Python 엑셀 처리

키워드 : [openpyxl (https://openpyxl.readthedocs.io/en/stable/)](https://openpyxl.readthedocs.io/en/stable/)

이번 장은 앞서 배운 내용을 토대로 파이썬 언어를 통해 엑셀 파일을 다루는 방법에 대해서 간단히 알아보도록 하겠습니다.

우선, 파이썬 언어를 통해 엑셀 파일을 다루기 위해 우리는 이미 잘 만들어진 외부 패키지를 사용합니다. 파이썬에서 지원되는 여러 종류의 패키지가 있지만, 우리는 가장 널리 사용되는 엑셀 파일 처리를 위한 패키지인 openpyxl 패키지를 설치하고 사용합니다. 아직 패키지 설치가 이루어지지 않으신 분은 패키지 설치를 먼저 진행해주세요!

들어가기에 앞서, 우리가 사용할 openpyxl에 대해 간단히 먼저 소개하도록 하겠습니다. openpyxl는 오픈 소스(원본 소스 코드가 공개된 형태)로 제공되는 패키지입니다. 수 만 줄의 파이썬 코드로 이루어져 있으며, 약 2013년도부터 현재까지 꾸준히 업데이트되고 있는 파이썬 패키지입니다.

강의에서는 엑셀 2010 이상의 *.xlsx 확장자를 가진 파일을 읽고, 쓰는 게 가능하다고 말씀드렸습니다. 더 정확히 말하면, *.xlsx, *.xlsm, *.xltx, *.xltm 형식의 엑셀 파일을 읽고 쓰는 게 가능합니다.

어떻게 파이썬 코드로 엑셀 파일을 읽고, 쓸 수 있는지 예시 코드와 함께 알아보도록 하겠습니다. (미리 준비된 엑셀 파일이 없다면, test.xlsx 파일을 준비합니다.)

```
from openpyxl import load_workbook as load # 모듈 불러오기
```

엑셀 파일을 불러와 Workbook 객체를 만들기 위해, load_workbook 함수를 import 합니다. 이름이 조금 길기 때문에, load라는 약어를 사용합니다.

```
from openpyxl import load_workbook as load
DIR = 'test.xlsx'
wb = load(DIR) # 엑셀 파일을 읽고 쓸 수 있는 Workbook 객체를 선언합니다.
```

강의에서는 엑셀 파일의 전체 경로를 작성했습니다. 만약, 저장된 엑셀 파일의 위치가 현재 파이썬 코드와 동일한 폴더에 저장되어 있다면, 위 코드 예시와 같이 파일의 이름과 확장자만을 작성해줘도 됩니다.

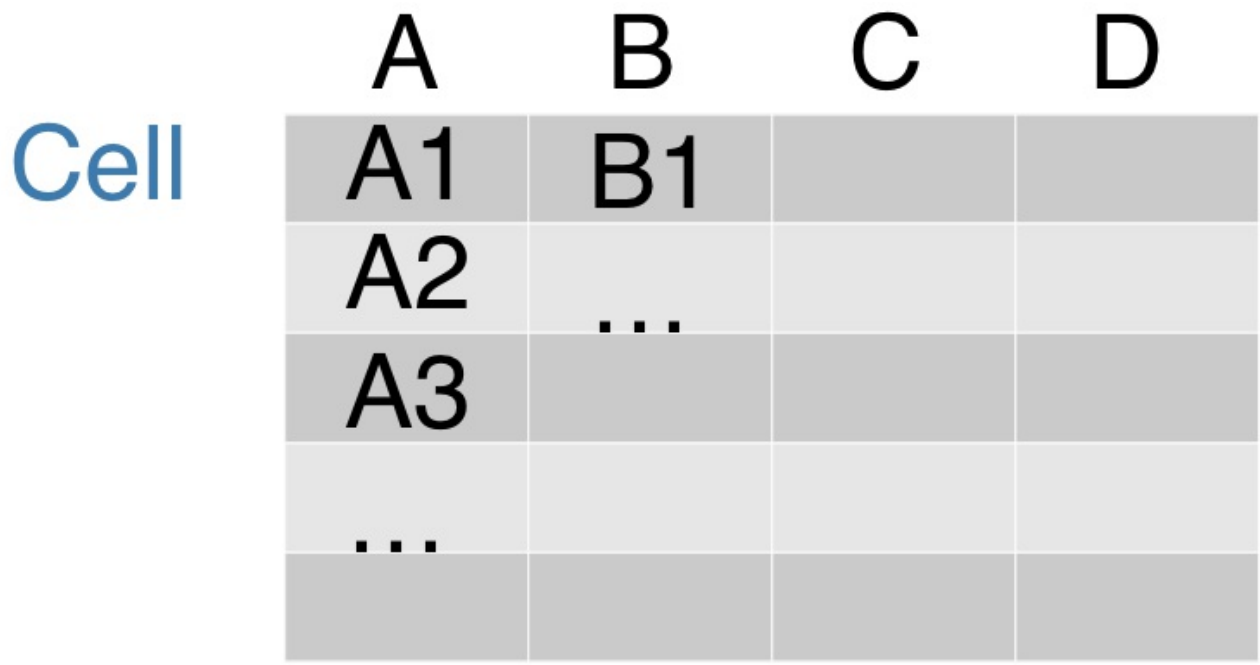
```
from openpyxl import load_workbook as load
DIR = 'test.xlsx'
wb = load(DIR)
ws = wb.create_sheet('test')
# ws = wb['test']
```

wb라는 이름으로 하나의 엑셀 파일을 나타내는 Workbook 객체를 선언했다면, 이제 하나의 시트(Sheet)를 나타낼 수 있는 Worksheet 객체를 선언해야 합니다.

새로운 시트를 만들기 위해 create_sheet 메서드를 이용합니다. 인자로 새로 생성하고자 하는 시트의 이름을 입력합니다. 기존의 존재하는 시트를 불러오는 경우엔, 딕셔너리의 키로 값을 접근하듯이, ws = wb[시트 이름] 형태로 작성해주면 됩니다.

```
from openpyxl import load_workbook as load
DIR = 'test.xlsx'
wb = load(DIR)
ws = wb.create_sheet('test')
try:
    for i in range(1, 10):
        ws[chr(65) + str(i)] = i # 1번 코드, 엑셀 파일 쓰기
    wb.save(DIR) # 엑셀 파일 저장
except Exception as e:
    print(e)
finally:
    wb.close() # 2번 코드, 외부 자원 반환
```

위 코드는 불러온 엑셀 파일에 쓰기(Write) 작업을 하는 간단한 예시입니다. 하나하나 살펴보도록 하겠습니다.



코드를 살펴보기 전에, 강의에서 살펴봤지만, 한 번 더 Cell에 대해 상기해보도록 하겠습니다. 엑셀 파일은 작성되는 모든 것이 Cell 단위로 이루어지게 됩니다. 왼쪽에서부터 A, B, C... 순서로 열(Columns)이 증가하고, 첫 번째 줄부터 1, 2, 3... 순서로 행(Rows)이 증가하게 됩니다.

위 코드 예시로 다시 돌아와서, 먼저 반복문을 통해 1 ~ 9까지의 수를 i에 할당하게 됩니다. 1번 코드의 경우 반복문이 진행되는 동안 `ws [Cell] = 값` 형태로 `test` 엑셀 파일에 1 ~ 9까지의 값을 쓰는 작업을 하기 위한 코드라는 것을 알 수 있습니다.

1번 코드에서 이해가 되지 않으실 수 있는 코드가 있습니다. 바로, `chr(65) + str(i)` 부분인데요, 이 코드가 의미하는 바를 알아보도록 하겠습니다.

제가 정리해드린 자바스크립트 키보드 이벤트에 대한 참고 자료 내용중, 아스키(ASCII) 코드라는 것에 대해 언급한 적이 있습니다. 이 아스키코드 값 중에서, 세 가지는 기억하고 계시면 유용하다고 말씀드린 부분이 있습니다. 기억하시나요..?^^

- 알파벳 소문자 a: 97
- 알파벳 대문자 A: 65
- 숫자 0: 48

chr 내장 함수는, 인자로 입력받은 값을 유니코드 문자열의 아스키코드 값으로 반환해줍니다. 그렇기 때문에, chr(65) 코드는 A와 같은 의미를 갖게 됩니다. 여기에 str(i)를 더하는 연산을 통해 문자열 결합하게 되면, 반복문이 진행되는 동안에 A1, A2, A3... A9의 값을 나타내게 됩니다.

결국, A1 ~ A9까지의 셀에 1 ~ 9를 쓰는 코드입니다. 반복문이 종료되면, wb 객체의 save 메서드를 호출하여, 쓰기 작업을 저장합니다.

앞서 예외 처리를 이야기할 때, finally 구문에 외부 자원을 사용하는 경우 예외 발생 여부와 상관없이 무조건 실행되어야 하는 코드를 작성하는 구문이라고 말씀드렸습니다.

엑셀 파일을 다루는 것도 외부 자원을 사용하는 것이기 때문에, 2번 코드와 같이 wb 객체의 close 메서드를 호출하여 같이 엑셀 파일 사용을 반환하는 코드를 작성해야 합니다.

```
from openpyxl import load_workbook as load
DIR = 'test.xlsx'
wb = load(DIR)
ws = wb.create_sheet('test')
try:
    for i in range(1, 10):
        val = ws[chr(65) + str(i)].value # 엑셀 파일 읽기
        print(val, end = ' ') # 1 2 3...9
except Exception as e:
    print(e)
finally:
    wb.close()
```

다음은, 읽기 작업입니다. 읽기 작업을 위해서, value라는 전용 속성을 이용할 수 있습니다. 마찬가지로, 셀을 통해 값에 접근할 수 있고, 가져온 값을 val이라는 임시 변수에 할당하고 출력하는 코드 예시입니다. 필요에 따라서 더 다양한 연산이 가능하겠죠? :)

이 밖에도 더 다양한 openpyxl 패키지와 관련된 메서드와 속성이 존재합니다. 셀 병합, 색 지정을 비롯한 심지어 엑셀의 함수식까지 파이썬 코드로 작성해서 적용할 수 있습니다. 패키지 개발자 분의 능력에 새삼 놀랍죠..? ^^ openpyxl과 관련된 더 다양한 사용법과 성능 이슈는 [여기 (https://openpyxl.readthedocs.io/en/stable/usage.html)](https://openpyxl.readthedocs.io/en/stable/usage.html)를 참고해주세요.

이번 장에서는 파이썬 엑셀 처리에 대해서 간단히 알아보았습니다. 우리가 흔히 사용하는 파일 형식인 엑셀 파일을 파이썬 언어로 다룰 수 있다는 것은 굉장히 큰 축복입니다. 다시 한번 말씀드리지만, 파이썬에는 이미 잘 만들어진 패키지들이 많이 존재합니다. 기초적인 파이썬 코드 구문만 이해할 수 있다면, 어떻게 활용할 수 있는지만을 고민하면 됩니다.

나중에 점차 실력이 쌓여서, 직접 이러한 코드를 구현해보고, 저를 비롯한? 많은 사람들이 유용하게 사용할 수 있는 파이썬 패키지를 만들어 주시는 날이 머지않아 찾아오길 희망합니다! :)