단기 집중교육

4일차 실습

Computational Thinking

연세대학교
YONSEI MIRAE CAMPUS

pandas

YONSEI MIRAE CAMPUS

Series 데이터

◆ Series 데이터

: 라벨이 있는 1차원 데이터

(numpy 배열과 차이점: 요소 데이터 타입이 달라도 가능)

> 생성: s= pd.Series(data[, index = index_data])

```
import pandas as pd
s1 = pd.Series([10,20,30,40,50]) #리스트로 series데이터 생성
```

```
s1
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

index values

Series 데이터

1. Series 데이터 index, value

   I. S.index

   II. S.values

```
s1
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

```
s1.index
```

```
RangeIndex(start=0, stop=5, step=1)
```

```
s1.values
```

```
array([10, 20, 30, 40, 50], dtype=int64)
```

Series 데이터

1. Series 데이터 생성

   I. index_data = ['2020-02-27','2020-02-28','2020-02-29','2020-03-01']

   II. data = [3500, 3579, np.nan, 3782]

   >위의 두 데이터를 이용하여 series 데이터를 생성하시오.

Series 데이터

1. Series 데이터 생성

   I. index_data = ['2020-02-27','2020-02-28','2020-02-29','2020-03-01']

   II. data = [3500, 3579, np.nan, 3782]

```
import numpy as np

index_data = ['2020-02-27','2020-02-28','2020-02-29','2020-03-01']
data = [3500, 3579, np.nan, 3782]      # 데이터 지정

s2 = pd.Series(data, index=index_data) # Series 데이터 생성
s2
```

```
2020-02-27    3500.0
2020-02-28    3579.0
2020-02-29       NaN
2020-03-01    3782.0
dtype: float64
```

Dataframe 데이터

◆ Dataframe

: 행과 열이 있는 표 형식의 데이터

> df= pd.DataFrame(data, [, index = index_data, columns=columns_data])

(index, column 자동 생성)

```
import pandas as pd

data = [[1,2,3], [4,5,6],[7,8,9]]
df = pd.DataFrame(data)
df
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

Column name

Index label

values

Dataframe 데이터

◆ Dataframe 데이터 생성 문제

- Data: numpy 배열 데이터 생성(1에서 12까지, (4,3) 배열 데이터)
- index_data = pd.date_range('2020-01-11', periods=4)
  (pd.date_range: 2020-01-11부터 4일 동안의 날짜 데이터 생성)
- columns_data = ['A', 'B', 'C']

	A	B	C
2020-01-11	1	2	3
2020-01-12	4	5	6
2020-01-13	7	8	9
2020-01-14	10	11	12

Dataframe 데이터


```
import numpy as np
import pandas as pd

data = np.array([[1, 2, 3], [4, 5, 6], [7, 8 ,9], [10, 11, 12]])
# data 생성
index_data = pd.date_range('2020-01-11', periods=4)
# index를 위한 날짜 데이터
columns_data = ['A', 'B', 'C']
# columns를 위한 리스트 데이터

pd.DataFrame(data, index=index_data, columns=columns_data)
# DataFrame 데이터 생성
```

	A	B	C
2020-01-11	1	2	3
2020-01-12	4	5	6
2020-01-13	7	8	9
2020-01-14	10	11	12

Dataframe 데이터


```
df= pd.DataFrame(data, index=index_data, columns=columns_data)
```

```
df.index
```

```
DatetimeIndex(['2020-01-11', '2020-01-12', '2020-01-13', '2020-01-14'],
dtype='datetime64[ns]', freq='D')
```

```
df.columns
```

```
Index(['A', 'B', 'C'], dtype='object')
```

```
df.values
```

```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])
```

Dataframe 데이터


```
dict_data = {'A': [10, 20, 30, 40, 50, 60],
             'B': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6],
             'C': [100, 200, 300, 400, 500, 600]} # 딕셔너리 데이터

df2 = pd.DataFrame(dict_data)
# 딕셔너리 데이터로부터 DataFrame 데이터 생성
df2
```

	A	B	C
0	10	0.1	100
1	20	0.2	200
2	30	0.3	300
3	40	0.4	400
4	50	0.5	500
5	60	0.6	600

Dataframe 데이터

1. 행 데이터 재배열

```
df2.reindex([4, 2, 5, 3, 1])
```

	A	B	C
4	50	0.5	500
2	30	0.3	300
5	60	0.6	600
3	40	0.4	400
1	20	0.2	200

2. 열 데이터 재배열

```
df2.reindex(columns=['B', 'C', 'A'])
```

	B	C	A
0	0.1	100	10
1	0.2	200	20
2	0.3	300	30
3	0.4	400	40
4	0.5	500	50
5	0.6	600	60

csv 파일 읽고 쓰기

◆ df = pd.read_csv(file_name[, encoding=인코딩 방식
index_col=열 이름 혹은 숫자,
header = 숫자 혹은 none
sep= 구분자(기본: ' , ')
names = 열 이름 리스트])

- Index_col: 특정 열을 dataframe index로 지정
- Header: 특정 행을 dataframe columns로 지정
- Sep: 데이터 필드 사이 구분하는 구분자 지정

csv 파일 읽고 쓰기


```
import pandas as pd

#  CSV 파일 경로
folder = 'C:/myPyExcel/data/ch05/' # 폴더 경로를 지정
csv_file = folder + 'korea_rain1.csv' # 파일 경로를 지정

# CSV 파일을 읽어와서 DataFrame 데이터 생성
df = pd.read_csv(csv_file, encoding = "utf-8")
df
```

	연도	봄	여름	가을	겨울
0	2014	215.9	599.8	293.1	76.9
1	2015	223.2	387.1	247.7	109.1
2	2016	312.8	446.2	381.6	108.1
3	2017	118.6	609.7	172.5	75.6
4	2018	368.1	586.5	351.2	66.5

csv 파일 읽고 쓰기

◆ 특정 열을 index로 저장

```
df = pd.read_csv(csv_file, index_col="연도")
df
```

연도	봄	여름	가을	겨울
2014	215.9	599.8	293.1	76.9
2015	223.2	387.1	247.7	109.1
2016	312.8	446.2	381.6	108.1
2017	118.6	609.7	172.5	75.6
2018	368.1	586.5	351.2	66.5

◆ 공백 구분자로 구분된 데이터

```
# 텍스트 파일 경로
folder = 'C:/myPyExcel/data/ch05/'
txt_file = folder + 'korea_rain1_space.txt'

# 공백 구분자가 있는 텍스트 데이터 파일을 읽어서 DataFrame 데이터 생성
df = pd.read_csv(txt_file, sep=" ", encoding="utf-8")
df
```

	연도	봄	여름	가을	겨울
0	2014	215.9	599.8	293.1	76.9
1	2015	223.2	387.1	247.7	109.1
2	2016	312.8	446.2	381.6	108.1
3	2017	118.6	609.7	172.5	75.6
4	2018	368.1	586.5	351.2	66.5

csv 파일 읽고 쓰기

1. Csv파일을 읽어 dataframe을 생성하라
   (header와 names 옵션 지정)

	A	B	C	D	E
1	2014	215.9	599.8	293.1	76.9
2	2015	223.2	387.1	247.7	109.1
3	2016	312.8	446.2	381.6	108.1
4	2017	118.6	609.7	172.5	75.6
5	2018	368.1	586.5	351.2	66.5

<원본 데이터>

< 출력 결과 >

	Year	Spring	Summer	Fall	Winter
0	2014	215.9	599.8	293.1	76.9
1	2015	223.2	387.1	247.7	109.1
2	2016	312.8	446.2	381.6	108.1
3	2017	118.6	609.7	172.5	75.6
4	2018	368.1	586.5	351.2	66.5

csv 파일 읽고 쓰기


```
# 텍스트 파일 경로
folder = 'C:/myPyExcel/data/ch05/'
txt_file = folder + 'korea_rain2.csv' # 열 이름이 없는 CSV 파일

# CSV 파일을 읽어 DataFrame 데이터 생성(names 옵션 지정)
names_list = ["Year", "Spring", "Summer", "Fall", "Winter"]
df2 = pd.read_csv(txt_file, names=names_list)
# df2 = pd.read_csv(txt_file, header=None, names=names_list) 도 동일
df2
```

	Year	Spring	Summer	Fall	Winter
0	2014	215.9	599.8	293.1	76.9
1	2015	223.2	387.1	247.7	109.1
2	2016	312.8	446.2	381.6	108.1
3	2017	118.6	609.7	172.5	75.6
4	2018	368.1	586.5	351.2	66.5

csv 파일 읽고 쓰기

◆ Csv 파일 저장하기

> df.to_csv(file_name [, encoding = 인코딩_방식
, index = true(기본) / false,
, header = true / false,
, sep = 구분자(기본: ‘,’)])

- Index / header : dataframe의 index를 csv 파일에 포함할지 여부 지정

1. 예제: 오른쪽의 데이터 프레임을 생성하여 csv 파일로 저장하시오.
(to_csv -> utf-8로 인코딩 됨
엑셀은 cp949로 인코딩되어 깨져보일수 있음)

	제품ID	판매가격	판매량
0	P1001	5000	50
1	P1002	7000	93
2	P1003	8000	70
3	P1004	10000	48

csv 파일 읽고 쓰기

◆ Csv 파일 저장하기

> df.to_csv(file_name [, encoding = 인코딩_방식
, index = true(기본) / false,
, header = true / false,
, sep = 구분자(기본: ‘,’)])

- Index / header : dataframe의 index를 csv 파일에 포함할지 여부 지정

1. 예제: 오른쪽의 데이터 프레임을 생성하여 csv 파일로 저장하시오.
(to_csv -> utf-8로 인코딩 됨
엑셀은 cp949로 인코딩되어 깨져 보일수 있음)

	제품ID	판매가격	판매량
0	P1001	5000	50
1	P1002	7000	93
2	P1003	8000	70
3	P1004	10000	48

csv 파일 읽고 쓰기


```
import pandas as pd
df = pd.DataFrame({ '제품ID': ['P1001', 'P1002', 'P1003', 'P1004'],
                    '판매가격':[5000, 7000, 8000, 10000],
                     '판매량':[50, 93, 70, 48]}  )

folder = 'C:/myPyExcel/data/ch05/'
csv_file = folder + 'product.csv'
df.to_csv(csv_file, encoding = 'cp949', index=False)
print("생성한 csv파일", csv_file)
```

```
생성한 csv파일 C:/myPyExcel/data/ch05/product.csv
```

엑셀 파일 읽고 쓰기

◆ 엑셀 파일 읽기

df = pd.read_excel(excel_file [, Sheet_name = 시트_이름,
index_col = 숫자 혹은 열_이름,
header = 숫자 혹은 none,
names = 열 이름 리스트])

```
C:₩Users₩user>pip install openpyxl
Collecting openpyxl
  Downloading openpyxl-3.0.10-py2.py3-none-any.whl (242 kB)
     ---------------------------------------- 242.1/242.1 kB 7.5 MB/s eta 0:00:00
Collecting et-xmlfile
  Using cached et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.0.10
```

엑셀 파일 읽고 쓰기


```
# 엑셀 파일 경로
folder = 'C:/myPyExcel/data/ch05/'
excel_file = folder + '사원별_월간_판매현황.xlsx'
# 첫 줄에 열 이름 있는 엑셀 파일

# 엑셀 파일을 읽어서 DataFrame 데이터를 생성(header와 names 옵션 지정)
names_list = ["사원명", "1월달", "2월달", "3월달", "4월달", "5월달", "6월달"]
df = pd.read_excel(excel_file, header=0, names=names_list)
df
```

	사원명	1월달	2월달	3월달	4월달	5월달	6월달
0	양동호	69	54	76	34	67	56
1	조순열	65	47	85	12	56	34
2	박영순	76	85	57	42	89	91
3	고지영	98	69	23	82	67	87
4	지수경	45	39	56	98	34	53
5	오선호	56	34	56	76	95	73
6	진가연	90	57	34	44	58	96
7	최소진	45	63	76	15	85	54
8	한영미	81	75	23	97	53	95

엑셀 파일 읽고 쓰기

◆ 엑셀 파일 쓰기

```
df.to_excel(excel_file
            [, index = true / false
             , header = true / false
             , sheet_name = 시트_이름 혹은 시트 번호
             , startrow = 숫자
             , startcol = 숫자 ])
```

※to_excel()을 이용해 수행하기 전 반드시 해당 파일을 닫아야 함

엑셀 파일 읽고 쓰기

1. '사원별_월간_판매현황2.xlsx' 엑셀의 첫번째 워크시트를 읽어서 '사원별_월간_판매현황_new.xlsx'로 엑셀 파일 작성하기

엑셀 파일 읽고 쓰기

1. '사원별_월간_판매현황2.xlsx' 엑셀의 첫번째 워크시트를 읽어서 '사원별_월간_판매현황_new.xlsx'로 엑셀 파일 작성하기

```
# 엑셀 파일 경로
folder = 'C:/myPyExcel/data/ch05/'
excel_file = folder + '사원별_월간_판매현황2.xlsx'

# 엑셀 파일의 첫 번째 워크시트를 읽어서 DataFrame 데이터(df1) 생성
df1 = pd.read_excel(excel_file, sheet_name=0)
# 엑셀 파일의 두 번째 워크시트를 읽어서 DataFrame 데이터(df2) 생성
df2 = pd.read_excel(excel_file, sheet_name=1)
df1
```

	이름	1월	2월	3월	4월	5월	6월
0	양동호	69	54	76	34	67	56
1	조순열	65	47	85	12	56	34
2	박영순	76	85	57	42	89	91
3	고지영	98	69	23	82	67	87
4	지수경	45	39	56	98	34	53
5	오선호	56	34	56	76	95	73
6	진가연	90	57	34	44	58	96
7	최소진	45	63	76	15	85	54
8	한영미	81	75	23	97	53	95

```
# 엑셀 파일 경로
folder = 'C:/myPyExcel/data/ch05/'
excel_file = folder + '사원별_월간_판매현황_new.xlsx'

# DataFrame 데이터를 엑셀 파일로 쓰기
df1.to_excel(excel_file)

print("생성한 엑셀 파일:", excel_file) # 생성한 파일 이름 출력
```

```
생성한 엑셀 파일: C:/myPyExcel/data/ch05/사원별_월간_판매현황_new.xlsx
```

표 데이터 연산

◆ 기본연산

- Series 데이터 끼리 사칙연산 -> 데이터의 value 값끼리 계산
- Series 데이터에 상수 계산 -> 데이터의 value 값에 상수 계산

- Dataframe 데이터 끼리 사칙연산 -> 데이터의 value 값끼리 계산
- Dataframe 데이터에 상수 계산 -> 데이터의 value 값에 상수 계산

표 데이터 선택

◆ Series나 dataframe은 loc, iloc를 이용하여 행 데이터를 선택

1. Series_data.loc[index_label_item]
2. Dataframe_data.loc[index_label_item]
   -> index_label_item: index 라벨에 기반한 지정방식

3. Series_data.iloc[index_pos_item]
4. Dataframe_data.loc[index_pos_item]
   -> index_label_item: index 위치에 기반한 지정방식

표 데이터 선택

◆ Series나 dataframe은 loc, iloc를 이용하여 행 데이터를 선택

1. 예시

```
import pandas as pd
import numpy as np

index_data = ['a', 'b', 'c', 'd', 'e']
data = [0.0, 1.0, 2.0, 3.0, 4.0]
s1 = pd.Series(data, index = index_data)
s1
```

```
a    0.0
b    1.0
c    2.0
d    3.0
e    4.0
dtype: float64
```

```
s1.loc[['a', 'c', 'e']]
# index 라벨 리스트 지정으로 여러 행의 데이터를 선택
```

```
a    0.0
c    2.0
e    4.0
dtype: float64
```

```
s1.iloc[[0, 2, 4]]
# index 위치 리스트 지정으로 여러 행의 데이터를 선택
```

```
a    0.0
c    2.0
e    4.0
dtype: float64
```

표 데이터 선택

◆ Series나 dataframe은 loc, iloc를 이용하여 행 데이터를 선택

1. 문제: df1.loc['b':'d'], df1.iloc[[1,3,4]] 구하기

```
dict_data = {'A': [0, 10, 20, 30, 40],
             'B': [0, 0.1, 0.2, 0.3, 0.4],
             'C': [0, 100, 200, 300, 400]} # 딕셔너리 데이터

index_data = ['a', 'b', 'c', 'd', 'e'] # index 지정용 데이터

df1 = pd.DataFrame(dict_data, index=index_data)
# 딕셔너리 데이터로부터 DataFrame 데이터 생성
df1
```

	A	B	C
a	0	0.0	0
b	10	0.1	100
c	20	0.2	200
d	30	0.3	300
e	40	0.4	400

표 데이터 선택

◆ Series나 dataframe은 loc, iloc를 이용하여 행 데이터를 선택

1. 실습: df1.loc['b':'d'], df1.iloc[[1,3,4]] 구하기

```
dict_data = {'A': [0, 10, 20, 30, 40],
             'B': [0, 0.1, 0.2, 0.3, 0.4],
             'C': [0, 100, 200, 300, 400]} # 딕셔너리 데이터

index_data = ['a', 'b', 'c', 'd', 'e'] # index 지정용 데이터

df1 = pd.DataFrame(dict_data, index=index_data)
# 딕셔너리 데이터로부터 DataFrame 데이터 생성
df1
```

	A	B	C
a	0	0.0	0
b	10	0.1	100
c	20	0.2	200
d	30	0.3	300
e	40	0.4	400

```
df1.loc['b':'d']
```

	A	B	C
b	10	0.1	100
c	20	0.2	200
d	30	0.3	300

```
df1.iloc[[1,3,4]]
```

	A	B	C
b	10	0.1	100
d	30	0.3	300
e	40	0.4	400

표 데이터 선택

◆ 조건을 지정하여 행 데이터 선택 -> 불 인덱싱

```
# DataFrame 데이터 생성
dict_data = {'지점': ['서울', '대전', '대구', '부산', '광주'],
             '1월': [558, 234, 340, 380, 213],
             '2월': [437, 216, 238, 290, 194],
             '3월': [337, 196, 209, 272, 186]} # 딕셔너리 데이터

df = pd.DataFrame(dict_data)
# 딕셔너리 데이터로부터 DataFrame 데이터 생성
df
```

	지점	1월	2월	3월
0	서울	558	437	337
1	대전	234	216	196
2	대구	340	238	209
3	부산	380	290	272
4	광주	213	194	186

1. Df에서 1월 열 데이터 값이 300이상인 조건을 만족하는 행 데이터
2. Df에서 지점 데이터가 서울이거나 부산인 행 데이터

표 데이터 선택


```
df[df['1월'] >= 300] # 조건을 만족하는 행 데이터 가져오기
```

	지점	1월	2월	3월
0	서울	558	437	337
2	대구	340	238	209
3	부산	380	290	272

```
df[(df['지점'] == '서울') | (df['지점'] == '부산')]
# 둘 중 하나만 만족해도 행을 선택
```

	지점	1월	2월	3월
0	서울	558	437	337
3	부산	380	290	272

행이나 열 데이터 삭제


```
import numpy as np
import pandas as pd

s3 = pd.Series([10, 20, 30, 40, np.nan, 60]) # Series 데이터 생성
s3
```

```
0    10.0
1    20.0
2    30.0
3    40.0
4     NaN
5    60.0
dtype: float64
```

```
s3.drop(index = 0) # Series 데이터에서 하나의 행을 제거
```

```
1    20.0
2    30.0
3    40.0
4     NaN
5    60.0
dtype: float64
```

xlsxwriter

YONSEI MIRAE CAMPUS

XIsxwriter 기본 사용법


```
C:₩Users₩user>pip install xlsxwriter
Collecting xlsxwriter
  Downloading XlsxWriter-3.0.3-py3-none-any.whl (149 kB)
     ---------------------------------------- 150.0/150.0 kB 4.5 MB/s eta 0:00:00
Installing collected packages: xlsxwriter
Successfully installed xlsxwriter-3.0.3
```

Cmd창에서 pip install xlsxwriter 실행

Xlsxwriter 기본 사용법

1. 생성할 액셀 파일이름을 지정해 워크북 생성

   > workbook = xlswriter.Workbook(excel_file)

2. 워크북 내에 사용할 워크시트 생성

   > worksheet = workbook.add_worksheet([worksheet_name])

3. 워크시트의 셀에 쓰기 작업 수행

   > worksheet.write(row, col, cell_data) #셀 행과 열 위치로 지정

   > worksheet.write(cell_address, cell_data) # 셀 주소로 지정

4. 워크북 객체를 닫고 엑셀 파일 생성

   > workbook.close()

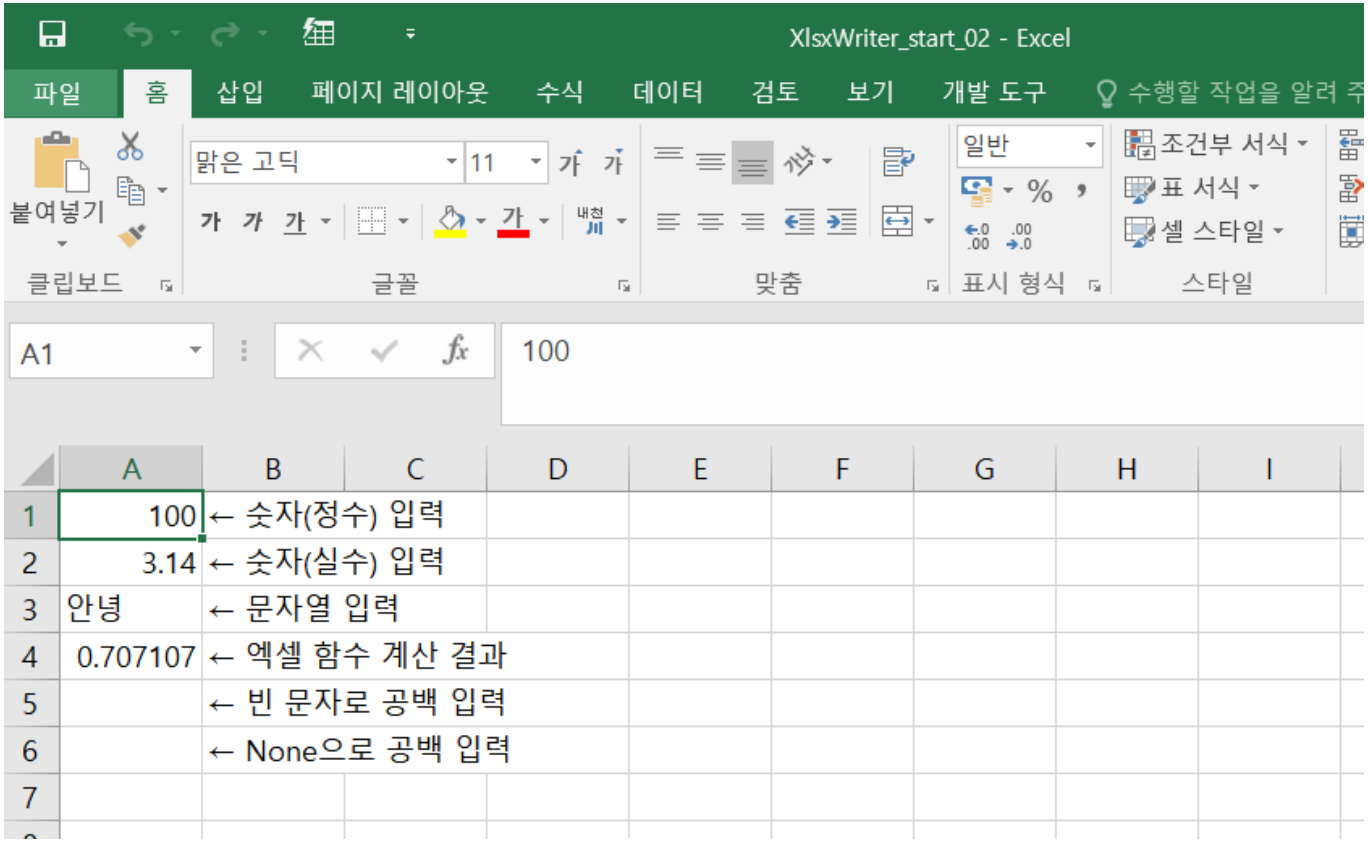
Xlsxwriter 기본 사용법


```
import xlsxwriter

# 엑셀 파일 경로
folder = 'C:/myPyExcel/data/ch06/'
excel_file = folder + 'XlsxWriter_start_01.xlsx'

workbook = xlsxwriter.Workbook(excel_file) # 워크북 객체 생성
worksheet = workbook.add_worksheet() # 워크시트 생성

worksheet.write(0, 0, 100) # 셀의 행과 열의 위치를 지정해 셀에 데이터 쓰
# worksheet.write('A1', 100) # 셀의 주소 지정 후 셀에 데이터 쓰기

workbook.close() # 워크북 객체를 닫고 엑셀 파일 생성

print("생성한 엑셀 파일:", excel_file) # 생성한 파일 이름 출력
```

```
생성한 엑셀 파일: C:/myPyExcel/data/ch06/XlsxWriter_start_01.xlsx
```

Xlsxwriter 기본 사용법

파일 이름을 'XlsxWriter_start_02.xlsx'로 지정해서 아래와 같은 엑셀을 작성하시오



XlsxWriter 기본 사용법


```python
import xlsxwriter

# 엑셀 파일 경로
folder = 'C:/myPyExcel/data/ch06/'
excel_file = folder + 'XlsxWriter_start_02.xlsx'

workbook = xlsxwriter.Workbook(excel_file) # 워크북 객체 생성
worksheet = workbook.add_worksheet() # 워크시트 생성

# 행과 열의 위치로 셀을 지정해 데이터 입력
worksheet.write(0, 0, 100) # 숫자(정수) 입력
worksheet.write(1, 0, 3.14)  # 숫자(실수) 입력
worksheet.write(2, 0, '안녕') # 문자열 입력
worksheet.write(3, 0, '=COS(PI()/4)') # 엑셀 함수를 입력
worksheet.write(4, 0, '') # 공백 입력
worksheet.write(5, 0, None) # 공백 입력

# 주소로 셀을 지정해 데이터 입력
worksheet.write('B1', '← 숫자(정수) 입력') # 문자열 입력
worksheet.write('B2', '← 숫자(실수) 입력') # 문자열 입력
worksheet.write('B3', '← 문자열 입력') # 문자열 입력
worksheet.write('B4', '← 엑셀 함수 계산 결과') # 문자열 입력
worksheet.write('B5', '← 빈 문자로 공백 입력') # 문자열 입력
worksheet.write('B6', '← None으로 공백 입력') # 문자열 입력

workbook.close() # 워크북 객체를 닫고 엑셀 파일 생성

print("생성한 엑셀 파일:", excel_file) # 생성한 파일 이름 출력
```

XlsxWriter 기본 사용법

1. 리스트 데이터를 엑셀에서 쓰기

   > worksheet.write_row(row, col, list_data)

   > worksheet.write_column(row, col, list_data)

```
# 엑셀 파일 경로
folder = 'C:/myPyExcel/data/ch06/'
excel_file = folder + 'XlsxWriter_list_data_03.xlsx'

workbook = xlsxwriter.Workbook(excel_file) # 워크북 객체 생성
worksheet = workbook.add_worksheet()       # 워크시트 생성

list_num = [10, 20, 30, 40]
list_num2 = [50, 60, 70, 80]
worksheet.write_row(0, 1, list_num)        # 셀 B1에서 시작해 행 방향으로 쓰기
worksheet.write_column(1, 0, list_num2)    # 셀 A2에서 시작해 열 방향으로 쓰기

workbook.close()                           # 워크북 객체를 닫고 엑셀 파일 생성

print("생성한 엑셀 파일:", excel_file)     # 생성한 파일 이름 출력
```

```
생성한 엑셀 파일: C:/myPyExcel/data/ch06/XlsxWriter_list_data_03.xlsx
```

Xlsxwriter 기본 사용법

2. 딕셔너리 데이터를 엑셀의 셀에 작성하기**(실습)**

   - 딕셔너리 데이터의 키는 엑셀의 첫번째 행에 작성
   - 딕셔너리 데이터의 값(value)는 그 아래에 작성

   tip: 딕셔너리의 key와 value를 추출하여 리스트에 작성해서 사용

```
dict_data = { '제품ID': ['P1001', 'P1002', 'P1003', 'P1004'],
              '판매가격': [5000, 7000, 8000, 10000],
              '판매량': [50, 93, 70, 48] }
dict_data
```

```
{'제품ID': ['P1001', 'P1002', 'P1003', 'P1004'],
 '판매가격': [5000, 7000, 8000, 10000],
 '판매량': [50, 93, 70, 48]}
```

XlsxWriter 기본 사용법


```python
import xlsxwriter

# 엑셀 파일 경로
folder = 'C:/myPyExcel/data/ch06/'
excel_file = folder + 'XlsxWriter_dict_data_01.xlsx'

workbook = xlsxwriter.Workbook(excel_file) # 워크북 객체 생성
worksheet = workbook.add_worksheet()       # 워크시트 생성

list_keys  = list(dict_data.keys())        # 딕셔너리 키를 추출해 리스트로 변환
list_values = list(dict_data.values())     # 딕셔너리 값을 추출해 리스트로 변환

worksheet.write_row(0, 0, list_keys)       # 첫 번째 행에 키를 행 방향으로 쓰기

# 두 번째 행에 리스트 데이터를 열 방향으로 쓰기
for col, list_value in enumerate(list_values):
    worksheet.write_column(1, col, list_value)

workbook.close()                           # 워크북 객체를 닫고 엑셀 파일 생성

print("생성한 엑셀 파일:", excel_file)     # 생성한 파일 이름 출력
```

Openpyxl

YONSEI MIRAE CAMPUS

Openpyxl


```
C:₩Users₩user>pip install openpyxl
Collecting openpyxl
  Downloading openpyxl-3.0.10-py2.py3-none-any.whl (242 kB)
     ---------------------------------------- 242.1/242.1 kB 7.5 MB/s eta 0:00:00
Collecting et-xmlfile
  Using cached et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.0.10
```

Cmd창에서 pip install openpyxl 실행

Openpyxl

1. 새 워크북 생성

   > wb = openpyxl.Workbook()

   #워크북 생성시 자동으로 1개의 워크시트 생성됨

   > ws = wb.active #현재 활성화 된 워크시트 가리킴

2. 기존 워크북 불러오기

   > wb = openpyxl.load_workbook(filename = 'filename.xlsx')

   #기존 엑셀 파일 불러오기

   > ws= wb.active #현재 활성화되어 있는 시트 가리킴

   > ws = wb['sheet1'] #시트명 가리킴

3. 워크시트 생성

   > wb.create_sheet('새 시트이름',0) #0번째 제일 왼쪽에 위치시킴

Openpyxl


```
import openpyxl

wb = openpyxl.Workbook()
#워크북을 생성하면 그 안에 워크시트 1개가 자동으로 생성
ws = wb.active
# 활성화 된 워크시트를 가리킴
```

```
print(ws['A1']) # A1 셀 자체를 가리킴
print(ws['A1'].value)   # A1 셀의 내용을 확인

#또 다른 셀 접근방법
ws.cell(row = 5, column = 2)
# 세로방향(row)로 5번째, 가로방향(column)으로 2번째 셀을 의미

# 첫째행 타이틀 적기 예제
# 제목 적기
sub = ['번호', '이름', '주소', '이메일']
for kwd, j in zip(sub, list(range(1, len(sub)+1))):
    ws.cell(row=1, column=j).value = kwd
```

```
<Cell 'Sheet'.A1>
None
```

```
wb.save(filename='C:/myPyExcel/data/ch06/filename.xlsx')
wb.close()
```

- 셀 접근법

  -ws['A1'] 셀 자체

  -ws['A1'].value

  A1 셀의 내용 확인

  -ws.cell(row=a, column=b)

  (a,b) 셀 의미

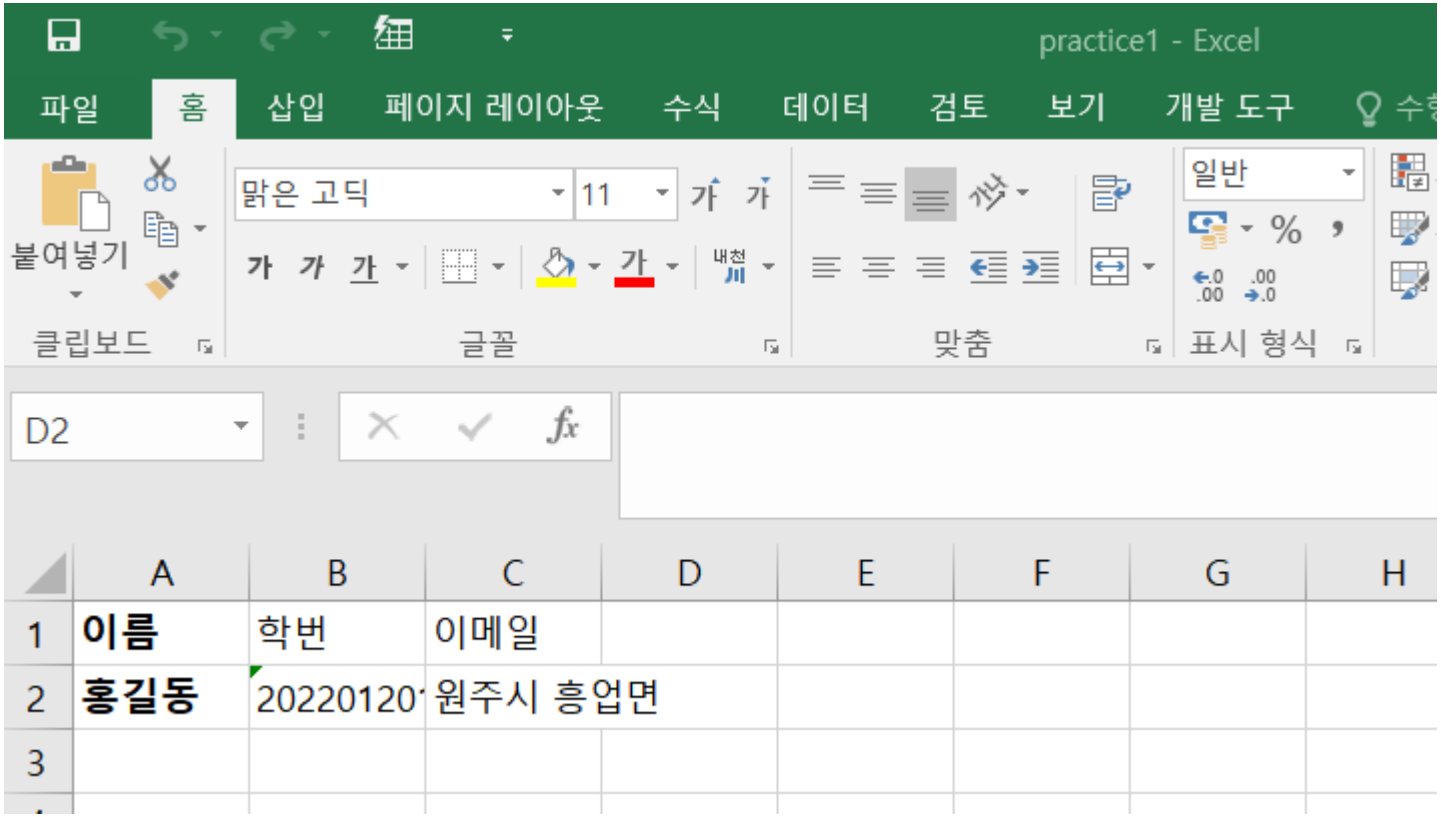
- Wb.save 파일 저장
- Wb.close() 파일 닫기

Openpyxl


```
import openpyxl

wb = openpyxl.Workbook()
ws = wb.active

ws['B2'] = 'cell' #두번째 worksheet2의 'B2'에 'cell' 쓰기

from openpyxl.styles import Alignment, Font, Border, Side, PatternFill

#가운데 정렬
align_center = Alignment(horizontal='center', vertical='center')

#글씨체 굵게
font_bold = Font(size=12, bold=True, color='000000') # 000000: black

#셀 색깔 채우기
fill_blue = PatternFill('solid', fgColor='819FF7')

# 테두리 선넣기
thin_border = Border(left=Side(border_style='thin', color='000000'),
right=Side(border_style='thin', color='000000'),
top=Side(border_style='thin', color='000000'),
bottom=Side(border_style='thin', color='000000'))

#위의 4가지 B2 cell에 적용시키기 + 값 'cell'에서 'text'로 바꾸기
ws['B2'].alignment = align_center
ws['B2'].font = font_bold
ws['B2'].fill = fill_blue
ws['B2'].border = thin_border
ws['B2'].value = 'text'

filename = "C:/myPyExcel/data/ch06/test.xlsx"
wb.save(filename)
wb.close()
```

Openpyxl

◆ 아래와 같은 내용이 들어가는 엑셀파일을 작성하시오



Openpyxl

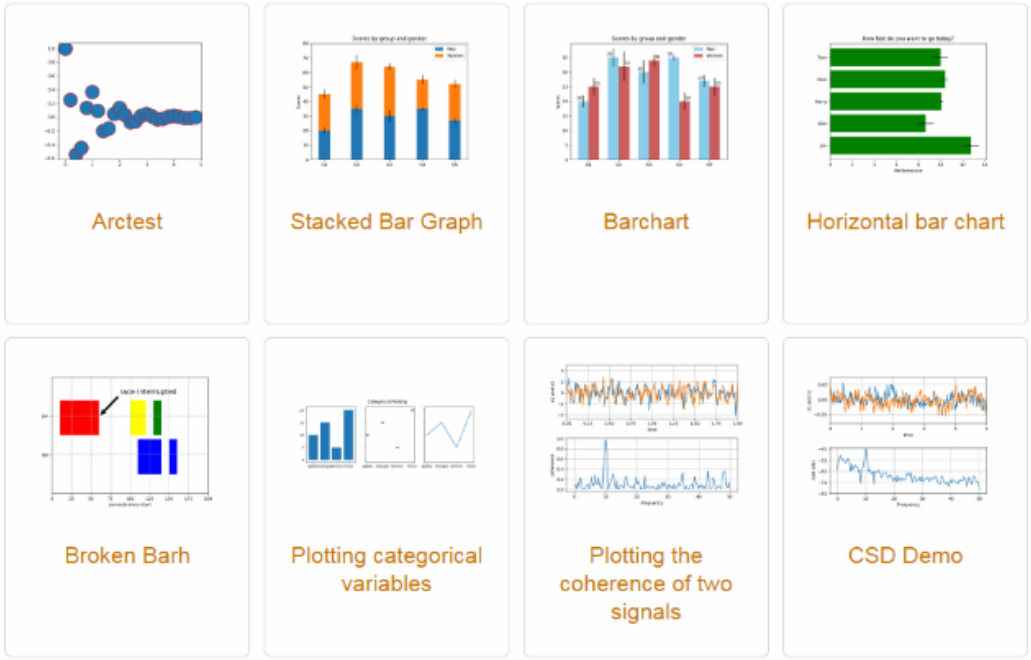

```
import openpyxl

wb = openpyxl.Workbook()
ws = wb.active

sub = ['이름', '학번','이메일']
for kwd, j in zip(sub, list(range(1, len(sub)+1))):
    ws.cell(row=1, column=j).value = kwd

ws.append(['홍길동','2022012011','원주시 흥업면'])

from openpyxl.styles import Alignment, Font, Border, Side, PatternFill
font_bold = Font(size=12, bold=True, color='000000') # 000000: black

ws['A1'].font = font_bold
ws['A2'].font = font_bold
#반복문으로 설정가능 my_range = ws['A1':'A3']

filename = "C:/myPyExcel/data/ch06/practice1.xlsx"
wb.save(filename)
wb.close()
```

Pandas, matplotlib

YONSEI MIRAE CAMPUS

matplotlib

- 데이터 시각화와 2D 그래프 플롯에 사용되는 파이썬 라이브러리
- 꺾은선, 막대 그래프, 히스토그램 등 다양한 유형의 그래프
- 데이터 곡선, 눈금, 범례, 제목 등 다양한 그래프 구성 요소를 커스터마이즈



https://codetorial.net/matplotlib/index.html

그래프 그리기

[그래프를 그리기 위한 기본 구조]

Import pandas as pd

Import matplotlib.pyplot as plt

Series_data.plot([kind = 'graph_kind'][,options])

Dataframe_data.plot([kind = 'graph_kind'][,x=label , y=label] [,options])

plt.show()

그래프 그리기

[그래프를 그리기 위한 기본 구조]

*kind -> 기본값 선 그래프
(bar: 수직 막대 / barh: 수평 막대 / hist: 히스토그램 / box: 박스 /
pie:파이그래프 / scatter: 산점도)

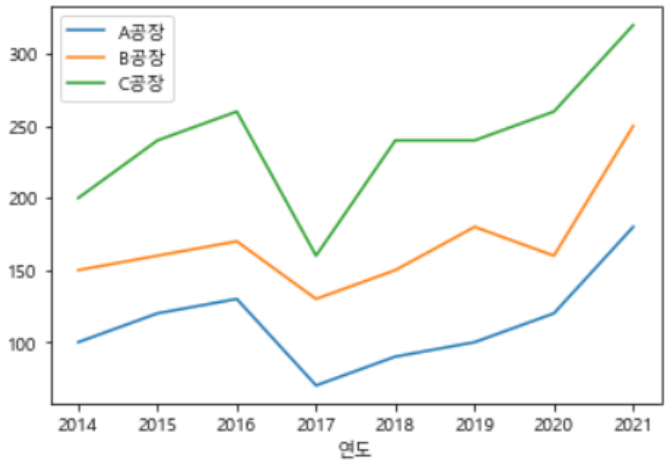
*한글폰트 깨지기 방지

```
Import matplotlib

matplotlib.rcParams['font.family'] = 'Malgun Gothic'

matplotlib.rcParams['axes.Unicode_minus'] = False
```

그래프 그리기

```
import pandas as pd
folder = 'C:/myPyExcel/data/ch08/'                  # 엑셀 파일이 있는 디렉터리(폴더)
excel_file = folder + '공장별_생산현황.xlsx'        # 원본 엑셀 파일

import matplotlib as mpl

mpl.rcParams['font.family'] = 'Malgun Gothic' # '맑은 고딕'으로 폰트 설정
mpl.rcParams['axes.unicode_minus'] = False # 마이너스(-) 폰트 깨짐 방지

import matplotlib.pyplot as plt
```

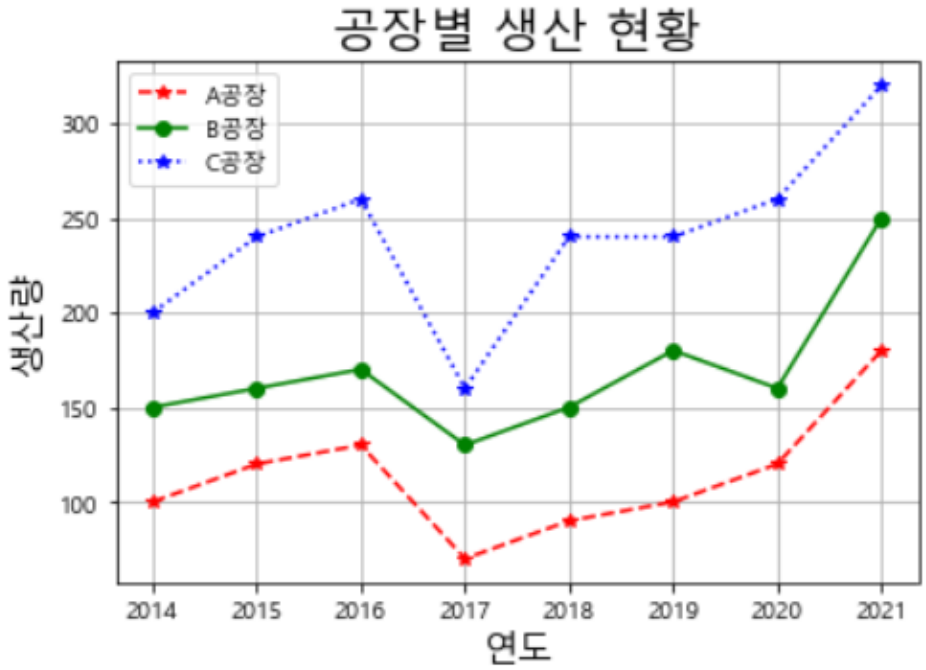
```
df = pd.read_excel(excel_file, index_col='연도') # DataFrame 데이터(df)로 읽어 오기
df

df.plot()  # 선 그래프. df.plot(kind='line')도 동일
plt.show() # 그래프 화면 출력
```




```
df.plot(y='A공장')
plt.show()
```

그래프 그리기

```
ax = df.plot(grid=True, style=['r--*', 'g-o', 'b:*']) # 격자와 스타일 지정

ax.set_xlabel("연도", fontsize=15) # x축 라벨을 지정
ax.set_ylabel("생산량", fontsize=15) # y축 라벨을 지정
ax.set_title("공장별 생산 현황", fontsize=20) # 그래프 제목을 지정

plt.show()
```



r,g,b :색상

-- : 파선

- : 실선

: : 점선

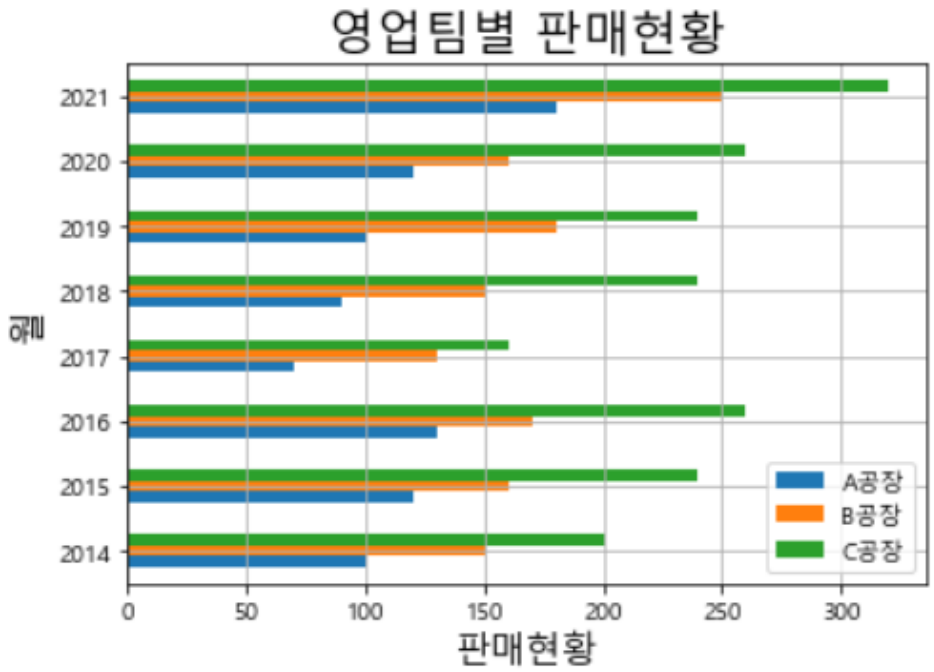
*: 별 모양

o: 원 모양

그래프 그리기

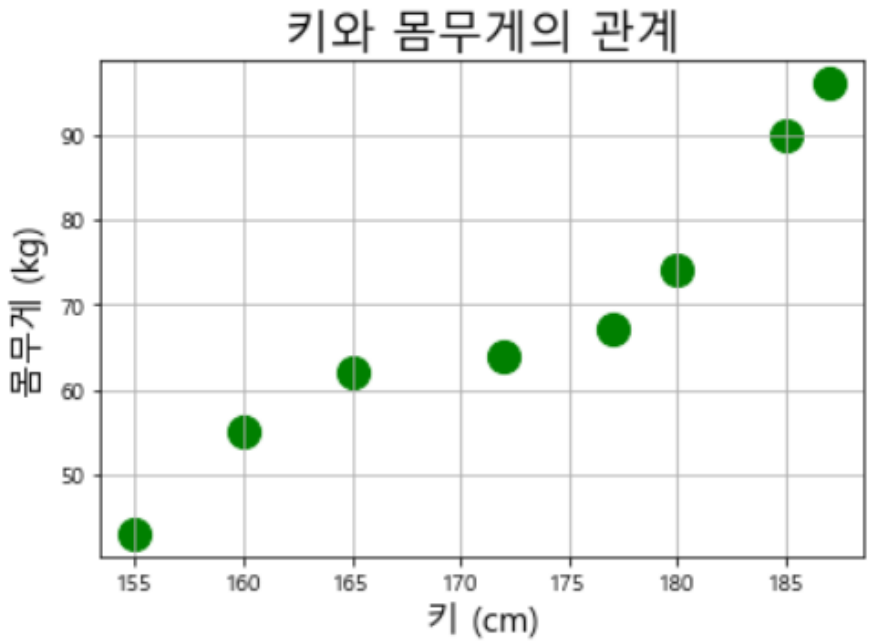
```
ax = df.plot.barh(grid=True) # 수평 막대 그래프(격자 추가)
# ax = df.plot(kind='barh', grid=True)도 동일

ax.set_xlabel("판매현황", fontsize=15) # x축 라벨을 지정
ax.set_ylabel("월", fontsize=15)       # y축 라벨을 지정
ax.set_title("영업팀별 판매현황", fontsize=20) # 그래프 제목을 지정

plt.show()
```

dataframe_Data.plot.bar([x=label][,y=label][,rot=rot_angle(기본90)][,options])



그래프 그리기


```
ax_scatter = df.plot.scatter(x='키', y='몸무게', s=200, c='g', grid=True)

ax_scatter.set_xlabel("키 (cm)", fontsize=15)
ax_scatter.set_ylabel("몸무게 (kg)", fontsize=15)
ax_scatter.set_title("키와 몸무게의 관계", fontsize=20)

plt.show()
```



S: 마커 사이즈
C: 원 색상
Grid: 격자 옵션

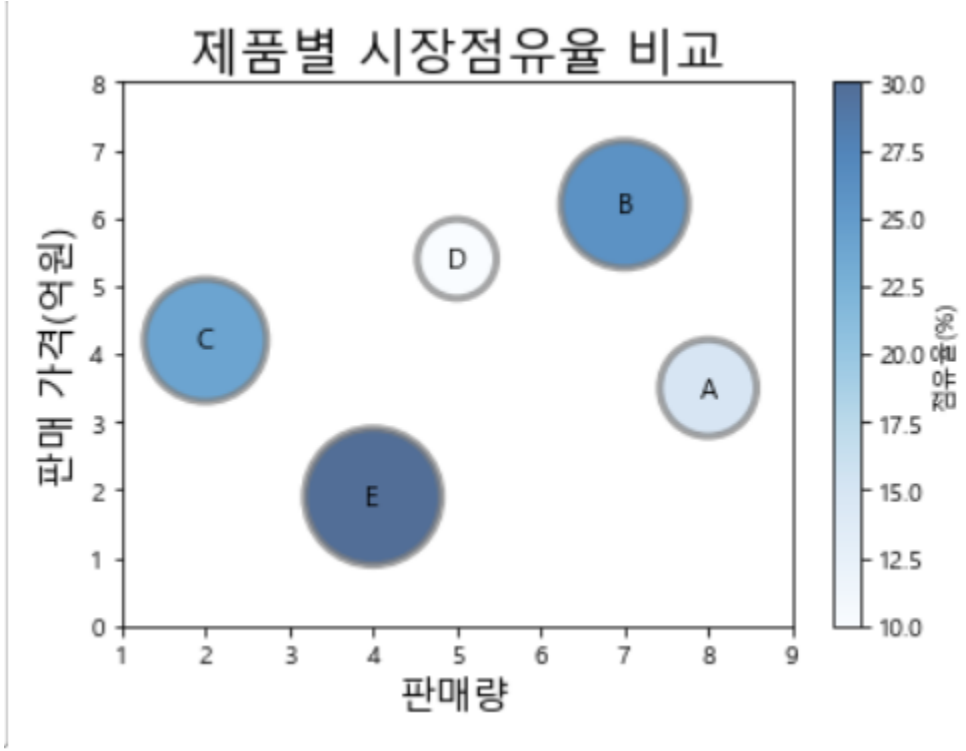
그래프 그리기


```
import pandas as pd

folder = 'C:/myPyExcel/data/ch08/' # 엑셀 파일이 있는 디렉터리(폴더)
excel_file = folder + '제품별_시장점유율.xlsx' # 원본 엑셀 파일

df = pd.read_excel(excel_file)     # 엑셀 파일을 DataFrame 데이터(df)로 읽어 오기
df

# 각 데이터의 원 마커 크기는 최댓값 대비 상대적 크기로 계산
maker_size = df['점유율(%)'] / df['점유율(%)'].max() * 3000
```

그래프 그리기


```
ax  = df.plot.scatter(x='판매량', y='가격(억원)',
                      s=maker_size,      # 마커 크기 지정
                      c='점유율(%)',     # df의 '점유율(%)'열 데이터로 마커 색 지정
                      colormap="Blues",  # 미리 정의된 컬러맵 중 하나를 지정
                      alpha=0.7,         # 투명도 선택
                      edgecolors="gray", # 마커 테두리 색 지정
                      linewidth=3,       # 마커 테두리 두께 지정
                      sharex=False)      # x축 라벨이 보이도록 지정

ax.set_xlabel("판매량", fontsize=15)
ax.set_ylabel("판매 가격(억원)", fontsize=15)
ax.set_title("제품별 시장점유율 비교", fontsize=20)

plt.axis([1, 9, 0, 8]) # x축과 y축 좌표의 범위를 지정

for x, y, text_str in zip(df['판매량'], df['가격(억원)'], df['제품명']):
    plt.text(x, y, text_str, fontsize=11, ha='center', va='center')

plt.show()
```

그래프 그리기


제품별 시장점유율 비교
판매 가격(억원)
판매량
점유율(%)
A
B
C
D
E
0
1
2
3
4
5
6
7
8
9
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0

pyautogui

YONSEI MIRAE CAMPUS

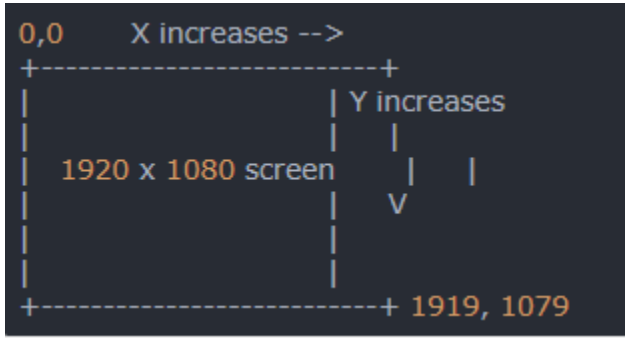
PyAutoGUI

◆ PyAutoGUI는 마우스/키보드 자동제어를 위한 패키지

➢ Pip install pyautogui 설치

```
C:₩Users₩user>pip install pyautogui
Collecting pyautogui
  Downloading PyAutoGUI-0.9.53.tar.gz (59 kB)
     ---------------------------------------- 59.0/59.0 kB 3.3 MB/s eta 0:00:00
```

➢ PyAutoGUI는 모니터 화면의 가장 왼쪽 위 꼭지점을 영점(0,0)으로 하며, 픽셀 단위로 x,y좌표를 가짐

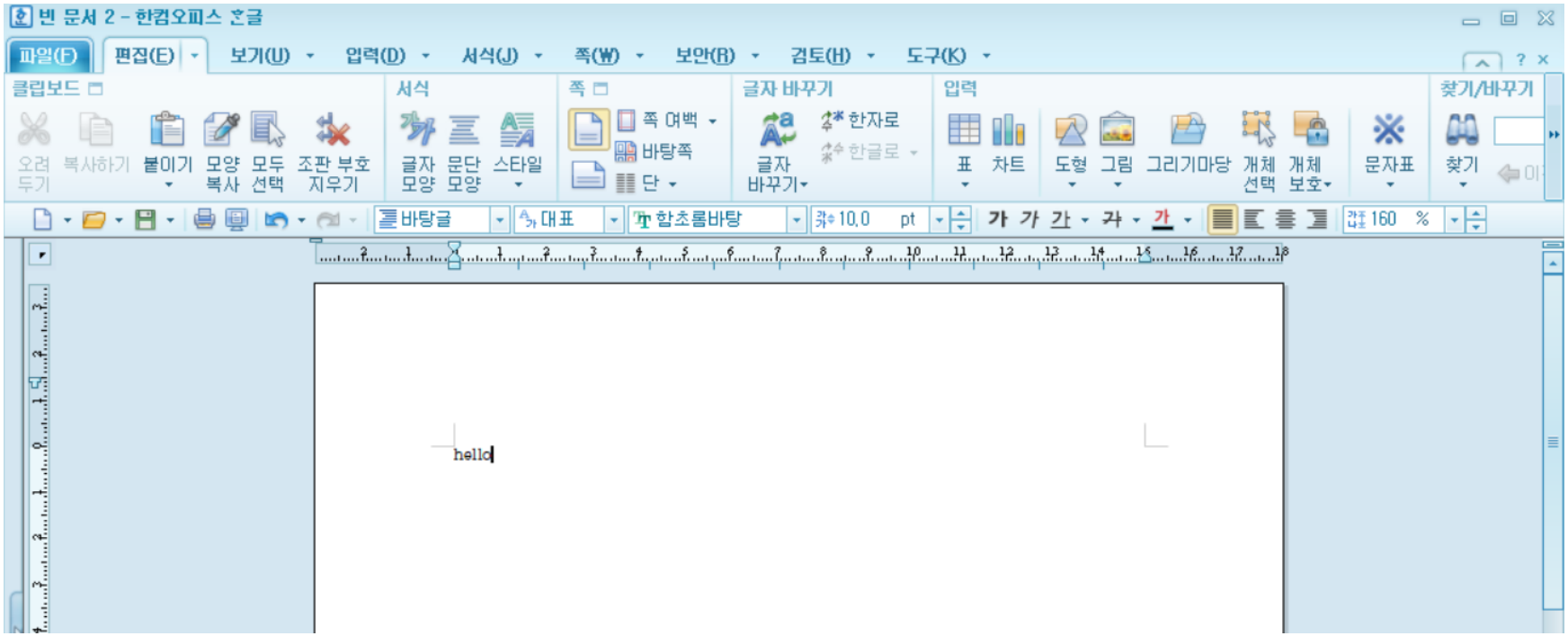


PyAutoGUI


```
import pyautogui

a=pyautogui.position() #현재 마우스 좌표


#이동
pyautogui.moveTo(x,y) #해당 좌표로 이동함

pyautogui.moveRel(0, 300) #현재 위치에서 y로 300만큼 이동

#클릭

pyautogui.click()

pyautogui.doubleClick()

#타이핑

pyautogui.typewrite("hello")
pyautogui.typewrite(['enter]) #줄바꿈
```

PyAutoGUI

```
import pyautogui
import time

pyautogui.moveTo(43,904) # 메모장 위치
pyautogui.doubleClick()
time.sleep(1) #메모장 켜지는 시간을 위한 딜레이
pyautogui.typewrite("hello")
```



Question and Answer

YONSEI MIRAE CAMPUS