엑셀 문서 자동화

OpenPyXL

- **openpyxl**은 파이썬에서 Microsoft Excel 2010 이상의 파일을 읽고, 쓰고, 조작할 수 있는 라이브러리
- 이 라이브러리를 통해 사용자는 Excel 파일에 대한 자동화 작업을 수행할 수 있으며, 복잡한 데이터를 시각화하고, 데이터 분석 결과를 보고 가능
- 파이썬 엑셀 자동화는 openpyxl 라이브러리 사용

openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files — openpyxl 3.1.2 docume...

openpyxl is a Python library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files.

https://openpyxl.readthedocs.io/en/stable/index.html

- 라이브러리 설치

```
pip install openpyxl
```

워크북 생성

- 워크북은 Excel 파일 자체를 의미하고, 하나 이상의 워크시트(Worksheet)를 포함
- 워크시트는 워크북 내의 각 시트를 의미하고, 데이터를 저장하고 조작하는 데 사용

워크북 생성

- 파일 시스템에 파일을 만들 필요 없이 `Workbook` 클래스를 이용하여 바로 워크북 생성

```
from openpyxl import Workbook wb = Workbook()
```

활성 워크시트 접근

- 생성된 워크북은 적어도 하나의 워크시트를 포함
- `active` 속성으로 기본 워크시트에 접근

```
ws = wb.active
```

파일로 저장하기

- 워크북을 저장하는 가장 간단하고 안전한 방법은 `Workbook.save()` 메서드를 사용하는 것

```
wb = Workbook() wb.save('balances.xlsx')
```

- **경고:** 이 작업은 기존 파일을 경고 없이 덮어 쓸 수 있음
- **참고:** 파일 확장자는 반드시 xlsx 또는 xlsm일 필요는 없으나, 공식 확장자를 사용하지 않을 경우 다른 응용 프로그램에서 파일을 직접 열 때 문제가 발생할 수 있음

- Visual Studio Code에서 **Office Viewer(Markdown Editor)** 확장 프로그램 설치
- Visual Studio Code에서 엑셀 파일 조회 가능

새로운 워크시트생성

- `create_sheet()` 메소드로 워크북에 새로운 워크시트를 추가

```
ws1 = wb.create_sheet("Mysheet") # 마지막에 삽입 ws2 = wb.create_sheet("Mys
heet", 0) # 첫 번째에 삽입 ws3 = wb.create_sheet("Mysheet", -1) # 끝에서 두
번째에 삽입
```

워크시트 이름 변경

- 워크시트는 생성될 때 자동으로 이름이 부여되어 순차적으로 번호 할당
  - (Sheet, Sheet1, Sheet2, ...)

- `title` 속성을 사용하여 언제든지 워크시트의 이름을 변경

```
ws.title = "New Title"
```

- 특정 워크시트 접근

```
ws3 = wb["New Title"]
```

- 워크북의 모든 워크시트 이름을 Workbook.sheetnames 속성으로 확인

```
print(wb.sheetnames) # ['Sheet2', 'New Title', 'Sheet1']
```

- 워크시트를 순환하며 다루기

```
for sheet in wb: print(sheet.title)
```

데이터 다루기

특정 셀 접근

- 워크시트 내 특정 셀에 접근하는 방법은 주로 두 가지

직접 접근 방법

- 셀은 워크시트의 키로 직접 접근
- 예를 들어, 'A4' 셀에 접근하려면 다음과 같이 작성

```
c = ws['A4']
```

- 셀에 값 할당

```
ws['A4'] = 4
```

cell() 메서드 사용방법

- `Worksheet.cell()` 메서드는 행과 열 번호를 사용하여 셀에 접근
- 이 방법은 좀 더 동적인 셀 접근에 유용
- 예를 들어, 4번째 행, 2번째 열에 위치한 셀에 10이라는 값을 직접 할당

```
# row 행(1,2,3...), column 열(A,B,C...) ws.cell(row=4, column=2, value=10)
d = ws.cell(row=4, column=2)
```

셀 범위 접근

- 한 번에 여러 셀에 접근 필요할 때 사용
- 특정 범위 내의 모든 셀 값을 변경하거나 정보를 추출

- `cell.value` 속성은 셀에 저장된 데이터를 가져오거나 설정
- 'A1'에서 'D4'까지의 모든 셀에 "Hello World"를 입력하는 코드

```
for row in ws['A1:D4']: for cell in row: cell.value = 'Hello World'
```

- 실습 코드

```
from openpyxl import Workbook wb = Workbook() ws = wb.active ws.title = "N
ew Title" for row in ws['A1:D4']: print(f"row:{row}") for cell in row: pri
nt(f"cell:{cell}") cell.value = 'Hello World' wb.save('test_ex1.xlsx')
```

실용적인 예시: 데이터 채우기

- 대규모 데이터를 처리하거나 특정 패턴으로 셀에 데이터를 입력할 때는 아래와 같은 방식을 사용
- 이 코드는 첫 번째 열의 첫 10개 행에 'Item 1', 'Item 2' ... 'Item 10'을 입력합니다.

```
for i in range(1, 11): ws.cell(row=i, column=1).value = f'Item {i}'
```

```
import openpyxl as excel book = excel.Workbook() sheet = book.active for i
in range(10): row_cell = sheet.cell(row=(i+1), column=1) row_cell.value =
str(i+1) + " 번째 데이터 저장" book.save("py_excel01.xlsx")
```

엑셀 파일 불러오기

기존 워크북 열기

- `openpyxl.load_workbook()` 함수를 사용하여 기존 워크북을 열기
- 파일 이름을 인자로 받아 해당 파일을 불러옴
- 불러온 워크북을 반환하며, 이 워크북 객체를 통해 파일의 내용을 접근하고 수정 가능

```
from openpyxl import load_workbook # 파일을 불러와서 워크북 객체 생성 wb = l
oad_workbook(filename='empty_book.xlsx') # 워크시트 이름으로 접근 ws = wb['S
heet1'] # 'Sheet1'이라는 이름의 워크시트에 접근 # D18 셀의 값을 읽음 cell_val
ue = ws['D18'].value print("The value in cell D18 is:", cell_value)
```

```
from openpyxl import load_workbook wb = load_workbook("excel_data.xlsx")
ws = wb['Sheet1'] cell_value = ws['A1'].value print(f"value :
{cell_value}")
```

for ~ in ~으로 데이터 가져오기

```
from openpyxl import load_workbook # 엑셀 파일 불러오기 wb = load_workbook
("excel_data.xlsx") ws = wb.active cell = ws["A1":"C7"] for row in ws["A
1":"C7"]: result = [] for cell in row: result.append(cell.value) print(res
ult)
```

<excel_data.xlsx 파일>

excel_data.xlsx 8.5KB

학습 과정	시간	교육방식
인프라 활용을 위한 파이썬	56	온라인
애플리케이션보안	56	온라인
시스템/네트워크 보안 기술	56	온라인
클라우드 보안 기술	56	온라인
데이터 3법과 개인정보보호	32	온라인
모듈 프로젝트	40	오프라인

- 좀 더 간단한 방식

```
from openpyxl import load_workbook # 엑셀 파일 불러오기 book = load_workboo
k("excel_data.xlsx") sheet = book.active cell = sheet["A1":"C7"] for row i
n sheet["A1":"C7"]: # 리스트 컴프리헨션으로 결과를 리스트로 저장 values = [ce
ll.value for cell in row] print(values)
```

엑셀 수식 적용 데이터 값만 가져오기

- 수식이 적용된 데이터를 가져올 때 data_only 옵션 사용

```
from openpyxl import load_workbook # 엑셀 파일 불러오기 book = load_workboo
k("excel_data2.xlsx", data_only=True) sheet = book.active cell = sheet["A
1":"E7"] for row in sheet["A1":"E7"]: # 리스트 컴프리헨션으로 결과를 리스트
로 저장 values = [cell.value for cell in row] print(values)
```

excel_data2.xlsx 9.6KB

학습 과정	시간	교육방식	수강 인원	총 시간
인프라 활용을 위한 파이썬	56	온라인	50	2800
애플리케이션보안	56	온라인	50	2800
시스템/네트워크 보안 기술	56	온라인	50	2800
클라우드 보안 기술	56	온라인	50	2800
데이터 3법과 개인정보보호	32	온라인	50	1600
모듈 프로젝트	40	오프라인	50	2000

수업 시간에 다뤘던 비교 표 (다시 언급)

```
from openpyxl import load_workbook book = load_workbook("excel_data2.xlsx", data_only=True) ws = book.active for row in ws["A1":"E7"]: result = [] for cell in row: result.append(cell.value) print(result) for row in ws["A1":"E7"]: values = [cell.value for cell in row] print(values)
```

- 기본 실습 정리

```
from openpyxl import Workbook # 새로운 엑셀 파일 생성 wb = Workbook() # 현재 활성화된 시트 선택 ws = wb.active # 모든 시트 이름 출력 print(wb.sheetnames) # 시트 이름 변경 ws.title = "New Title" # 모든 시트 이름 출력 print(wb.sheetnames) # 엑셀 저장 wb.save("01_엑셀 자동화.xlsx")
```

- 시트 관리

```
from openpyxl import Workbook # 새로운 엑셀 파일 생성 wb = Workbook() # 현재 활성화된 시트 선택 ws = wb.create_sheet("SK 쉴더스 루키즈") # 모든 시트 이름 출력 print(wb.sheetnames) # 시트 삭제 del wb["Sheet"] # 엑셀 저장 wb.save("02_엑셀 자동화.xlsx")
```

- 데이터 관리

```
from openpyxl import Workbook, load_workbook save_path = "02_엑셀 자동화.xlsx" # 기존 엑셀 파일 불러오기 wb = load_workbook(save_path) # 활성화된 시트 선택 ws = wb.active # 데이터 추가(1) # 셀 이름 지정해서 셀에 직접 데이터 추가 ws['A1'] = '날짜' ws['B1'] = '제품명' ws['C1'] = '가격' ws['D1'] = '수량' ws['E1'] = '합계' # 데이터 추가(2) ws.cell(row=2, column=1, value='2024-05-07') ws.cell(row=2, column=2, value='아이패드') ws.cell(row=2, column=3, value=780000) ws.cell(row=2, column=4, value=30) # 엑셀 수식 ws.cell(row=2, column=5, value='=C2*D2') # 데이터 추가(3) # 리스트 형태로 한 번에 데이터 추가 ws.append(['2024-05-08', '아이폰', 1380000, 3, '=C3*D3']) # 데이터 수정 ws['C2'] = 420000 ws['D2'] = 25 # 데이터 삭제 del ws['A3'] # 엑셀 저장 wb.save(save_path)
```

날짜 문자열 표현

```
from datetime import datetime now = datetime.now().strftime("%Y-%m-%d")
print(now)
```

11시 50분!!!

```
앞에서 진행했던 malwares 트래픽 사이트의 결과를 엑셀 파일로 저장하시오. 1. "설명", "링크"를 헤더로 미리 입력 2. 그 아래로 데이터들을 저장
```

```
import requests from bs4 import BeautifulSoup from openpyxl import
Workbook url = "https://www.malware-traffic-analysis.net/2023/index.html"
header_info = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36'} r
= requests.get(url, headers=header_info, verify=False) soup =
BeautifulSoup(r.text, 'html.parser') tags = soup.select("#main_content >
div.content > ul > li > a.main_menu") #워크북 wb = Workbook() ws =
wb.active ws['A1'] = "설명" ws['B1'] = "URL 링크" i = 2 for tag in tags:
ws.cell(row=i, column=1, value=tag.text) ws.cell(row=i, column=2,
value=f"https://www.malware-traffic-analysis.net/2023/{tag.get('href')}")
i = i + 1 wb.save("malwares.xlsx")
```

append를 이용한 추가 사례

```
import requests from bs4 import BeautifulSoup from openpyxl import
Workbook url = "https://www.malware-traffic-analysis.net/2023/index.html"
headers = { 'User-Agent': 'Mozilla/5.0', 'Content-Type': 'text/html;
charset=utf-8' } req = requests.get(url, headers=headers) soup =
BeautifulSoup(req.text, "lxml") # 워크북 생성 wb = Workbook() ws =
```