관리 기본(os, shutil 활용)

❖ 실습 문제

- 특정 폴더에서 텍스트 파일만 찾아 복사/이동/삭제하는 프로그램
  - 1. 특정 폴더에서 모든 .txt 파일을 검색하여 리스트로 반환하는 함수를 작성하시오.
    - param: 검색할 폴더 경로 (folder_path)
    - return: .txt 파일 리스트

1. 파일관리 기본(os, shutil 활용)

❖ 실습 문제

- 특정 폴더에서 텍스트 파일만 찾아 복사/이동/삭제하는 프로그램
  - 2. 특정 폴더에서 .txt 파일을 찾아 다른 폴더로 복사하는 함수를 작성하시오.
    - param: 원본 폴더 경로 (source_folder)
    - param: 대상 폴더 경로 (destination_folder)

1. 파일관리 기본(os, shutil 활용)

❖ 실습 문제

- 특정 폴더에서 텍스트 파일만 찾아 복사/이동/삭제하는 프로그램
  - 3. 특정 폴더에서 .txt 파일을 찾아 다른 폴더로 이동하는 함수를 작성하시오.
    - param: 원본 폴더 경로 (source_folder)
    - param: 대상 폴더 경로 (destination_folder)

1. 파일관리 기본(os, shutil 활용)

❖ 실습 문제

- 특정 폴더에서 텍스트 파일만 찾아 복사/이동/삭제하는 프로그램
  - 4. 특정 폴더에서 .txt 파일을 찾아 삭제하는 함수를 작성하시오.
    - param: 대상 폴더 경로 (destination_folder)

1. 파일관리 기본(os, shutil 활용)

❖ 실습 문제

- 특정 폴더에서 텍스트 파일만 찾아 복사/이동/삭제하는 프로그램
  - 5. 앞서 작성한 함수를 호출하여 프로그램을 테스트 하시오.

2. 파일 정리 프로그램 제작(pathlib 활용)

❖ pathlib 모듈

- pathlib 모듈 개요
  - 객체 지향 방식으로 파일 및 디렉터리 관리 가능

- 불러오기
  - 라이브러리 임포트
    - from pathlib import Path

- 주요 기능
  - pathlib의 기본
    - from pathlib import Path

    - # 현재 작업 디렉토리 확인
    - print(Path.cwd())

2. 파일 정리 프로그램 제작(pathlib 활용)

❖ pathlib 모듈

▪ 주요 기능

- 경로 생성 및 조작
  - path = Path("folder")

  - # 폴더와 파일 경로 결합
  - # '/' 연산자는 경로 결합 연산자(pathlib.Path 객체에서 지원)
  - print(path / "file.txt")

- 디렉토리 및 파일 존재 여부 확인
  - path = Path("file.txt")
  - path = Path("folder")

  - print(path.exists()) # 존재 여부 확인
  - print(path.is_file()) # 존재 및 파일 여부 확인
  - print(path.is_dir()) # 디렉토리 여부 확인

2. 파일 정리 프로그램 제작(pathlib 활용)

❖ pathlib 모듈

- 주요 기능 : 파일 및 디렉터리 조작
  - 디렉토리 생성 및 삭제
    - path = Path("new_folder")
    - path.mkdir(exist_ok=True) # 폴더 생성(기존 존재 시, 에러발생 없음)
    - path.rmdir() # 폴더 삭제
  - 파일 생성 및 삭제
    - file_path = Path("test.txt")
    - file_path.touch() # 빈 파일 생성
    - file_path.unlink() # 파일 삭제
  - 파일 및 폴더 목록 조회
    - path = Path(dir)
    - for item in path.iterdir():
    - print(item)

2. 파일 정리 프로그램 제작(pathlib 활용)

❖ pathlib 모듈

- 주요 기능 : 파일 및 디렉터리 조작
  - 파일 확장자 가져오기
    - path = Path(file)
    - print(path.suffix) # 출력: .txt
  - 파일 이름 변경 및 이동하기
    - path = Path("example.txt")
    - path.touch() # 파일 생성
    - destination = Path("new_folder/example.txt")
    - destination.parent.mkdir(exist_ok=True) # 대상 폴더 생성
    - path.rename(destination)

00

2. 파일 정리 프로그램 제작(pathlib 활용)

❖ 실습 문제

- 특정 폴더 내의 파일을 확장자별로 자동 분류하는 프로그램
  - 1. pathlib를 사용하여 특정 폴더 내의 모든 파일 목록을 반환하는 함수를 작성하시오.
    - param: 검색할 폴더 경로 (folder_path)
    - return: 파일 리스트

00

2. 파일 정리 프로그램 제작(pathlib 활용)

❖ 실습 문제

- 특정 폴더 내의 파일을 확장자별로 자동 분류하는 프로그램
  - 2. 특정 폴더 내의 파일을 확장자 별로 폴더를 생성하여 이동하는 함수를 작성하시오.
    - param: 정리할 폴더 경로 (source_folder)

2. 파일 정리 프로그램 제작(pathlib 활용)

❖ 실습 문제

- 특정 폴더 내의 파일을 확장자별로 자동 분류하는 프로그램
  - 3. 앞서 작성한 함수를 호출하여 프로그램을 테스트 하시오.

3. Excel 읽기/쓰기 기초(openpyxl)

❖ openpyxl 모듈

▪ openpyxl 모듈 개요

- 파이썬에서 Excel 파일을 읽고 쓰기 위해 사용되는 라이브러리
- xlsx 포맷을 지원

▪ 설치 및 불러오기

- 설치
  - pip install openpyxl
- 라이브러리 임포트
  - import openpyxl
  - from openpyxl import Workbook
  - from openpyxl import load_workbook

3. Excel 읽기/쓰기 기초(openpyxl)

❖ openpyxl 모듈

▪ 주요 기능 : 기본 사용법

- **새 Excel 파일 생성**
  - » Workbook(): 새 Excel 파일을 생성
  - » sheet['A1'] = '이름': A1 셀에 "이름"을 입력
  - » wb.save('new_example.xlsx'): 새 파일을 저장

```
wb = Workbook()              # 새로운 워크북 생성
sheet = wb.active            # 활성 시트 선택

# 셀에 데이터 입력
sheet['A1'] = 'Hello'
sheet['B1'] = 'World'

# 저장하기
wb.save('new_example.xlsx')
```

3. Excel 읽기/쓰기 기초(openpyxl)

❖ openpyxl 모듈

▪ 주요 기능 : 기본 사용법

- **Excel 파일 열기**
  - » load_workbook('파일명.xlsx'): 기존 Excel 파일을 로딩
  - » wb.active: 기본(첫 번째) 시트를 선택
  - » sheet['A1'].value: A1 셀의 값을 출력

```
# 파일 열기
wb = load_workbook('example.xlsx')

# 활성 시트 선택
sheet = wb.active

# 특정 셀의 값 읽기
print(sheet['A1'].value)
print(sheet['B1'].value)
```

3. Excel 읽기/쓰기 기초(openpyxl)

❖ openpyxl 모듈

▪ 주요 기능 : 기본 사용법

- 여러 셀의 데이터 읽기
  - » 1행부터 5행까지, 1열부터 3열까지 반복
  - » cell.value: 각 셀의 값을 출력

```
for row in sheet.iter_rows(min_row=1, max_col=3, max_row=5):
    for cell in row:
        print(cell.value)
```

- 실행 결과
  - » Hello
  - » World
  - » None
  - » None
  - » …
  - » None

3. Excel 읽기/쓰기 기초(openpyxl)

❖ openpyxl 모듈

▪ 주요 기능 : 기본 사용법

- **Excel 파일 수정 : 셀 값 수정하기**
  - » A1 셀의 데이터를 변경
  - » 변경된 내용을 새로운 파일로 저장

```
# A1 셀 값 수정
sheet['A1'] = '새로운 값'

# 저장하기
wb.save('modified_example.xlsx')
```

3. Excel 읽기/쓰기 기초(openpyxl)

❖ openpyxl 모듈

▪ 주요 기능 : 다양한 데이터 형식 다루기

- 숫자, 날짜, 수식 입력
  - » datetime(2025, 1, 28): 날짜 데이터 입력
  - » sheet['C1'] = '=B1*2': B1 값을 이용한 수식 입력

```
from openpyxl.utils import get_column_letter
from datetime import datetime

wb = Workbook()
sheet = wb.active

sheet['A1'] = datetime(2025, 1, 28)          # 날짜 입력
sheet['B1'] = 100                            # 숫자 입력
sheet['C1'] = '=B1*2'                        # 수식 입력

wb.save('formulas_example.xlsx')
```

3. Excel 읽기/쓰기 기초(openpyxl)

❖ 실습 문제

- **openpyxl 모듈을 활용하여 엑셀 파일(.xlsx)에 간단한 데이터 입력 및 읽기 프로그램 작성**
  - **1. 새로운 엑셀 파일을 생성하고 기본 데이터를 입력하는 함수를 작성하시오.**
    - **param: 생성할 엑셀 파일 경로(file_path)**

	A	B	C	D
1	이름	나이	국가	
2	홍길동	30	대한민국	
3	Alice	25	USA	
4	Bob	28	UK	
5				

3. Excel 읽기/쓰기 기초(openpyxl)

❖ 실습 문제

- openpyxl 모듈을 활용하여 엑셀 파일(.xlsx)에 간단한 데이터 입력 및 읽기 프로그램 작성
  - 2. 기존 엑셀 파일을 열어 데이터를 읽고 출력하는 함수를 작성하시오.
    - param: 읽을 엑셀 파일 경로(file_path)

3. Excel 읽기/쓰기 기초(openpyxl)

❖ 실습 문제

- openpyxl 모듈을 활용하여 엑셀 파일(.xlsx)에 간단한 데이터 입력 및 읽기 프로그램 작성
  - 3. 앞서 작성한 함수를 호출하여 프로그램을 테스트 하시오.

4. Excel 데이터 정리 자동화(openpyxl, pandas 활용)

❖ Pandas와 Excel 기본

- pandas와 excel 파일 기본 다루기
- DataFrame을 활용한 Excel 데이터 연산
- Excel 파일 저장 및 수정

▪ 설치 및 불러오기

- 설치
  - pip install pandas openpyxl

- 라이브러리 임포트
  - import pandas as pd
  - import openpyxl

4. Excel 데이터 정리 자동화(openpyxl, pandas 활용)

❖ pandas와 excel 파일 기본 다루기

▪ Excel 파일 읽기

- Excel 데이터를 pandas의 DataFrame으로 변환하여 처리
  - » Excel 파일을 불러와 DataFrame으로 변환
  - » 특정 시트를 선택하여 불러옴
  - » 데이터의 처음 5개 행을 출력

```
pd.read_excel("파일명.xlsx “, sheet_name="Sheet1 “)
df.head()
```

▪ Excel 파일 정보 확인

- » 데이터 타입, Null 값 여부 확인
- » 수치형 데이터의 평균, 최댓값, 최솟값 등 요약 정보 제공

```
print(df.info())              # 데이터 타입 및 개요 확인
print(df.describe())          # 숫자 데이터 요약
```

4. Excel 데이터 정리 자동화(openpyxl, pandas 활용)

❖ DataFrame을 활용한 Excel 데이터 연산

▪ 특정 열 선택 및 연산

» "나이" 컬럼의 데이터 출력
» 데이터의 합계 및 평균 계산

```
print(df["나이"])              # 특정 열 출력
print(df["나이"].sum())        # 총 판매량 합계
print(df["나이"].mean())  # 평균 판매량
```

▪ 새로운 열 추가 (파생 변수 생성)

» "출생년도" 컬럼을 새로 만들고, 기존 컬럼의 데이터를 활용하여 값 계산
» 년도에서 "나이"를 뺀 값 추가

```
df["출생년도"] = 2025 - df["나이"]
print(df.head())
```

4. Excel 데이터 정리 자동화(openpyxl, pandas 활용)

❖ Excel 파일 저장 및 수정

▪ DataFrame을 Excel 파일로 저장

» DataFrame을 Excel 파일로 저장
» 자동 생성된 index 컬럼을 제외

- # 자동 생성된 index 열을 제외하고 저장
- df.to_excel("modified_example.xlsx", index=False)

- 실행 결과

	A	B	C	D	E
1	이름	나이	국가	출생년도	
2	홍길동	30	대한민국	1995	
3	Alice	25	USA	2000	
4	Bob	28	UK	1997	
5					

4. Excel 데이터 정리 자동화(openpyxl, pandas 활용)

❖ 실습 문제

- pandas 모듈을 활용하여 학생 점수를 자동으로 계산하고 결과를 Excel 파일(.xlsx)에 저장하는 프로그램
  - 1. pandas를 사용하여 학생 데이터를 생성하는 함수를 작성하시오.
    - return: 학생 점수 데이터프레임 (df)

❖ 실습 문제

- pandas 모듈을 활용하여 학생 점수를 자동으로 계산하고 결과를 Excel 파일(.xlsx)에 저장하는 프로그램
  - 2. pandas를 사용하여 데이터를 엑셀 파일로 저장하는 함수를 작성하시오.
    - param: 저장할 데이터프레임(df)
    - param: 저장할 엑셀 파일 경로(file_path)

❖ 실습 문제

- pandas 모듈을 활용하여 학생 점수를 자동으로 계산하고 결과를 Excel 파일(.xlsx)에 저장하는 프로그램
  - 3. pandas를 사용하여 엑셀 파일에서 데이터를 읽고 출력하는 함수를 작성하시오.
    - param: 읽을 엑셀 파일 경로 (file_path)

❖ 실습 문제

- pandas 모듈을 활용하여 학생 점수를 자동으로 계산하고 결과를 Excel 파일(.xlsx)에 저장하는 프로그램
  - 4. 앞서 작성한 함수를 호출하여 프로그램을 테스트 하시오.

5. 웹 데이터 가져오기(requests, BeautifulSoup)

❖ 웹 스크래핑

▪ 웹 스크래핑 개요

- 웹에서 데이터를 자동으로 가져와서 활용하는 기술
- 뉴스, 쇼핑몰, 날씨 정보 등을 수집하는 데 활용

▪ 사용 모듈

- requests: 웹페이지 요청을 보내고 HTML을 가져옴
- beautifulsoup4: 가져온 HTML을 분석하고 필요한 데이터를 추출

- 설치 및 불러오기
  - 설치
    - » pip install requests beautifulsoup4
  - 라이브러리 임포트
    - » import requests
    - » from bs4 import BeautifulSoup

5. 웹 데이터 가져오기(requests, BeautifulSoup)

❖ requests를 이용한 웹페이지 요청

▪ 웹페이지 HTML 가져오기

- 해당 URL에서 HTML 페이지를 가져옴
  - » requests.get(url)
- 응답 상태 코드(2000이면 성공)
  - » response.status_code
- HTML 소스 코드 확인 가능
  - » response.text

▪ 예) 특정 사이트의 HTML 가져오기

```
url = "https://example.com"          # 크롤링할 웹페이지 URL
response = requests.get(url)         # GET 요청으로 페이지 가져오기

# 응답 코드 출력 (200 정상)
print(response.status_code)
# 가져온 HTML 일부 출력 (첫 500자)
print(response.text[:500])
```

5. 웹 데이터 가져오기(requests, BeautifulSoup)

❖ BeautifulSoup을 활용한 HTML 데이터 추출

▪ HTML 파싱 및 원하는 데이터 찾기

- 해당 URL에서 HTML 페이지를 가져옴
  - » requests.get(url)
- 응답 상태 코드(2000이면 성공)
  - » response.status_code
- HTML 소스 코드 확인 가능
  - » response.text
- # 파싱: 데이터를 분석하고 원하는 정보를 추출하는 과정

▪ 예) 특정 사이트의 HTML 가져오기

```
url = "https://example.com"          # 크롤링할 웹페이지 URL
response = requests.get(url)         # GET 요청으로 페이지 가져오기

# 응답 코드 출력 (200 정상)
print(response.status_code)
# 가져온 HTML 일부 출력 (첫 500자)
print(response.text[:500])
```

5. 웹 데이터 가져오기(requests, BeautifulSoup)

❖ BeautifulSoup을 활용한 HTML 데이터 추출

▪ 웹페이지에서 특정 태그의 데이터 가져오기 (find 함수 활용)

- 해당 태그의 첫 번째 요소 가져오기
  - » find("태그명")

- 특정 클래스를 가진 첫 번째 요소 가져오기
  - » find("태그명", class_="클래스명")

- 특정 ID를 가진 요소 가져오기
  - » find("태그명", id="아이디명")

5. 웹 데이터 가져오기(requests, BeautifulSoup)

❖ BeautifulSoup을 활용한 HTML 데이터 추출

▪ 웹페이지에서 특정 태그의 데이터 가져오기 (find 함수 활용)

- 예) 특정 태그의 데이터 가져오기

```
url = "https://example.com"
response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")

# 첫 번째 <h1> 태그 가져오기
header = soup.find("h1")
print(header.text)

# 특정 클래스(.title)를 가진 첫 번째 요소 가져오기
title = soup.find("div", class_="title")
print(title.text)

# 특정 ID를 가진 요소 가져오기
content = soup.find("div", id="content")
print(content.text)
```

5. 웹 데이터 가져오기(requests, BeautifulSoup)

❖ 실습 문제

- 뉴스 웹사이트에서 제목과 URL을 수집하는 프로그램
  - requests 모듈과 pandas 모듈, bs4 모듈의 BeautifulSoup 클래스를 사용하여 수집할 뉴스 URL 리스트를 작성하고 해당 URL의 뉴스 제목을 스크래핑 하여 URL과 제목 목록을 출력하는 프로그램을 작성하시오.

6. 간단한 데이터 수집 및 저장 프로그램

❖ pandas를 활용한 데이터 저장

▪ pandas로 CSV 파일 저장하기

- pandas 라이브러리를 사용하여 테이블 형식(DataFrame)을 저장 가능

- 예)
- 데이터를 pandas DataFrame으로 변환
  - pd.DataFrame(data)
- DataFrame을 CSV 파일로 저장
  - df.to_csv()
- 인덱스 열을 파일에 저장하지 않음
  - index=False

6. 간단한 데이터 수집 및 저장 프로그램

❖ pandas를 활용한 데이터 저장

▪ pandas로 CSV 파일 저장하기

- 예)

```
import pandas as pd

# 데이터 준비
data = {
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35],
    "City": ["New York", "Los Angeles", "Chicago"]
}
# DataFrame 생성
df = pd.DataFrame(data)

# CSV 파일로 저장
file = "./ch22/sava_file/output.csv"
df.to_csv(file, index=False)          # 인덱스 열 저장하지 않음
print(f"Data has been written to {file}")
```

6. 간단한 데이터 수집 및 저장 프로그램

❖ 실습 문제

- **뉴스 데이터를 정리하고 파일로 저장하는 프로그램**
  - 앞서 수집한 뉴스 URL과 제목 목록을 CSV 파일로 저장하는 소스코드를 추가하여 수정하시오.

Notice

❖ 과제

- 1. os, shutil 모듈의 함수 사용하여 파일 및 디렉터리 관리하기
- 2. pathlib 모듈의 함수를 사용하여 파일 및 디렉터리 관리하기
- 3. openpyxl 모듈의 함수를 사용하여 엑셀 생성 및 데이터 처리하기
- 4. requests, BeautifulSoup 을 활용하여 웹 스크래핑 하기

❖ 다음 수업 내용

- Python 활용 심화(2)
  - JSON과 API 활용 기초
  - API 데이터 활용
  - 정규 표현식 활용1
  - 정규 표현식 활용2
  - 데이터 통합 및 보고서 자동화