

<https://openpyxl.readthedocs.io/en/stable/api/openpyxl.workbook.workbook.html>

```
del workbook1['Sheet1']
```

```
import openpyxl as xl
wb = xl.load_workbook()
ws = wb.active
ws.delete_rows(firstrow, numberofrows) #for multiple row deletion
ws.delete_rows(1, ws.max_row + 1) #
for entire sheet
ws.delete_rows(rownum) #for single row
```

```
=====
remove(worksheet)[source]
Remove worksheet from this workbook.
```

```
remove_named_range(named_range)[source]
```

Remove a *named_range* from this workbook.

Deprecated: Use `del workbook.defined_names[name]`

```
remove_sheet(worksheet)[source]
```

Remove *worksheet* from this workbook.

Deprecated: Use `wb.remove(worksheet)` or `del wb[sheetname]`

```
=====
https://doitnow-man.tistory.com/159
```

python으로 엑셀 다루기
2018. 11. 22. 00:17
python, 엑셀, 엑셀 생성, 엑셀 쓰기
[python] 17. python으로 엑셀 다루기

1. 엑셀의 구조

- 1) workbook
 - 엑셀 파일을 의미
- 2) worksheet

- ```

graph LR
 WB[Workbook] --- WS1[Worksheet]
 WB --- WS2[Worksheet]
 WB --- WS3[Worksheet]
 WS1 --- C1[cell]
 WS1 --- C2[cell]
 WS1 --- C3[cell]
 WS2 --- C4[cell]
 WS2 --- C5[cell]
 WS2 --- C6[cell]
 WS3 --- C7[cell]
 WS3 --- C8[cell]
 WS3 --- C9[cell]

```

• 워크북은 하나 이상의 워크시트로 구성됨

• 현재 사용 중인 (화면에 노출된) 워크시트를 액티브 시트라고 함

**Workbook**

**Worksheet**

**Worksheet**

**Worksheet**

cell

cell

cell

cell

cell

cell

cell

cell

cell

• 엑셀 문서를 워크북이라고 함

• 하나의 워크북은 확장자가 .xlsx로 끝나는 파일로 저장됨

• 워크시트는 컬럼과 로우로 구성됨

• 컬럼과 로우가 만나는 사각형 공간을 셀(cell)이라고 함

• 각 셀은 숫자나 문자 등을 포함할 수 있음

- close()

Ex) wb.close()

### 3) sheet 열기

(1) index를 sheet이름으로하여 찾기(이게 더 좋다)

- wb['sheet1']

(2) get\_sheet\_by\_name('Sheet1') 함수 이용(비 추천 pychame에서 경고 발생)

- sheet = wb.get\_sheet\_by\_name('Sheet1')

### 4) 현재 열린 sheet를 열기

- wb.actvie

Ex) sheet = wb.actvie

## 5. OpenPyXL cell 조작 사용법

### 1) cell 접근 방법

- sheet = wb['Sheet1'] 를 통해 우선 sheet 정보를 얻는다.

(1) cell의 index 를 이용한 방법

>> A1 = sheet['A1']

>> A1.value

Sheet1 A1 value

(2) cell()함수를 이용

- 파라미터로 row와 column을 사용한다.

- 마지막 값이 존재하는 셀(row) : sheet.max\_row

- 마지막 값이 존재하는 셀(column) : sheet.max\_column

Ex)

>> A1 = sheet.cell(row = 1, column = 1)

Sheet1 A1 value

(3) 전체 예제 코드

```
import openpyxl
```

```
wb = openpyxl.load_workbook('./test.xlsx')
```

```
sheet = wb['Sheet1']
```

```
print(sheet.cell(row=1, column=1).value)
```

```
print("%s" % sheet.max_column)
```

### 2) 여러 cell 접근 하기

(1) 특정 범위

- cell\_range = sheet['A1':'C2']

(2) 특정 row

- row10 = sheet[10]

(3) 특정 row 범위

- row\_range = sheet[5:10]

(4) 특정 Column

- colC = sheet['C']

(5) 특정 Column 범위

- col\_range = sheet['C:D']

Ex) 예제 코드

- 전체 컬럼과 로우에 접근하는 코드

## 6. OpenPyXL 엑셀파일 생성

1) workbook 생성

- wb = openpyxl.Workbook() # workbook 함수를 통해 workbook 객체 생성  
# 생성시 기본적으로 sheet1이 생성된다.

2) sheet 생성

- sheet2 = wb.create\_sheet('sheet2') #마지막에 추가
- sheet3 = wb.create\_sheet('sheet3', 1) #sheet1 자리에 삽입 하여 추가

3) sheet 이름 변경

- sheet2.title = '업무자동화'

4) workbook 저장

- wb.save('저장할 파일이름.xlsx') #워크북에 저장
- wb.close() #파일 닫기

Ex) 전체 코드

```
import openpyxl

wb = openpyxl.Workbook()
sheet2 = wb.create_sheet('sheet2')
sheet3 = wb.create_sheet('sheet3', 1)
sheet2.title = '업무자동화'
wb.save('./new_test_file.xlsx')
wb.close()
```

## 7. OpenPyXL 엑셀파일에 쓰기

1) index를 이용하여 cell에 데이터 쓰기

- sheet['A1'] = 'hello' #A1에 쓰기

2) cell()함수를 이용하여 데이터 쓰기

- sheet.cell(row = 1, column = 1, value='wolrd') # A1에 값쓰기

- \* wb.save('./new\_test\_file.xlsx')를 사용하여 저장을 해야 완전히 반영된다.

8. OpenPyXL에서 지원하는 기능?

- 거의 다 지원함으로 메뉴얼을 보시기 바랍니다.

(<https://openpyxl.readthedocs.io/en/stable/>)