

# 파이썬을 활용한 업무자동화

4회차: 업무자동화로 일 편하게 하기

# 목차

4회차: 파일 및 엑셀 다루기

- 복습
- 파일 다루기
  - ▶ 텍스트 파일 읽기
  - ▶ 텍스트 파일 쓰기
- 라이브러리 소개 및 설치
- 엑셀 다루기
  - ▶ 엑셀로부터 데이터 가져오기
  - ▶ 엑셀에 데이터 쓰기
  - ▶ 다양한 활용방법

# 복습

지난 강의 때 뭐했지..?

01

## ■ 함수, 클래스

# 복습

02

지난 강의 때 뭐했지..?

## ■ 함수, 클래스

```
class Email():
    from_email = ''
    to_email = ''
    subject = ''
    contents = ''

    def send_mail(self):
        print('Send to ' + self.to_email)
```

지난 강의 때 뭐했지..?

## 함수, 클래스

```
from my_email import Email
from my_news import News
from my_excel import Excel

m_email = Email()
m_news = News()
m_excel = Excel()

news_list = m_news.find_news('fastcampus')

m_email.from_email = 'alghost.lee@gmail.com'
m_email.to_email = 'yskim@fastcampus.com'
m_email.subject = 'Dear. '

for news in news_list:
    m_email.contents = m_email.contents + news + '\n'

m_email.send_mail()

m_excel.excel_file = 'result.xlsx'
m_excel.save_to_excel(news_list)
```

# 파일 다루기

어디에 쓰일까?

04

## ■ 파일 입출력

- 텍스트 파일에 대한 읽기/쓰기를 다룸
- 어디에 쓰일까?
  - ▶ 텍스트파일의 데이터를 읽고 쓰기 위해
  - ▶ 자동화 스케줄링시 입력값/결과값으로 사용
    - 특정파일의 값을 입력값으로, 특정파일에 결과값을 씀
  - ▶ csv 파일의 데이터를 읽고 쓰기 위해

# 파일 다루기

텍스트 파일 읽기

05

## ■ 파일을 읽어보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미
- 텍스트 편집기를 이용하여 아무 내용이나 작성한 후 코드에서 읽어보자
  - ▶ 아래 작성된 텍스트를 data.txt 라는 파일명으로 저장

```
1 안녕하세요 .  
2 파일 읽기 테스트를 위한 글입니다 .  
3 여러분들은 다른 텍스트를 작성해보세요 :)  
4 후후
```

# 파일 다루기

텍스트 파일 읽기

06

## ■ 파일을 읽어보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미
- 텍스트 편집기를 이용하여 아무 내용이나 작성한 후 코드에서 읽어보자

```
datafile = open('data.txt', 'r')  
data = datafile.read()  
print(data)
```



# 파일 다루기

텍스트 파일 읽기

07

## ■ 파일을 읽어보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미
- 텍스트 편집기를 이용하여 아무 내용이나 작성한 후 코드에서 읽어보자

```
alghost:4th Alghost$ python3 007.py  
안녕하세요.  
파일 읽기 테스트를 위한 글입니다.  
여러분들은 다른 텍스트를 작성해보세요 :)  
후후
```

# 파일 다루기

텍스트 파일 읽기

08

## ■ 파일을 한줄씩 읽어보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미
- 텍스트 편집기를 이용하여 아무 내용이나 작성한 후 코드에서 읽어보자

```
datafile = open('data.txt', 'r')  
  
line = 'init'  
while line:  
    line = datafile.readline()  
    print(line)
```

# 파일 다루기

텍스트 파일 읽기

09

## ■ 파일을 한줄씩 읽어보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미
- 텍스트 편집기를 이용하여 아무 내용이나 작성한 후 코드에서 읽어보자

```
alghost:4th Alghost$ python3 009.py  
안녕하세요.
```

파일 읽기 테스트를 위한 글입니다.

여러분들은 다른 텍스트를 작성해보세요 :)

후후

# 파일 다루기

텍스트 파일 쓰기

10

## ■ 파일에 써보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미
- 사용자 입력을 받아서 입력받은 내용을 파일에 써보자
  - ▶ 입력받은 내용을 textfile.txt에 저장

```
user_input = input('User input: ')\ndatafile = open('textfile.txt', 'w')\ndatafile.write(user_input+'\n')
```

# 파일 다루기

텍스트 파일 쓰기

11

## ■ 파일에 써보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미
- 사용자 입력을 받아서 입력받은 내용을 파일에 써보자

```
alghost:4th Alghost$ python3 011.py
User input: alghost
alghost:4th Alghost$ cat textfile.txt
alghost
```

# 파일 다루기

텍스트 파일 쓰기

12

## ■ 파일에 추가로 써보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미
- 사용자 입력을 받아서 입력받은 내용을 파일에 추가로 써보자

```
user_input = input('User input: ')\ndatafile = open('textfile.txt', 'a')\ndatafile.write(user_input+'\n')
```

# 파일 다루기

텍스트 파일 쓰기

13

## ■ 파일에 추가로 써보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미
- 사용자 입력을 받아서 입력받은 내용을 파일에 추가로 써보자

```
alghost:4th Alghost$ python3 013.py
User input: 처음 입력입니다
alghost:4th Alghost$ python3 013.py
User input: 두번째
alghost:4th Alghost$ python3 013.py
User input: 세번째
alghost:4th Alghost$ cat textfile.txt
alghost
처음 입력입니다
두번째
세번째
```

# 라이브러리 소개

라이브러리란!

14

## ■ 라이브러리란?

- 특정 기능을 여러 클래스로 구성해놓은 코드 집합
- 파이썬이 기본적으로 가진 라이브러리도 많이 있음 => 기본 라이브러리
  - 강의를 진행하면서 필요할 때마다 소개/설명 할 예정
- 대부분의 라이브러리는 기능을 나열해놓은 문서가 있음 => 레퍼런스

## ■ 기본 라이브러리

- 엄청 많음.. => 수업시간에 안다름!
- 업무에 적용해보면서 나오는 질문을 포럼에 남기면?
- 관련 라이브러리에 대해 예제와 설명을 달아드리겠습니다! :D
- 모두가 알면 좋은 라이브러리인 경우 다음 수업에 반영



# 라이브러리 소개

openpyxl

15

## ■ 엑셀을 다루는 라이브러리

- 엑셀을 다루는(읽고, 쓰는) 라이브러리도 당연히 여러가지
- 우리가 사용할 라이브러리는 OpenPyXL
- 레퍼런스: <https://openpyxl.readthedocs.io>
  - ▶ 안타깝게 영어다..

# 라이브러리 설치

openpyxl

16

## ■ 아~주 쉬운 라이브러리 설치

- 파이썬에서 라이브러리 설치를 위한 프로그램을 제공: pip
- 이 프로그램을 이용해서 라이브러리를 설치할 예정
- Mac: 터미널 실행
- Windows: CMD 실행

# 라이브러리 설치

openpyxl

17

## ■ 아~주 쉬운 라이브러리 설치: Mac

- `sudo pip3 install openpyxl`
- sudo의 의미
  - ▶ 관리자 권한으로 실행하겠다!!
  - ▶ 따라서 경우에 따라 비밀번호를 요구

# 라이브러리 설치

openpyxl

18

## - 아~주 쉬운 라이브러리 설치

```
alghost:4th Alghost$ pip3 install openpyxl
Collecting openpyxl
  Downloading openpyxl-2.4.8.tar.gz (156kB)
    100% |#####| 163kB 12kB/s
Requirement already satisfied: jdcal in /usr/local/lib/python3.6/site-packages (from openpyxl)
Requirement already satisfied: et_xmlfile in /usr/local/lib/python3.6/site-packages (from openpyxl)
Building wheels for collected packages: openpyxl
  Running setup.py bdist_wheel for openpyxl ... done
  Stored in directory: /Users/Alghost/Library/Caches/pip/wheels/02/59/94/f8c9ebb9d31191e1373df8c3192a57b59e4230accf15658009
Successfully built openpyxl
Installing collected packages: openpyxl
Successfully installed openpyxl-2.4.8
alghost:4th Alghost$
```

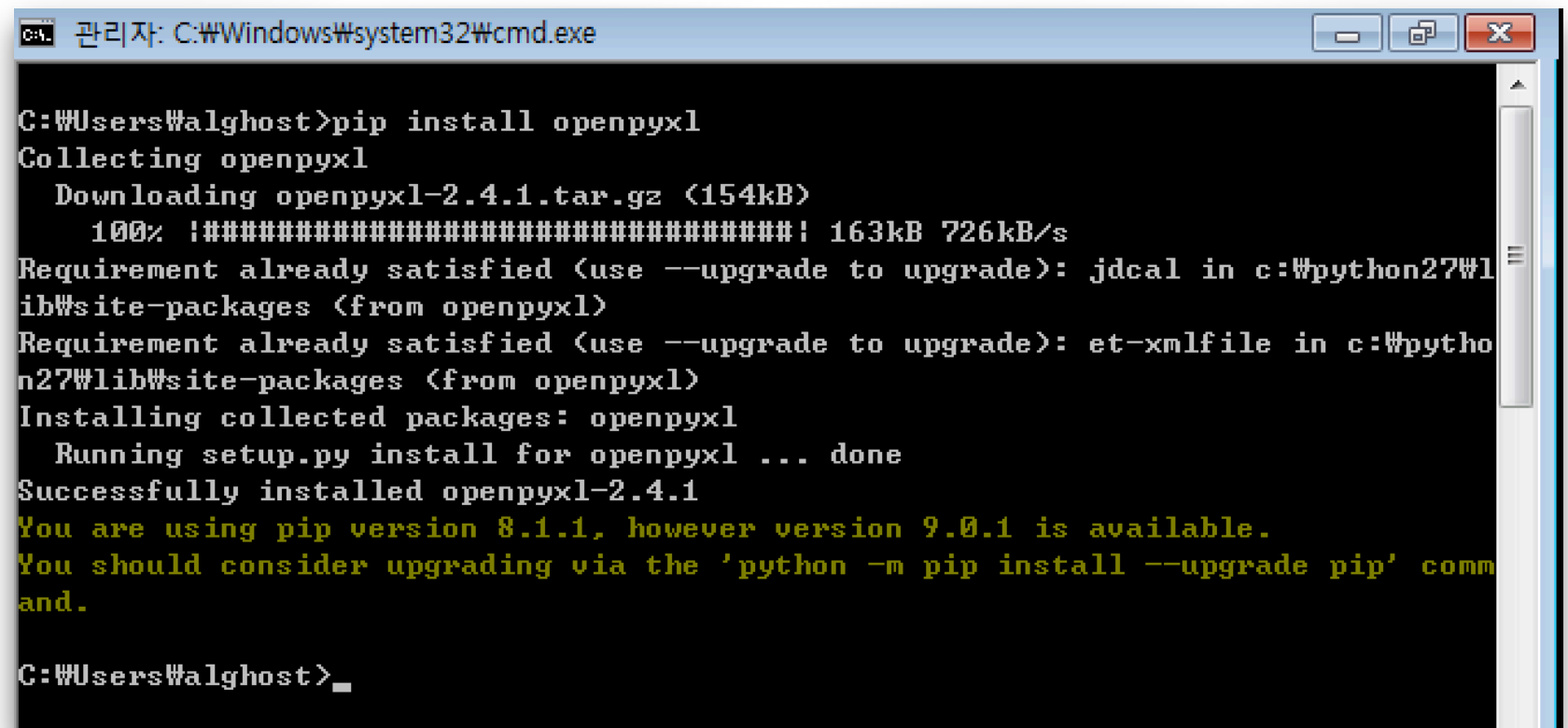
# 라이브러리 설치

19

openpyxl

## ■ 아~주 쉬운 라이브러리 설치: Windows

- pip install openpyxl
  - ▶ 윈도우 계정이 한글명인 경우 설치 경로를 지정
  - ▶ pip install --target="C:\python\_lib" openpyxl
  - ▶ 이 경우에는 환경변수 PYTHONPATH="C:\python\_lib" 필요



```
관리자: C:\Windows\system32\cmd.exe

C:\Users\alghost>pip install openpyxl
Collecting openpyxl
  Downloading openpyxl-2.4.1.tar.gz (154kB)
    100% |#####| 163kB 726kB/s
Requirement already satisfied (use --upgrade to upgrade): jdcal in c:\python27\lib\site-packages (from openpyxl)
Requirement already satisfied (use --upgrade to upgrade): et-xmlfile in c:\python27\lib\site-packages (from openpyxl)
Installing collected packages: openpyxl
  Running setup.py install for openpyxl ... done
Successfully installed openpyxl-2.4.1
You are using pip version 8.1.1, however version 9.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\alghost>
```

# 라이브러리 설치

openpyxl

20

## ■ 설치 확인 방법

- python에서 openpyxl를 사용할 수 있는지 확인
- import openpyxl로 확인 가능
- 단순 확인을 위해 터미널/CMD 에서 바로 확인해보면 됨!

```
alghost:4th Alghost$ python3
Python 3.6.1 (default, Apr  4 2017, 09:40:21)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import openpyxl
>>> █
```

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\alghost>python
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import openpyxl
>>> █
```

# 라이브러리 설치

openpyxl

20

## ■ 설치가 안되었다면..?

```
alghost:4th Alghost$ python3
Python 3.6.1 (default, Apr  4 2017, 09:40:21)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import openpyxl
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'openpyxl'
>>> █
```

# 엑셀 다루기

엑셀로 부터 데이터 가져오기

21

## ■ 사람이 엑셀로부터 데이터를 확인할 때

- 데이터가 들어있는 파일을 찾아서 연다.
- 데이터가 들어있는 시트로 이동한다.
- 데이터가 있는 위치(예: A4)에 가서 데이터를 확인한다.

## ■ 프로그램은 어떻게 할까?

- 데이터가 들어있는 파일명으로 클래스 변수 생성
- 클래스 변수에서 시트이름으로 원하는 시트를 가져옴
- 데이터가 있는 위치의 데이터를 확인

엥..? 똑같다..?



# 엑셀 다루기

엑셀로 부터 데이터 가져오기

22

## ■ 진짜인지 코드를 확인해보자

	A	B	C	D
1	name	age	position	
2	taehwa	10	developer	
3	yongseong	20	manager	
4	john	2	no	
5				
6				
7				

◀ ▶ sheet1 +

# 엑셀 다루기

엑셀로 부터 데이터 가져오기

23

## ■ 진짜인지 코드를 확인해보자

```
1 from openpyxl import load_workbook
2 wb = load_workbook('simple_data.xlsx')
3 data = wb.active
4 print(data['A1'].value)
5 print(data['A2'].value)
6 print(data['B1'].value)
7 print(data['B2'].value)
```

1. 데이터가 들어있는 파일명으로 클래스 변수 생성
2. 클래스 변수에서 시트이름으로 원하는 시트를 가져옴
  1. active => 활성화된 첫 시트를 선택
3. 데이터가 있는 위치의 데이터를 확인

# 엑셀 다루기

엑셀로 부터 데이터 가져오기

24

## ■ 진짜인지 코드를 확인해보자

```
from openpyxl import load_workbook
wb = load_workbook('simple_data.xlsx')
data = wb.active
print(data['A1'].value)
print(data['A2'].value)
print(data['B1'].value)
print(data['B2'].value)
```

```
alghost:4th Alghost$ python3 025.py
name
taehwa
age
10
```

# 엑셀 다루기

엑셀에 데이터 쓰기

25

## ■ 사람이 엑셀에 데이터를 쓸 때

- 데이터를 쓸 엑셀파일을 연다. 혹은 생성한다.
- 데이터를 쓸 시트를 연다. 혹은 생성한다.
- 원하는 위치에 데이터를 쓴후 저장한다.

## ■ 프로그램은 어떻게 할까?

- 데이터를 쓸 엑셀파일의 파일명으로 클래스 변수 생성  
혹은 파일명 없이 클래스 변수 생성
- 클래스 변수에서 시트이름으로 원하는 시트를 가져오거나 생성
- 원하는 위치에 데이터를 쓴후 저장한다.

엥..? 똑같다..?

# 엑셀 다루기

엑셀에 데이터 쓰기

26

## 진짜인지 코드를 확인해보자

```
from openpyxl import Workbook

1 wb = Workbook()
2 ws = wb.create_sheet('sheet_test')

3 ws['A1'] = 'alghost'
  ws['B1'] = 'test'

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

1. 데이터를 쓸 클래스 변수 생성 (파일명 없이)
2. 클래스 변수에서 시트이름으로 원하는 시트를 가져오거나 생성
3. 원하는 위치에 데이터를 쓴후 저장한다.

# 엑셀 다루기

엑셀에 데이터 쓰기

27

## ■ 진짜인지 코드를 확인해보자

	A	B	C	D	E
1	alghost	test			
2					
3					
4					
5					

◀ ▶

Sheet

sheet\_test

+

# 엑셀 다루기

다양한 활용방법

28

## ■ 데이터를 쓰는 또 다른 방법!

- 셀 하나씩 언제 다 넣어..
- append 활용

```
from openpyxl import Workbook

wb = Workbook()
ws = wb.create_sheet('sheet_test')

ws.append(['Numbner', 'Name'])

for i in range(10):
    ws.append([i, str(i) + ' data'])

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

# 엑셀 다루기

다양한 활용방법

29

## ■ 데이터를 쓰는 또 다른 방법!

- 셀 하나씩 언제 다 넣어..
- append 활용

```
from openpyxl import Workbook

wb = Workbook()
ws = wb.create_sheet('sheet_test')

ws.append(['Numberrer', 'Name'])

for i in range(10):
    ws.append([i, str(i) + ' data'])

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

	A	B	C	D
1	Numberrer	Name		
2	0	0 data		
3	1	1 data		
4	2	2 data		
5	3	3 data		
6	4	4 data		
7	5	5 data		
8	6	6 data		
9	7	7 data		
10	8	8 data		
11	9	9 data		
12				
13				
14				
15				
16				
17				
18				
19				



# 엑셀 다루기

다양한 활용방법

30

## ■ 데이터를 가져오는 또 다른 방법!

- 셀 하나씩 언제 다 가져와..

```
from openpyxl import load_workbook
wb = load_workbook('simple_result.xlsx')
data = wb['sheet_test']

row = data['2']
for cell in row:
    print(cell.value)

print('-'*20)

col = data['A']
for cell in col:
    print(cell.value)
```

# 엑셀 다루기

다양한 활용방법

31

## ■ 데이터를 가져오는 또 다른 방법!

- 셀 하나씩 언제 다 가져와..

```
from openpyxl import load_workbook
wb = load_workbook('data.xlsx')
data = wb.get_sheet_names()

row = data[0]
for cell in row:
    print(cell.value)

print('-')

col = data[1]
for cell in col:
    print(cell.value)
```

```
alghost:4th Alghost$ python3 032.py
0
0 data
-----
Numbrer
0
1
2
3
4
5
6
7
8
9
```

# 엑셀 다루기

다양한 활용방법

32

## ■ 데이터를 가져오는 또 다른 방법!

- 셀 하나씩 언제 다 가져와.. (2)
- 일부분만 가져오고 싶어!

## 예제

다양한 활용

## 데이터

- 셀 하나
- 일부분

```
from openpyxl import load_workbook
wb = load_workbook('simple_result.xlsx')
data = wb['sheet_test']

area = data['A1:B2']
for row in area:
    for cell in row:
        print(cell.value)

print('-'*20)

cols = data['A:B']
for col in cols:
    for cell in col:
        print(cell.value)

print('-'*20)

rows = data['1:2']
for row in rows:
    for cell in row:
        print(cell.value)
```

# 엑셀 다

다양한 활용방법

## 데이터를

- 셀 하나씩 안
- 일부분만 가

```
alghost:4th Alghost$ python3 035.py
Numbrer
Name
0
0 data
-----
Numbrer
0
1
2
3
4
5
6
7
8
9
Name
0 data
1 data
2 data
3 data
4 data
5 data
6 data
7 data
8 data
9 data
-----
Numbrer
Name
0
0 data
```

## ■ load\_workbook의 문제점

- load\_workbook: 모든 엑셀의 내용을 파이썬으로 **한번에** 가져옴
  - ▶ 엑셀 파일이 매우 큰 경우 못 가져오는 경우 발생
  - ▶ 한번에 가져오는 과정이 매우 느림

**모든 내용을 한번에 가져오지 않는 방법을 사용!**

# 엑셀 다루기

다양한 활용방법

36

## ■ 데이터를 가져오는 방법

- load\_workbook을 read\_only 모드로 수행
  - ▶ 모든 데이터를 가져오지 않음
  - ▶ 한 행씩 가져오는 함수: iter\_rows(...)

```
from openpyxl import load_workbook
wb = load_workbook('simple_result.xlsx', read_only=True)
data = wb['sheet_test']

for row in data.iter_rows():
    for cell in row:
        print(cell.value)
```

# 엑셀 다루기

다양한 활용방법

37

## ■ 데이터를 가져오는 방법

- load\_workbook

- ▶ 모든 데이터 가져오기

- ▶ 한 행씩 가져오기

```
from openpyxl import load_workbook
wb = load_workbook('data.xlsx')
data = wb['sheet1']

for row in data.iter_rows():
    for cell in row:
        print(cell.value)
```

```
alghost:4th Alghost$ python3 038.py
```

```
Number
```

```
Name
```

```
0
```

```
0 data
```

```
1
```

```
1 data
```

```
2
```

```
2 data
```

```
3
```

```
3 data
```

```
4
```

```
4 data
```

```
5
```

```
5 data
```

```
6
```

```
6 data
```

```
7
```

```
7 data
```

```
8
```

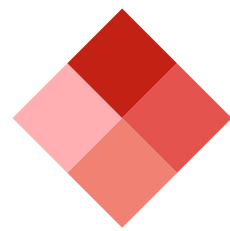
```
8 data
```

```
9
```

```
9 data
```

```
load_only=True)
```





# Good Bye

See you next time

# Appendix

유용한 함수 및 기능

## ■ merge\_cells / unmerge\_cells

- 셀을 병합/해제 함수

```
from openpyxl import Workbook
wb = Workbook()
ws = wb.active

ws.merge_cells('A1:B1')
ws.unmerge_cells('A1:B1')

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

# Appendix

유용한 함수 및 기능

## ■ Font

- 글자 스타일을 지정할 수 있는 클래스
  - underline: 'single', 'double'

```
from openpyxl.style import Font

font = Font(name='Calibri',
             size=11,
             bold=False,
             italic=False,
             underline='none',
             color='FF000000')

cell.font = font
```

# Appendix

유용한 함수 및 기능

## ■ Border

- 셀의 테두리를 지정할 수 있는 클래스

- ▶ border\_style

- thick
    - dashDot
    - dashed
    - medium
    - dotted
    - thin
    - ...

```
from openpyxl.style import Border, Side

border = Border(left=Side(border_style=None,
                           color='FF000000'),
                right=Side(border_style=None,
                           color='FF000000'),
                top=Side(border_style=None,
                         color='FF000000'),
                bottom=Side(border_style=None,
                           color='FF000000'),
                diagonal=Side(border_style=None,
                              color='FF000000'),
                diagonal_direction=0,
                outline=Side(border_style=None,
                             color='FF000000'),
                vertical=Side(border_style=None,
                              color='FF000000'),
                horizontal=Side(border_style=None,
                                color='FF000000'),

cell.border = border
```

# Appendix

유용한 함수 및 기능

## ■ Alignment

- 셀의 정렬을 다루는 클래스
  - ▶ horizontal: 'right', 'center', 'fill', 'left', ...
  - ▶ vertical: 'bottom', 'center', 'top', ...
  - ▶ text\_rotation: 0~180

```
from openpyxl.style import Alignment

align = Alignment(horizontal='general',
                  vertical='bottom',
                  text_rotation=0)

cell.alignment = align
```