

# 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
- 진짜 데이터를 가지고 해보자
  - ▶ 공공데이터를 가지고 실습

# 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 + "\n"
```

지난 강의 때 뭐했지..?

함

```
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

텍스트 파일 읽기

04

## ■ 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

05

## ■ 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

06

## 파일을 읽어보자

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

```
bash-3.2$ python fio_read.py
```

안녕하세요 .

파일 읽기 테스트를 위한 글입니다 .

여러분들은 다른 텍스트를 작성해보세요 :)

후후



# 파일 다루기

텍스트 파일 읽기

07

## ■ 파일을 한줄씩 읽어보자

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

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

line = ''
while True:
    line = datafile.readline()
    if not line:
        break

    print line
```

# 파일 다루기

텍스트 파일 읽기

08

## ■ 파일을 한줄씩 읽어보자

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

```
bash-3.2$ python fio_readline.py  
안녕하세요 .
```

파일 읽기 테스트를 위한 글입니다 .

여러분들은 다른 텍스트를 작성해보세요 :)

후 후

# 파일 다루기

텍스트 파일 쓰기

09

## ■ 파일에 써보자

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

```
user_input = raw_input('User input: ')
datafile = open('textfile.txt', 'w')

datafile.write(user_input+'\n')
```

# 파일 다루기

텍스트 파일 쓰기

10

## ■ 파일에 써보자

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

```
[bash-3.2$ python fio_write.py
User input: 안녕하세요 . 입력내용입니다 !
[bash-3.2$ cat textfile.txt
안녕하세요 . 입력내용입니다 !
```

# 파일 다루기

텍스트 파일 쓰기

11

## ■ 파일에 추가로 써보자

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

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

# 파일 다루기

텍스트 파일 쓰기

12

## ■ 파일에 추가로 써보자

- 파일 입출력은 기본적으로 텍스트 파일을 의미

```
[bash-3.2$ python fio_append.py
User input: 추가 처음 입력이네요
[bash-3.2$ python fio_append.py
User input: 앗 추가할 내용이 또 있다.
[bash-3.2$ python fio_append.py
User input: 또 또 또 또
[bash-3.2$ cat textfile.txt
안녕하세요. 입력내용입니다!
추가 처음 입력이네요
앗 추가할 내용이 또 있다.
또 또 또 또
```

# 라이브러리 소개

라이브러리란!

13

## ■ 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 페이스북에 남기면?
- 관련 라이브러리에 대해 예제와 설명을 달아드리겠습니다! :D
- 모두가 알면 좋은 라이브러리인 경우 다음 수업에 반영

# 라이브러리 소개

openpyxl

14

## ■ 엑셀을 다루는 라이브러리

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



# 라이브러리 설치

openpyxl

15

## ■ 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

16

## ■ 아~주 쉬운 라이브러리 설치: Mac

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

# 라이브러리 설치

17

openpyxl

```
[taehwui-MacBook-Pro:~ Alghost$  
[taehwui-MacBook-Pro:~ Alghost$ sudo pip install openpyxl  
Downloading/unpacking openpyxl  
  Downloading openpyxl-2.4.1.tar.gz (154kB): 154kB downloaded  
  Running setup.py egg_info for package openpyxl  
  
    no previously-included directories found matching 'openpyxl/tests'  
    no previously-included directories found matching 'openpyxl/sample'  
    no previously-included directories found matching 'openpyxl/benchmarks'  
    no previously-included directories found matching 'openpyxl/develop'  
    warning: no previously-included files matching 'test_*.py' found under directory 'openpyxl'  
    warning: no previously-included files matching 'tests/*.py' found under directory 'openpyxl'  
Requirement already satisfied (use --upgrade to upgrade): jdcal in /Library/Python/2.7/site-packages (from openpyxl)  
Requirement already satisfied (use --upgrade to upgrade): et-xmlfile in /Library/Python/2.7/site-packages (from openpyxl)  
Installing collected packages: openpyxl  
  Running setup.py install for openpyxl  
  
    no previously-included directories found matching 'openpyxl/tests'  
    no previously-included directories found matching 'openpyxl/sample'  
    no previously-included directories found matching 'openpyxl/benchmarks'  
    no previously-included directories found matching 'openpyxl/develop'  
    warning: no previously-included files matching 'test_*.py' found under directory 'openpyxl'  
    warning: no previously-included files matching 'tests/*.py' found under directory 'openpyxl'  
Successfully installed openpyxl  
Cleaning up...  
taehwui-MacBook-Pro:~ Alghost$
```

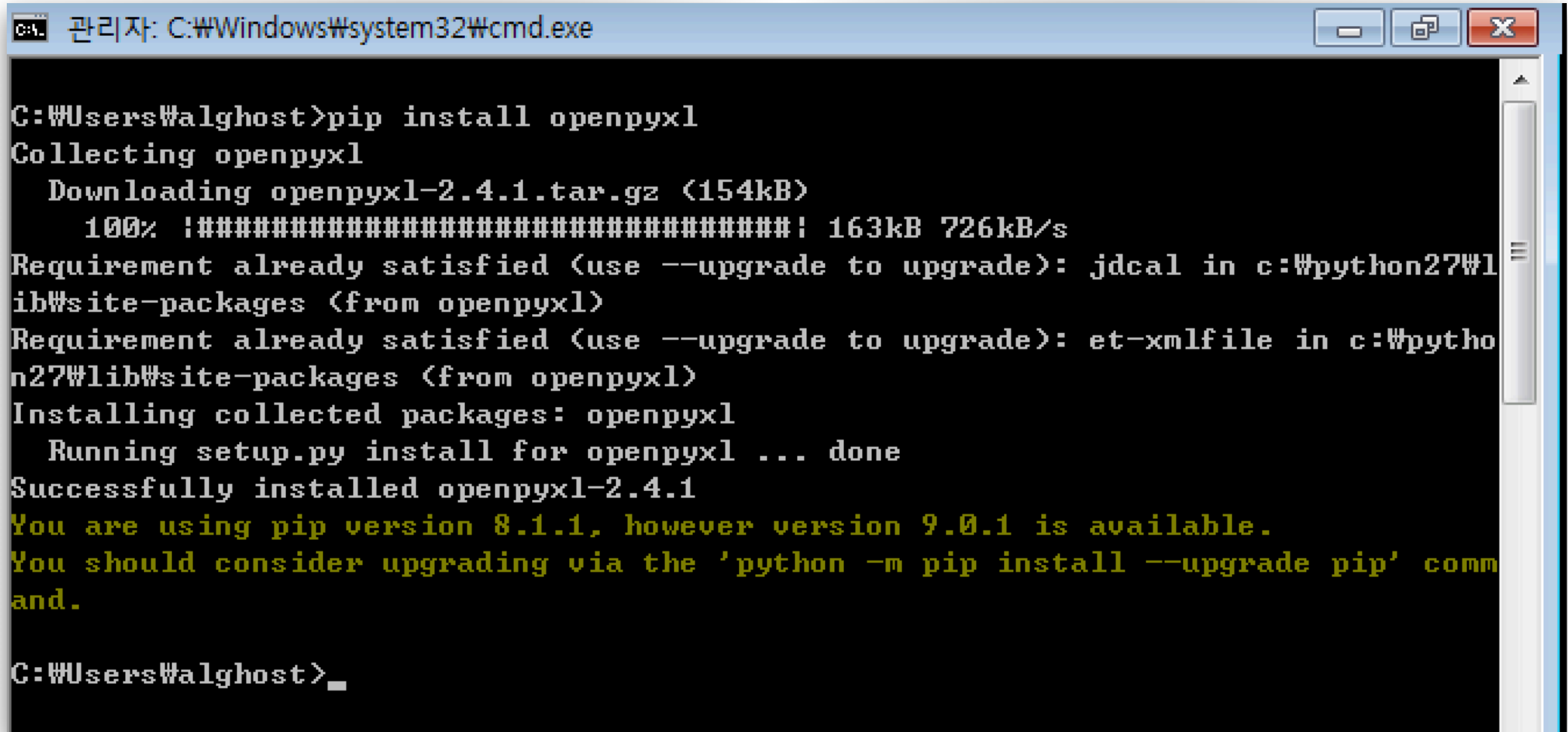
# 라이브러리 설치

18

openpyxl

## ■ 아~주 쉬운 라이브러리 설치: Windows

- pip install openpyxl



```
C:\> 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

19

## ■ 설치 확인 방법

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

```
taehwau-MacBook-Pro:~ Alghost$ python
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 26 2016, 12:10:39)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import openpyxl
>>>
```

```
C:\Users\alghost>python
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit
AMD64] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import openpyxl
>>>
```

# 라이브러리 설치

openpyxl

20

## ■ 설치가 안되었다면..?

```
taehwui-MacBook-Pro:~ Alghost$ python
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 26 2016, 12:10:39)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import openpyxl
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: 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

sheet1		
name	age	position
taehwa	10	developer
yongseong	20	manager
john	2	baby



# 엑셀 다루기

엑셀로 부터 데이터 가져오기

23

## ■ 진짜인지 코드를 확인해보자

```
from openpyxl import load_workbook
1 wb = load_workbook('simple_data.xlsx')
2 data = wb['sheet1']
3 print data['A1'].value
  print data['A2'].value
  print data['B1'].value
  print data['B2'].value
```

1. 데이터가 들어있는 파일명으로 클래스 변수 생성
2. 클래스 변수에서 시트이름으로 원하는 시트를 가져옴
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['sheet1']
print data['A1'].value
print data['A2'].value
print data['B1'].value
print data['B2'].value
```

```
taehwau-MacBook-Pro:강의용 예제 Alghost$ python real_excel.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(title='sheet_test')

3 ws['A1'] = 'alghost'
  ws['B1'] = 'wow!!!!'

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

sheet_test ▼				
alghost	wow!!!!			

# 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(title='sheet_test')

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

for i in range(20):
    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('test')

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

for i in range(20):
    ws.append([i, str(i) + ' data'])

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

Sheet ▾		sheet_test	
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		
10	10 data		
11	11 data		
12	12 data		
13	13 data		
14	14 data		
15	15 data		
16	16 data		
17	17 data		
18	18 data		
19	19 data		

# 엑셀 다루기

다양한 활용방법

30

## ■ 데이터를 가져오는 또 다른 방법!

- 셀 하나씩 언제 다 가져와..

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

monster = data[1]
for mon in monster:
    print mon.value

print "-"*20

monster = data['A']
for mon in monster:
    print mon.value
```



# 엑셀 다루기

다양한 활용방법

31

## ■ 데이터를 가져오는 또 다른 방법!

- 셀 하나씩 언제 다 가져와..

```
from openpyxl import load_workbook
```

```
taehwui-MacBook-Pro:강의용 예제 Alghost$ python real_excel.py
```

```
name
```

```
age
```

```
position
```

```
-----
```

```
name
```

```
taehwa
```

```
yongseong
```

```
john
```

```
monster = data['A']
```

```
for mon in monster:
```

```
    print mon.value
```

# 엑셀 다루기

다양한 활용방법

32

## ■ 데이터를 가져오는 또 다른 방법!

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

# 엑셀 [다양한 활용방]

다양한 활용방

33

## 데이터를

- 셀 하나씩
- 일부분만

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

```
monster = data['A1:B2']
print monster
for row in monster:
    for cell in row:
        print cell.value
```

```
monster = data['A:B']
print monster
for row in monster:
    for cell in row:
        print cell.value
```

```
monster = data[1:2]
print monster
for row in monster:
    for cell in row:
        print cell.value
```

# 엑셀 다루기

다양한 활용방법

34

## ■ 데이터를 가져오는 또 다른 방법!

```
((<Cell sheet1.A1>, <Cell sheet1.B1>), (<Cell sheet1.A2>, <Cell sheet1.B2>))
name
age
taehwa
10
((<Cell sheet1.A1>, <Cell sheet1.A2>, <Cell sheet1.A3>, <Cell sheet1.A4>), (<Cell sheet1.B1>, <Cell sheet1.B2>, <Cell sheet1.B3>, <Cell sheet1.B4>))
name
taehwa
yongseong
john
age
10
20
2
((<Cell sheet1.A1>, <Cell sheet1.B1>, <Cell sheet1.C1>), (<Cell sheet1.A2>, <Cell sheet1.B2>, <Cell sheet1.C2>))
name
age
position
taehwa
10
developer
```

```
monster = data[1:2]
print monster
for row in monster:
    for cell in row:
        print cell.value
```

## ■ 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(...)

```
1 # -*- coding: utf-8 -*-
2 from openpyxl import load_workbook
3 wb = load_workbook('sample_result.xlsx', read_only=True)
4 data = wb['sheet_test']
5
6 for row in data.iter_rows():
7     for cell in row:
8         print cell.value
```

# 예제 다루기

37

다양한 형

데이터

- load

- ▶ 모든

- ▶ 한

```
taehwui-MacBook-Pro:4th Alghost$ python 4th-11.py
```

```
Name
```

```
data
```

```
0
```

```
0 data!
```

```
1
```

```
1 data!
```

```
2
```

```
2 data!
```

```
3
```

```
3 data!
```

```
4
```

```
4 data!
```

```
1 #
```

```
5
```

```
5 data!
```

```
2 #
```

```
6
```

```
6 data!
```

```
3 v
```

```
7
```

```
7 data!
```

```
4 c
```

```
8
```

```
8 data!
```

```
5 c
```

```
9
```

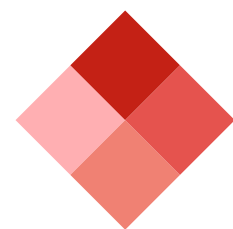
```
9 data!
```

```
6 f
```

```
7
```

```
8
```

ue)



# 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.styles 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.styles 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.styles import Alignment

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

cell.alignment = align
```