

3. 엑셀 파일

- 파이썬 기본 모듈에는 csv 모듈만 제공 ➔ *xlrd*, *xlwt* 패키지 설치

- 모든 OS에서 엑셀 파일 처리 가능
- 날짜 형식 지원

- 테스트용 통합문서 준비 (sales_2017.xlsx)

- january_2017
- february_2017
- march_2017

Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
1234	John Smith	100-0002	\$1,200.00	2017-01-01
2345	Mary Harrison	100-0003	\$1,425.00	2017-01-06
3456	Lucy Gomez	100-0004	\$1,390.00	2017-01-11
4567	Rupert Jones	100-0005	\$1,257.00	2017-01-18
5678	Susan Wallace	100-0006	\$1,725.00	2017-01-24
6789	John Smith	100-0007	\$1,995.00	2017-01-31

Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
9876	Daniel Farber	100-0008	\$1,115.00	2017-02-02
8765	Laney Stone	100-0009	\$1,367.00	2017-02-08
7654	Roger Lipney	100-0010	\$2,135.00	2017-02-15
6543	Thomas Haines	100-0011	\$1,346.00	2017-02-17
5432	Anushka Vaz	100-0012	\$1,560.00	2017-02-21
4321	Hamiet Cooper	100-0013	\$1,852.00	2017-02-28

Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
1234	John Smith	100-0014	\$1,350.00	3-4-17
8765	Tony Song	100-0015	\$1,167.00	3-8-17
2345	Mary Harrison	100-0016	\$1,789.00	3-17-17
6543	Rachel Paz	100-0017	\$2,042.00	3-22-17
3456	Lucy Gomez	100-0018	\$1,511.00	3-28-17
4321	Susan Wallace	100-0019	\$2,280.00	3-30-17

3-1. 통합 문서 정보

- CSV파일과 달리 텍스트 편집기로 열어 볼 수 없음
- 여러 개의 워크시트로 구성
- xlrd모듈의 open_workbook() 함수
 - 워크북의 sheets() 함수

```
workbook = open_workbook(input_file)
print('Number of worksheets:', workbook.nworksheets)
for worksheet in workbook.worksheets():
    print("Worksheet name:", worksheet.name, "\tRows:", \
          worksheet.nrows, "\tColumns:", worksheet.ncols)
```

```
Number of worksheets: 3
Worksheet name: january_2017    Rows: 7      Columns: 5
Worksheet name: february_2017   Rows: 7      Columns: 5
Worksheet name: march_2017      Rows: 7      Columns: 5
```

3-2. 엑셀 파일 읽기, 쓰기(xlra, xlwt)

- 단일 워크시트에 파일 읽기, 쓰기
- sheet_by_name() → 워크시트에 연결

```
output_workbook = Workbook()
output_worksheet = output_workbook.add_sheet('jan_2013_output')

with open_workbook(input_file) as workbook:
    worksheet = workbook.sheet_by_name('january_2013')
    for row_index in range(worksheet.nrows):
        for column_index in range(worksheet.ncols):
            output_worksheet.write(row_index, column_index, \
                                   worksheet.cell_value(row_index, column_index))
```

Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
1234	John Smith	100-0002	1200	42736
2345	Mary Harrison	100-0003	1425	42741

날짜형식 틀림

3-2. 엑셀 파일 읽기, 쓰기(xlra, xlwt)

- 날짜 형식 지정

```
with open_workbook(input_file) as workbook:
    worksheet = workbook.sheet_by_name('january_2017')
    for row_index in range(worksheet.nrows):
        row_list_output = []
        for col_index in range(worksheet.ncols):
            if worksheet.cell_type(row_index, col_index) == 3: → 날짜형식
                date_cell = xldate_as_tuple(worksheet.cell_value(row_index, col_index),\
                                           workbook.datemode) → (2017, 1, 1, 0, 0, 0)
                date_cell = date(*date_cell[0:3]).strftime('%m/%d/%Y')
                row_list_output.append(date_cell)
                output_worksheet.write(row_index, col_index, date_cell)
            else:
                non_date_cell = worksheet.cell_value(row_index, col_index)
                row_list_output.append(non_date_cell)
                output_worksheet.write(row_index, col_index, non_date_cell)
```

3-3. 특정 조건 충족하는 행 필터링(xlra, xlwt)

- Sale Amount 열의 데이터 값이 \$1,400.00 보다 큰 행 선택

```
worksheet = workbook.sheet_by_name('january_2017')
data = [] ➔ 출력파일에 쓰고자 하는 입력 파일의 모든 행을 저장
header = worksheet.row_values(0)
data.append(header)
for row_index in range(1, worksheet.nrows):
    row_list = []
    sale_amount = worksheet.cell_value(row_index, sale_amount_column_index)
    if sale_amount > 1400.0:
        for column_index in range(worksheet.ncols):
            cell_value = worksheet.cell_value(row_index, column_index)
            cell_type = worksheet.cell_type(row_index, column_index)
            if cell_type == 3:
                date_cell = xldate_as_tuple(cell_value, workbook.datemode)
                date_cell = date(*date_cell[0:3]).strftime('%m/%d/%Y')
                row_list.append(date_cell)
            else:
                row_list.append(cell_value)
    if row_list:
        data.append(row_list)
```

3-4. 특정 집합 값을 충족하는 행 필터링(xlra, xlwt)

- Purchase Date열의 특정 집합 (2017-01-24, 2017-01-31)에 포함되는 데이터 행 선택

```
important_dates = ['01/24/2017', '01/31/2017']
```

```
purchase_date_column_index = 4
with open_workbook(input_file) as workbook:
    worksheet = workbook.sheet_by_name('january_2017')
    data = []
    header = worksheet.row_values(0)
    data.append(header)
    for row_index in range(1, worksheet.nrows):
        purchase_datetime = xldate_as_tuple(\
            worksheet.cell_value(row_index, purchase_date_column_index),\
            workbook.datemode)
        purchase_date = date(*purchase_datetime[0:3]).strftime('%m/%d/%Y')
        row_list = []
        if purchase_date in important_dates:
            for column_index in range(worksheet.ncols):
```

3-5. 패턴을 이용한 필터링(xlra, xlwt)

- Customer Name 열이 'J'로 시작하는 행 필터링

```
pattern = re.compile(r'(?P<my_pattern>^J.*)') → ^J.*
```

```
for row_index in range(1, worksheet.nrows):
    row_list = []
    if pattern.search(worksheet.cell_value(row_index, customer_name_column_index)):
        for column_index in range(worksheet.ncols):
            cell_value = worksheet.cell_value(row_index, column_index)
            cell_type = worksheet.cell_type(row_index, column_index)
            if cell_type == 3:
                date_cell = xldate_as_tuple(cell_value, workbook.datemode)
                date_cell = date(*date_cell[0:3]).strftime('%m/%d/%Y')
                row_list.append(date_cell)
            else:
                row_list.append(cell_value)
```

3-6. 열 인덱스를 사용한 특정 열 선택 (xlra, xlwt)

- 여러 파일의 열의 위치가 일정할 때

```
my_columns = [1, 4]

with open_workbook(input_file) as workbook:
    worksheet = workbook.sheet_by_name('january_2017')
    data = []
    for row_index in range(worksheet.nrows):
        row_list = []
        for column_index in my_columns:
            cell_value = worksheet.cell_value(row_index, column_index)
            cell_type = worksheet.cell_type(row_index, column_index)
            if cell_type == 3:
                date_cell = xldate_as_tuple(cell_value, workbook.datemode)
                date_cell = date(*date_cell[0:3]).strftime('%m/%d/%Y')
                row_list.append(date_cell)
            else:
                row_list.append(cell_value)
        data.append(row_list)
```


3-7. 열 헤더를 사용하여 특정 열 선택 (xlra, xlwt)

- 여러 파일에 있는 열의 위치가 일정하지 않지만, 열의 이름이 같을 때

```
my_columns = ['Customer ID', 'Purchase Date']

with open_workbook(input_file) as workbook:
    worksheet = workbook.sheet_by_name('january_2017')
    data = [my_columns]
    header_list = worksheet.row_values(0)
    header_index_list = []
    for header_index in range(len(header_list)):
        if header_list[header_index] in my_columns:
            header_index_list.append(header_index)
    for row_index in range(1, worksheet.nrows):
        row_list = []
        for column_index in header_index_list:
            cell_value = worksheet.cell_value(row_index, column_index)
            cell_type = worksheet.cell_type(row_index, column_index)
            if cell_type == 3:
                date_cell = xldate_as_tuple(cell_value, workbook.datemode)
                date_cell = date(*date_cell[0:3]).strftime('%m/%d/%Y')
            row_list.append(date_cell)
```

3-8. 모든 워크시트에서 특정 행 필터링 (xlra, xlwt)

- 모든 워크시트에서 Sale Amount 열의 값이 \$2,000.00 이상인 모든 행을 필터링

```
output_workbook = Workbook()
output_worksheet = output_workbook.add_sheet('filtered_rows_all_worksheets')
```

```
sales_column_index = 3
threshold = 2000.0
```

```
for worksheet in workbook.sheets():
    if first_worksheet:
        header_row = worksheet.row_values(0)
        data.append(header_row)
        first_worksheet = False
    for row_index in range(1, worksheet.nrows):
        row_list = []
        sale_amount = worksheet.cell_value(row_index, sales_column_index)
        sale_amount = float(str(sale_amount).replace('$', '').replace(',', ''))
        if sale_amount > threshold:
            ...
```

3-9. 모든 워크시트에서 특정 열 선택 (xlra, xlwt)

- 모든 워크시트를 읽고, 불필요한 열 필터링 or 필요한 열 선택
- Customer Name or Sale Amount 열만 선택

```
output_worksheet = output_workbook.add_sheet('selected_columns_all_worksheets')
```

```
my_columns = ['Customer Name', 'Sale Amount']
```

```
data = [my_columns]
index_of_cols_to_keep = []
for worksheet in workbook.sheets():
    if first_worksheet:
        header = worksheet.row_values(0)
        for column_index in range(len(header)):
            if header[column_index] in my_columns:
                index_of_cols_to_keep.append(column_index)
    first_worksheet = False
for row_index in range(1, worksheet.nrows):
    row_list = []
    for column_index in index_of_cols_to_keep:
```

3-10. 워크시트 집합에서 특정 행 필터링 (xlra, xlwt)

- sheet_by_index() or sheet_by_name() 함수 이용
- 두개의 워크시트에서 Sale Amount 열의 값이 \$1,900.00 보다 큰 행 필터링

```
my_sheets = [0,1]
threshold = 1900.0
sales_column_index = 3
```

```
for sheet_index in range(workbook.nsheets):
    if sheet_index in my_sheets:
        worksheet = workbook.sheet_by_index(sheet_index)
        if first_worksheet:
            header_row = worksheet.row_values(0)
            data.append(header_row)
            first_worksheet = False
        for row_index in range(1,worksheet.nrows):
            row_list = []
            sale_amount = worksheet.cell_value(row_index, sales_column_index)
            if sale_amount > threshold:
```

3-10. 통합 문서의 개수 및 각 통합 문서의 행과 열 개수(xlra, xlwt)

- 파이썬 내장 모듈 glob() 사용
 - 여러 개의 통합 문서 준비

```
workbook_counter = 0
for input_file in glob.glob(os.path.join(input_directory, '*.xls*')):
    workbook = open_workbook(input_file)
    print('Workbook: {}'.format(os.path.basename(input_file)))
    print('Number of worksheets: {}'.format(workbook.nsheets))
    for worksheet in workbook.sheets():
        print('Worksheet name:', worksheet.name, '\tRows:', \
              worksheet.nrows, '\tColumns:', worksheet.ncols)
```

```
Workbook: sales_2013.xlsx
Number of worksheets: 3
Worksheet name: january_2013    Rows: 7      Columns: 5
Worksheet name: february_2013   Rows: 7      Columns: 5
Worksheet name: march_2013      Rows: 7      Columns: 5
Workbook: sales_2014.xlsx
Number of worksheets: 3
Worksheet name: january_2014     Rows: 7      Columns: 5
Worksheet name: february_2014   Rows: 7      Columns: 5
Worksheet name: march_2014      Rows: 7      Columns: 5
...
```

3-11. 여러 개의 통합 문서 합치기 (xlra, xlwt)

```
data = []
first_worksheet = True
for input_file in glob.glob(os.path.join(input_folder, '*.xls*')):
    print(os.path.basename(input_file))
    with open_workbook(input_file) as workbook:
        for worksheet in workbook.sheets():
            if first_worksheet:
                header_row = worksheet.row_values(0)
                data.append(header_row)
                first_worksheet = False
            for row_index in range(1, worksheet.nrows):
                row_list = []
                for column_index in range(worksheet.ncols):
                    cell_value = worksheet.cell_value(row_index, column_index)
                    cell_type = worksheet.cell_type(row_index, column_index)
```

3-12. 통합 문서 및 워크시트별 합계 및 평균 계산 (xlra, xlwt)

```
output_workbook = Workbook()
output_worksheet = output_workbook.add_sheet('sums_and_averages')

all_data = []
sales_column_index = 3

header = ['workbook', 'worksheet', 'worksheet_total', 'worksheet_average', \
          'workbook_total', 'workbook_average']
all_data.append(header)
```

```
for row_index in range(1, worksheet.nrows):
    try:
        total_sales +=
float(str(worksheet.cell_value(row_index, sales_column_index)).strip('$').replace(',', ''))
        number_of_sales += 1.
    except:
        total_sales += 0.
        number_of_sales += 0.
average_sales = '%.2f' % (total_sales / number_of_sales)
```

4-1. 엑셀 파일 읽기, 쓰기 (Pandas)

- Pandas의 엑셀 파일 읽고 쓰는 함수 사용

```
data_frame = pd.read_excel(input_file, sheetname='january_2017')

writer = pd.ExcelWriter(output_file)
data_frame.to_excel(writer, sheet_name='jan_17_output', index=False)
writer.save()
```

Customer ID	Customer Name	Invoice Number	Sale Amount	Purchase Date
1234	John Smith	100-0002	1200	2017-01-01 00:00:00
2345	Mary Harrison	100-0003	1425	2017-01-06 00:00:00
3456	Lucy Gomez	100-0004	1390	2017-01-11 00:00:00
4567	Rupert Jones	100-0005	1257	2017-01-18 00:00:00
5678	Jenny Walters	100-0006	1725	2017-01-24 00:00:00
6789	Samantha Donaldson	100-0007	1995	2017-01-31 00:00:00

4-2. 특정 조건 충족하는 행 필터링 (Pandas)

- Sale Amount 열의 데이터 값이 \$1,400.00 보다 큰 행 선택
- 대괄호([])를 사용하여 행 필터링
 - 여러 조건 필요 시 &, | 사용

```
data_frame = pd.read_excel(input_file, 'january_2017', index_col=None)
data_frame_value_meets_condition = \
    data_frame[data_frame['Sale Amount'].astype(float) > 1400.0]

writer = pd.ExcelWriter(output_file)
data_frame_value_meets_condition.to_excel(writer, sheet_name='jan_17_output', index=False)
writer.save()
```

4-3. 특정 집합 값을 충족하는 행 필터링 (Pandas)

- Purchase Date열의 특정 집합 (2017-01-24, 2017-01-31)에 포함되는 데이터 행 선택
- isin()
 - 특정 값이 리스트에 있는지 검사

```
data_frame = pd.read_excel(input_file, 'january_2017', index_col=None)

important_dates = ['01/24/2017', '01/31/2017']
data_frame_value_in_set = data_frame[data_frame['Purchase Date'].isin(important_dates)]

writer = pd.ExcelWriter(output_file)
data_frame_value_in_set.to_excel(writer, sheet_name='jan_17_output', index=False)
```

4-4. 패턴을 이용한 필터링 (Pandas)

- Customer Name 열이 'J'로 시작하는 행 필터링
- startswith(), endswith(), match(), search()

```
data_frame = pd.read_excel(input_file, 'january_2017', index_col=None)

data_frame_value_matches_pattern = \
    data_frame[data_frame['Customer Name'].str.startswith("J")]

writer = pd.ExcelWriter(output_file)
data_frame_value_matches_pattern.to_excel(writer, sheet_name='jan_17_output', index=False)
writer.save()
```

4-5. 열 인덱스를 사용한 특정 열 선택 (Pandas)

- 데이터프레임 사용
 - [] 안에 선택할 열의 인덱스 값 or 열 헤더 나열
- iloc() 함수 사용
 - 특정 행과 열을 동시에 선택 가능
 - 첫번째 인자에 ':'를 넣어야 함 → 사용하지 않으면 iloc() 해당 인덱스 값의 행을 필터링 함

```
data_frame = pd.read_excel(input_file, 'january_2017', index_col=None)

data_frame_column_by_index = data_frame.iloc[:, [1, 4]]

writer = pd.ExcelWriter(output_file)
data_frame_column_by_index.to_excel(writer, sheet_name='jan_17_output', index=False)
writer.save()
```

4-6. 열 헤더를 사용하여 특정 열 선택 (Pandas)

- 데이터프레임 뒤에 [] 열 헤더를 사용
- loc() 함수
 - 열 선택할 경우 열 헤더 리스트 앞에 ':' 사용

```
data_frame = pd.read_excel(input_file, 'january_2017', index_col=None)

data_frame_column_by_name = data_frame.loc[:, ['Customer ID', 'Purchase Date']]

writer = pd.ExcelWriter(output_file)
data_frame_column_by_name.to_excel(writer, sheet_name='jan_17_output', index=False)
writer.save()
```

4-7. 모든 워크시트에서 특정 행 필터링 (Pandas)

- 모든 워크시트에서 Sale Amount 열의 값이 \$2,000.00 이상인 모든 행을 필터링
- read_excel()
 - sheetname = None → 통합 문서의 모든 워크시트 읽음
 - 모든 워크시트는 데이터프레임으로 구성된 Dictionary 자료형으로 읽음

```
data_frame = pd.read_excel(input_file, sheetname=None, index_col=None)

row_output = []
for worksheet_name, data in data_frame.items():
    row_output.append(\
        data[data['Sale Amount'].replace('$', '').replace(',', ' ').astype(float) > 2000.0])
filtered_rows = pd.concat(row_output, axis=0, ignore_index=True)

writer = pd.ExcelWriter(output_file)
filtered_rows.to_excel(writer, sheet_name='sale_amount_gt2000', index=False)
writer.save()
```

4-8. 모든 워크시트에서 특정 열 선택 (Pandas)

- read_excel()
 - 모든 워크시트를 Dictionary 자료형으로 읽음
- Customer Name or Sale Amount 열만 선택

```
data_frame = pd.read_excel(input_file, sheetname=None, index_col=None)

column_output = []
for worksheet_name, data in data_frame.items():
    column_output.append(data.loc[:, ['Customer Name', 'Sale Amount']])
selected_columns = pd.concat(column_output, axis=0, ignore_index=True)

writer = pd.ExcelWriter(output_file)
selected_columns.to_excel(writer, sheet_name='selected_columns_all_worksheets', index=False)
writer.save()
```

4-9. 워크시트 집합에서 특정 행 필터링 (Pandas)

- read_excel()
 - 워크시트의 인덱스 번호나 이름을 지정

```
my_sheets = [0,1]
threshold = 1900.0

data_frame = pd.read_excel(input_file, sheetname=my_sheets, index_col=None)

row_list = []
for worksheet_name, data in data_frame.items():
    row_list.append(data[data['Sale Amount'].replace('$', '').replace(',', '').\
                        .astype(float) > threshold])
filtered_rows = pd.concat(row_list, axis=0, ignore_index=True)

writer = pd.ExcelWriter(output_file)
filtered_rows.to_excel(writer, sheet_name='set_of_worksheets', index=False)
writer.save()
```


4-10. 여러 개의 통합 문서 합치기 (Pandas)

- concat()
 - 데이터프레임 결합
 - axis=0(수직 결합), axis=1(수평 결합)
- merge()
 - 열을 기반으로 데이터프레임을 조인 할 때 사용

```
all_workbooks = glob.glob(os.path.join(input_path, '*.xls*'))
data_frames = []
for workbook in all_workbooks:
    all_worksheets = pd.read_excel(workbook, sheetname=None, index_col=None)
    for worksheet_name, data in all_worksheets.items():
        data_frames.append(data)
all_data_concatenated = pd.concat(data_frames, axis=0, ignore_index=True)

writer = pd.ExcelWriter(output_file)
all_data_concatenated.to_excel(writer, sheet_name='all_data_all_workbooks', index=False)
```

4-11. 통합 문서 및 워크시트별 합계 및 평균 계산 (Pandas)

- 각 워크시트에 대한 통계를 계산 → 데이터프레임으로 결합
- 통합 문서 차원의 통계를 계산 → 데이터프레임으로 결합

→ *Left Join*

```
for worksheet_name, data in all_worksheets.items():
    total_sales = pd.DataFrame([float(str(value).strip('$').replace(',','')) for value
in data.ix[:, 'Sale Amount']]).sum()
    number_of_sales = len(data.loc[:, 'Sale Amount'])
    average_sales = pd.DataFrame(total_sales / number_of_sales)

    workbook_total_sales.append(total_sales)
    workbook_number_of_sales.append(number_of_sales)

    data = {'workbook': os.path.basename(workbook),...}

...
worksheets_data_frame = pd.concat(worksheet_data_frames, axis=0, ignore_index=True)

workbook_total = pd.DataFrame(workbook_total_sales).sum()
workbook_total_number_of_sales = pd.DataFrame(workbook_number_of_sales).sum()
workbook_average = pd.DataFrame(workbook_total / workbook_total_number_of_sales)
```