

# Python을 활용한 데이터 처리 및 시각화

# 순서

시 간	주 제	내 용
2:00 ~ 2:50	Data Handling 소개	<ul style="list-style-type: none"> <li>- Data Handling ?</li> <li>- Python 소개</li> <li>- Pandas, Numpy, Matplotlib 패키지 소개</li> </ul>
	분석환경 소개	<ul style="list-style-type: none"> <li>- Jupyter Notebook 소개</li> <li>- 코드 공유 및 실행방법 소개</li> </ul>
3:00 ~ 3:50	Pandas를 활용한 Data Handling 소개 및 실습	<ul style="list-style-type: none"> <li>- 데이터 현황 파악</li> <li>- 데이터 추가</li> <li>- 데이터 결합</li> <li>- 데이터 삭제</li> <li>- 데이터 재구조화</li> </ul>
4:00 ~ 4:50	Matplotlib를 활용한 시각화 소개 및 실습	<ul style="list-style-type: none"> <li>- 시각화를 위한 데이터 구성</li> <li>- 시각화 구현</li> </ul>
	제주도 통계를 활용한 데이터 분석 실습	<ul style="list-style-type: none"> <li>- 인구통계자료를 활용하여 Data Handling 및 시각화 실습</li> </ul>

# **Data Handling & 분석환경 소개**

# Data Handling ?

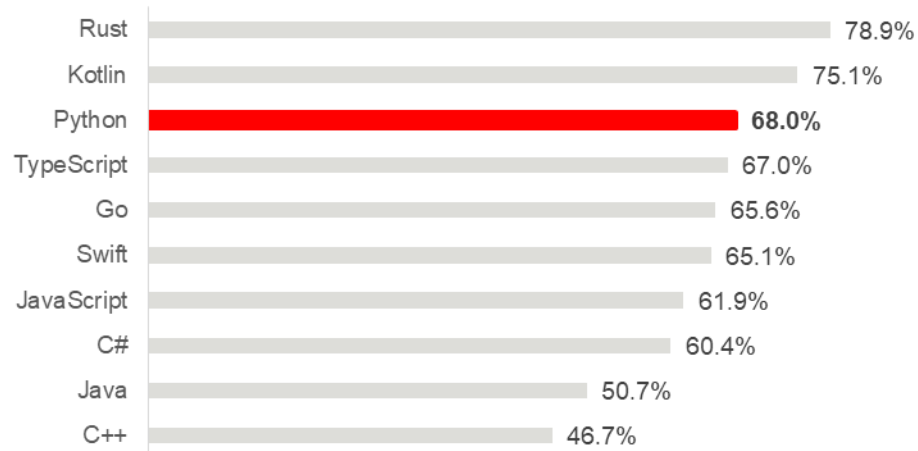
- Data Preprocessing, Data Cleaning, 데이터 전처리와 같은 의미로 쓰이며, 데이터를 분석하는 과정에서 가장 많은 시간을 차지하는 과정. 즉, 데이터를 분석에 필요한 형태로 만드는 단계
- 데이터 분석을 하기에 앞서 원하는 분석의 목적에 맞게 데이터를 분류, 변경, 추가, 제거와 같은 작업



# Python 소개

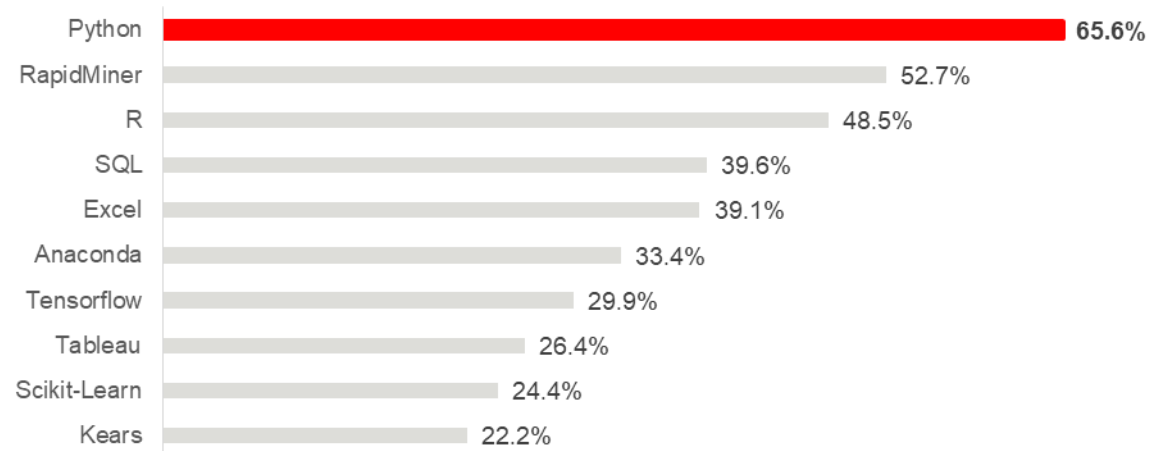
- Java, C++과 같은 프로그래밍 언어이며 쉬운 문법, 다양한 라이브러리 등의 장점으로 대학을 비롯한 연구기관 및 산업계에서 사용 중인 언어
- 데이터 분석 등 데이터를 쉽게 다룰 수 있는 라이브러리를 많이 구축하고 있기 때문에 R, SQL, Excel 등 분석가에게 반드시 필요한 Tool로 언급되는 언어

Most Loved Language in 2018



(출처: Stack Overflow - Developer Survey Results 2018)

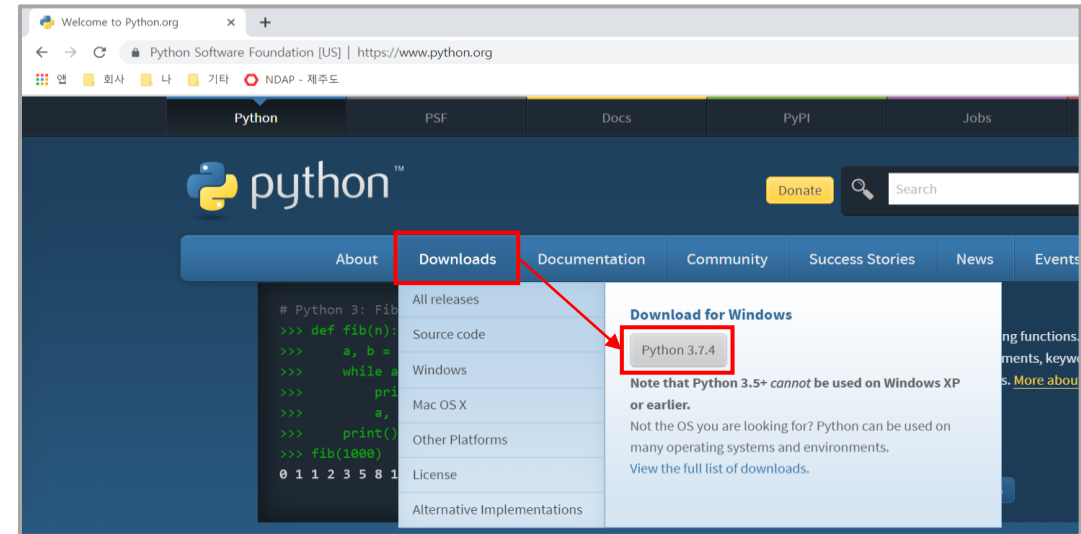
Data Science, Machine Learning Software Poll, 2018



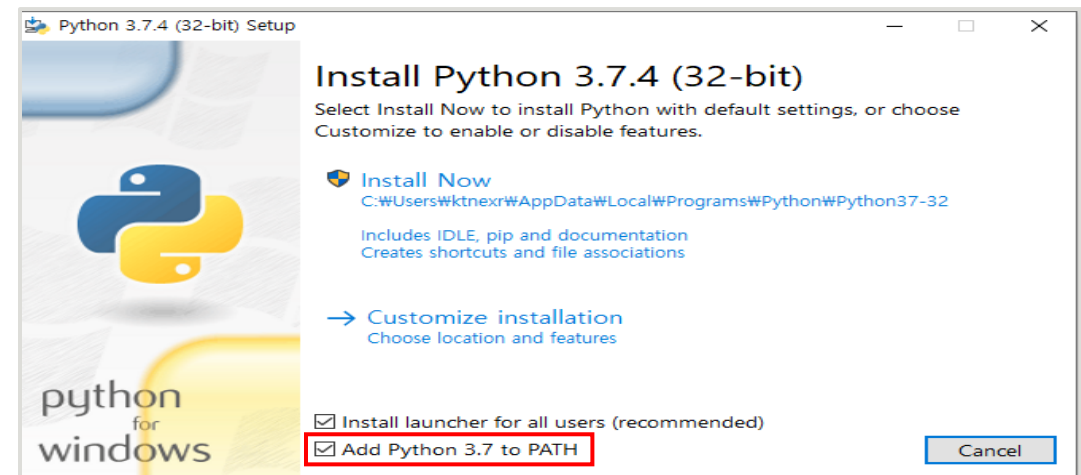
(출처: kdnuggets - Top Analytics, Data Science, Machine Learning Tools)

# Python 설치

1. <https://www.python.org> 에서  
Downloads – Python 3.x version 설치

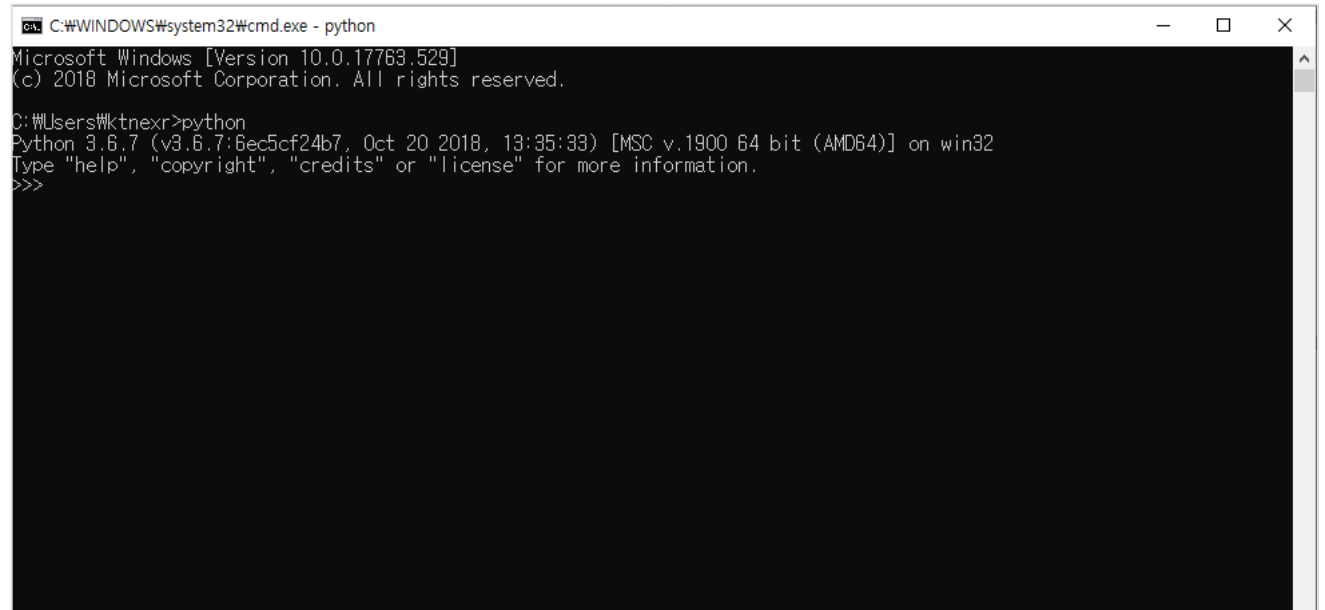
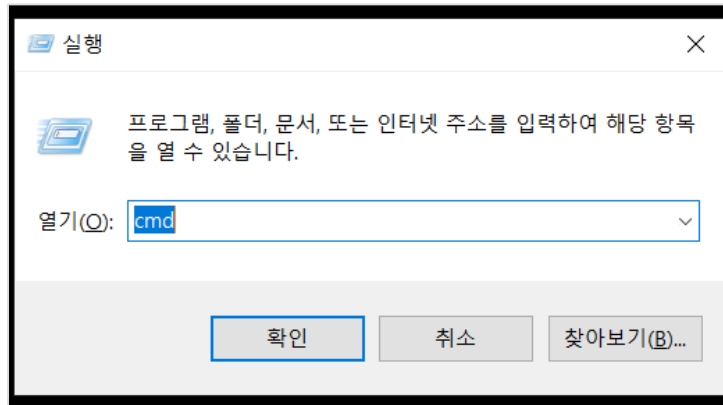


2. 설치된 파일 실행 후, Add Python 3.x to  
PATH 체크 확인 및 설치 진행



# Python 설치

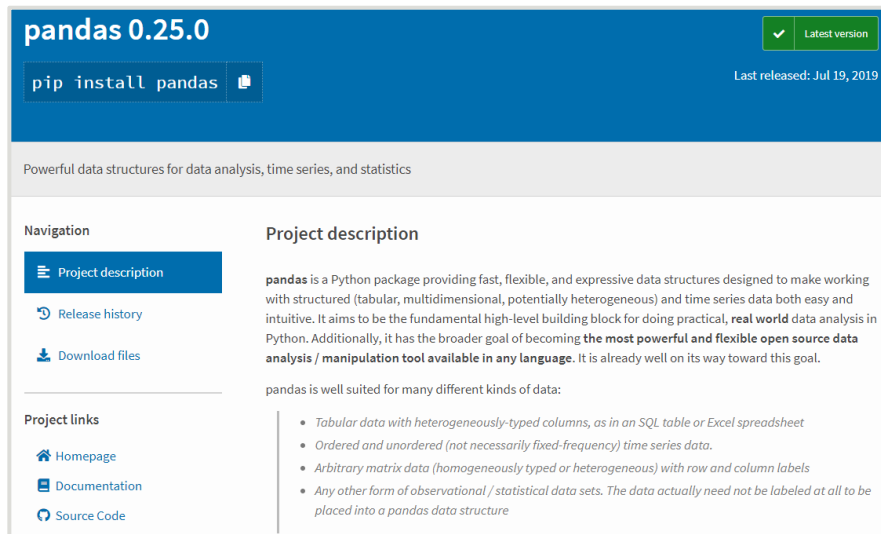
3. 실행 창에서 **cmd** 실행 후, command 창에서 **python** 실행



4. Python 3.x 가 출력 될 경우 정상 설치. Ctrl + z를 실행하여 Python 종료

# Python 라이브러리 소개

- 라이브러리란 자주 사용하는 기능을 큰 덩어리로 만들어 둔 것. 한 번 만들어 놓으면 여러 프로그램에서 재사용 할 수 있고, 다른 사람과 공유도 가능. Python의 대표적인 라이브러리로 TensorFlow, scikit-learn등 이 있음
- Library(라이브러리), Package(패키지) 두 용어를 혼용하여 사용



(출처: pypi - pandas)

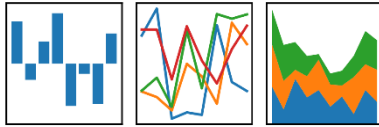




# Python 라이브러리 소개

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- 데이터 처리 및 분석을 위해 활용되는 Python 대표적인 라이브러리
- 데이터 분석을 쉽게 하기 위해 Data Frame 형식을 다룸

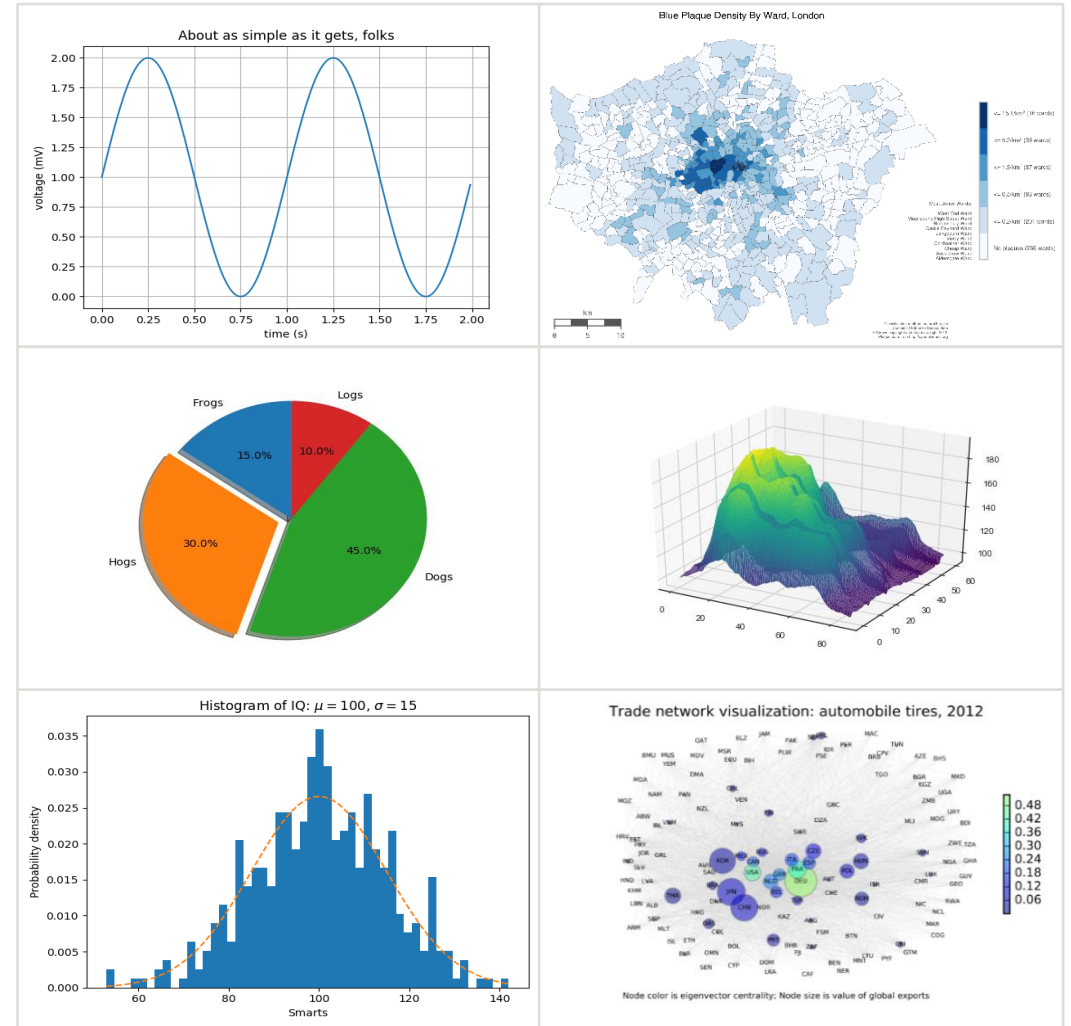


- Pandas를 사용하기 위해 반드시 필요한 라이브러리
- Pandas를 설치할 경우, 자동으로 Numpy가 설치 됨(Dependency, 의존성)
- 행렬, 다차원 배열 등을 빠르게 처리하도록 지원하는 라이브러리. 빠른 연산이 특징이며 Tensorflow와 같은 딥러닝 라이브러리와 함께 사용되기도 함

# Python 라이브러리 소개



- 데이터를 시각화(Visualization)하는 대표적인 라이브러리
- Line chart, Bar chart 등 기본 시각화 이외에도 다양한 라이브러리를 함께 사용하여 높은 수준의 시각화를 구성하는 라이브러리



# Python 라이브러리 설치

## 1. Command 창에서 `pip install pandas matplotlib` 실행

```
C:\Users\ktnexr>pip install pandas matplotlib
Requirement already satisfied: pandas in c:\python\python36\lib\site-packages (0.25.0)
Requirement already satisfied: matplotlib in c:\python\python36\lib\site-packages (3.1.1)
Requirement already satisfied: numpy>=1.13.3 in c:\python\python36\lib\site-packages (from pandas) (1.16.4)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\python\python36\lib\site-packages (from pandas) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in c:\python\python36\lib\site-packages (from pandas) (2019.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\python\python36\lib\site-packages (from matplotlib) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\python\python36\lib\site-packages (from matplotlib) (2.4.1.1)
Requirement already satisfied: cycler>=0.10 in c:\python\python36\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: six>=1.5 in c:\python\python36\lib\site-packages (from python-dateutil>=2.6.1->pandas) (1.12.0)
Requirement already satisfied: setuptools in c:\python\python36\lib\site-packages (from kiwisolver>=1.0.1->matplotlib) (39.0.1)
WARNING: You are using pip version 19.1, however version 19.2.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

## 2. Python 실행 후, `import pandas`, `import matplotlib` 실행. 에러가 발생하는지 확인

```
C:\Users\ktnexr>python
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas
>>> import matplotlib
>>> print('perfect !')
perfect !
>>>
```

# Jupyter Notebook 소개

- [Jupyter Notebook](#)은 웹 브라우저에서 Python 코드를 작성하고 실행할 수 있는 도구
- [아나콘다\(Anaconda\)](#)를 설치하면 Jupyter Notebook이 함께 설치되어 바로 사용할 수 있음
- 아나콘다를 사용하지 않는 경우 Python 설치 후 pip를 통해 Jupyter 패키지를 설치하여 사

```
In [21]: my_info['score_sum'] = my_info['score_1'] + my_info['score_2']
my_info

Out[21]:
```

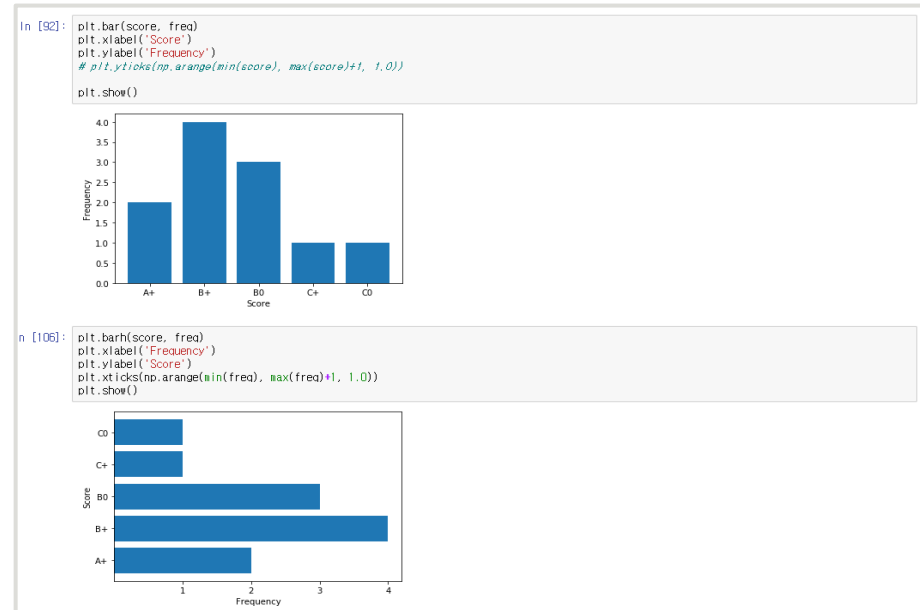
	class_code	class_name	score_1	score_2	total_score	major	score_sum
0	class_1	korean	100	100	A+	No	200
1	class_2	math	70	80	B+	Yes	150
2	class_3	english	50	40	C+	Yes	90
3	class_4	computer	65	95	B+	Yes	160
4	class_5	chinese	100	50	B+	No	150
5	class_6	stat	20	30	C0	Yes	50
6	class_7	music	60	70	B0	No	130
7	class_8	art	90	45	B0	No	135
8	class_9	python	85	95	A+	Yes	180
9	class_10	java	90	40	B0	Yes	130

```
In [22]: my_info = my_info.drop(columns=['score_sum'])
my_info

Out[22]:
```

	class_code	class_name	score_1	score_2	total_score	major
0	class_1	korean	100	100	A+	No
1	class_2	math	70	80	B+	Yes
2	class_3	english	50	40	C+	Yes
3	class_4	computer	65	95	B+	Yes
4	class_5	chinese	100	50	B+	No
5	class_6	stat	20	30	C0	Yes
6	class_7	music	60	70	B0	No
7	class_8	art	90	45	B0	No
8	class_9	python	85	95	A+	Yes
9	class_10	java	90	40	B0	Yes

Jupyter Notebook 예시



Jupyter Notebook 예시

# Jupyter Notebook 설치 및 실행

1. Command 창에서 **pip install jupyter** 실행. 설치 완료 후 jupyter-notebook 실행

```
C:\Users\ktnexr>jupyter-notebook
[I 20:08:01.982 NotebookApp] Serving notebooks from local directory: C:\Users\ktnexr
[I 20:08:01.983 NotebookApp] The Jupyter Notebook is running at:
[I 20:08:01.983 NotebookApp] http://localhost:8888/?token=be8ecc3488c65918b405d71d48142e88729abb0d2c8bd2fc
[I 20:08:01.983 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 20:08:01.991 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/ktnexr/AppData/Roaming/jupyter/runtime/nbserver-2396-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=be8ecc3488c65918b405d71d48142e88729abb0d2c8bd2fc
```

2. 웹 브라우저에서 Jupyter Notebook이 열리는지 확인



# 코드공유 및 실행방법

## 1. 구글 드라이브 접속

- <https://drive.google.com/drive/folders/1UZ2F4oiquJBjKaRwTkLmuQGdWgFmGrP0?usp=sharing>

## 2. practice\_code.ipynb 다운로드

## 3. Jupyter Notebook에서 원하는 위치에 다운받은 파일 업로드

- 적당한 위치가 없을 경우, 바탕화면(Desktop)에 work폴더를 생성하여 업로드



# 코드공유 및 실행방법

- 코드실행 단축기
  - Ctrl + Enter: 선택 셀 코드 실행
  - Shift + Enter: 선택 셀 코드 실행 후 다음 셀 이동
- 왼쪽에 In [ ] 으로 표시된 셀은 코드 작성 및 실행을 하는 셀
- In [ ] 없이 설명이 작성된 셀은 Markdown(글 작성) 셀

## 1. subset

데이터프레임에서 특정 Row 혹은 특정 Column을 선택하는 것

### 1-1 Row 선택

- column이 x보다 클 경우: `df[df.column > x]`
- column이 x보다 작을 경우: `df[df.column < x]`
- column이 x보다 크거나 같을 경우: `df[df.column >= x]`
- column이 x보다 작거나 같을 경우: `df[df.column <= x]`
- column이 x와 같을 경우: `df[df.column == x]`
- column이 x가 아닐 경우: `df[df.column != x]`
- column이 x보다 크고 y보다 작을 경우: `df[(df.column) > x & (df.column) < y]`
- column이 x보다 크거나 y보다 작을 경우: `df[(df.column) > x | (df.column) < y]`

```
In [ ]: # total_score가 B+인 경우
my_inform[my_inform.total_score == 'B+']
```

```
In [ ]: # score_1이 50보다 크고 score_2가 50보다 작을 경우
my_inform[(my_inform.score_1 > 50) & (my_inform.score_2 < 50)]
```

```
In [ ]: # score_1이 80보다 크거나 score_1이 20보다 작을 경우
my_inform[(my_inform.score_1 > 80) | (my_inform.score_1 < 20)]
```

# Tip: 검색 활용하기

- Googling

Google search results for "pandas group by average". The search bar shows the query. Below the search bar, there are links to documentation and Stack Overflow. The first result is "pandas.core.groupby.GroupBy.mean — pandas 0.25.0 documentation" with a link to the pandas website. The second result is "Python Pandas : group by in group by and average? - Stack Overflow" with a link to the Stack Overflow page. The search results also show the number of results (1,940,000) and the time taken (0.44초).

- 공식 Documentation

Documentation for `pandas.core.groupby.GroupBy.mean`. The title is "pandas.core.groupby.GroupBy.mean" with a [source] link. The description is "Compute mean of groups, excluding missing values." The "Returns:" section indicates it returns "pandas.Series or pandas.DataFrame". The "See also:" section lists `Series.groupby` and `DataFrame.groupby`. The "Examples" section shows a code snippet: 

```
>>> df = pd.DataFrame({'A': [1, 1, 2, 1, 2],
...                    'B': [np.nan, 2, 3, 4, 5],
...                    'C': [1, 2, 1, 1, 2]}, columns=['A', 'B', 'C'])
```

 and the output of `df.groupby('A').mean()` showing the mean values for columns B and C grouped by A.

- Stack Overflow

Stack Overflow question and answer. The question is "If you want to first take mean on ['cluster', 'org'] combination and then again take mean on cluster groups". The answer shows the code: 

```
In [59]: (df.groupby(['cluster', 'org'], as_index=False).mean()
...       .groupby('cluster')['time'].mean())
```

 and the output: 

```
Out[59]:
cluster
1      15
2      54
3       6
Name: time, dtype: int64
```

 The answer also includes a note: "If you want mean values by cluster only, then you could" and the code: 

```
In [58]: df.groupby(['cluster']).mean()
```

 and the output: 

```
Out[58]:
           time
cluster
1      12.333333
2      54.000000
3       6.000000
```



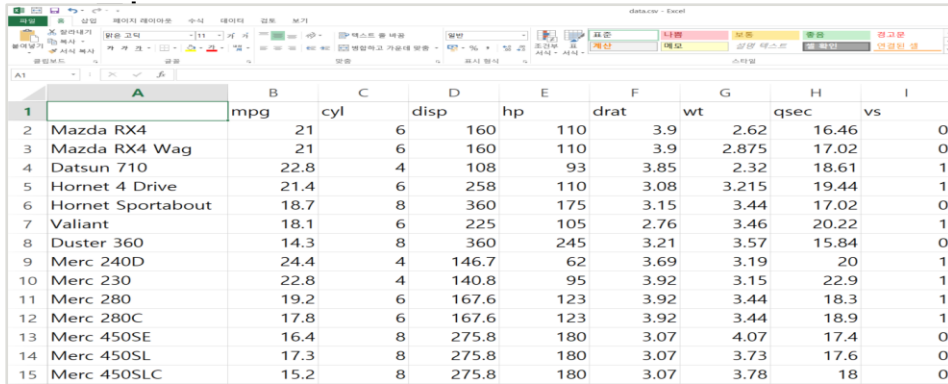
# Question & Answer



# **Pandas를 활용한 Data Handling 소개 및 실습**

# Data Import

- Data Import란 활용할 Python 등의 Tool로 불러오는 것을 의미
- 데이터 타입은 csv, xlsx, txt와 같이 PC에 물리적으로 저장되는 파일과 데이터베이스가 대



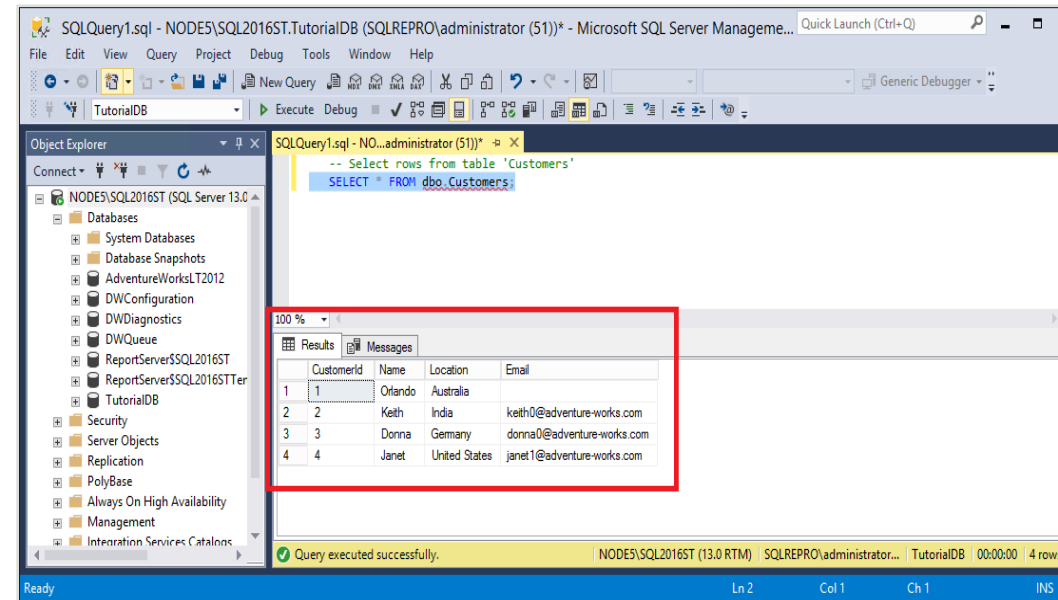
The screenshot shows an Excel spreadsheet with columns labeled A through I. The data represents various car models and their specifications. The first row (A1) contains the following values: 1, A, mpg, cyl, disp, hp, drat, wt, qsec, vs. The subsequent rows list car models like Mazda RX4, Datsun 710, etc., with their corresponding numerical values in the specified columns.

	A	B	C	D	E	F	G	H	I
1		mpg	cyl	disp	hp	drat	wt	qsec	vs
2	Mazda RX4	21	6	160	110	3.9	2.62	16.46	0
3	Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0
4	Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1
5	Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1
6	Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0
7	Valiant	18.1	6	225	105	2.76	3.46	20.22	1
8	Duster 360	14.3	8	360	245	3.21	3.57	15.84	0
9	Merc 240D	24.4	4	146.7	62	3.69	3.19	20	1
10	Merc 230	22.8	4	140.8	95	3.92	3.15	22.9	1
11	Merc 280	19.2	6	167.6	123	3.92	3.44	18.3	1
12	Merc 280C	17.8	6	167.6	123	3.92	3.44	18.9	1
13	Merc 450SE	16.4	8	275.8	180	3.07	4.07	17.4	0
14	Merc 450SL	17.3	8	275.8	180	3.07	3.73	17.6	0
15	Merc 450SLC	15.2	8	275.8	180	3.07	3.78	18	0



The screenshot shows a text file named 'data.txt' containing the Iris dataset. Each line represents a flower specimen with its sepal length, sepal width, petal length, petal width, and species name, all enclosed in double quotes and separated by commas.

"Sepal.Length"	"Sepal.Width"	"Petal.Length"	"Petal.Width"	"Species"
"1"	"5.1"	"3.5"	"1.4"	"setosa"
"2"	"4.9"	"3.1"	"1.4"	"setosa"
"3"	"4.7"	"3.2"	"1.3"	"setosa"
"4"	"4.6"	"3.1"	"1.5"	"setosa"
"5"	"5.3"	"6.1"	"4.0"	"setosa"
"6"	"5.4"	"3.9"	"1.7"	"setosa"
"7"	"4.6"	"3.4"	"1.4"	"setosa"
"8"	"5.3"	"4.1"	"1.5"	"setosa"
"9"	"4.4"	"2.9"	"1.4"	"setosa"
"10"	"4.9"	"3.1"	"1.5"	"setosa"
"11"	"5.4"	"3.7"	"1.5"	"setosa"
"12"	"4.8"	"3.4"	"1.6"	"setosa"
"13"	"4.8"	"3.1"	"1.4"	"setosa"
"14"	"4.3"	"3.1"	"1.0"	"setosa"
"15"	"5.8"	"4.1"	"2.0"	"setosa"
"16"	"5.7"	"4.4"	"1.5"	"setosa"
"17"	"5.4"	"3.9"	"1.3"	"setosa"
"18"	"5.1"	"3.5"	"1.4"	"setosa"
"19"	"5.7"	"3.8"	"1.7"	"setosa"
"20"	"5.1"	"3.8"	"1.5"	"setosa"
"21"	"5.4"	"3.4"	"1.7"	"setosa"
"22"	"5.1"	"3.7"	"1.5"	"setosa"
"23"	"4.6"	"3.6"	"1.0"	"setosa"
"24"	"5.1"	"3.3"	"1.7"	"setosa"
"25"	"4.8"	"3.4"	"1.9"	"setosa"
"26"	"5.3"	"1.6"	"0.2"	"setosa"
"27"	"5.3"	"4.1"	"6.0"	"setosa"
"28"	"5.2"	"3.5"	"1.5"	"setosa"
"29"	"5.2"	"3.4"	"1.4"	"setosa"



The screenshot shows the SQL Server Enterprise Manager interface. A query is executed in the 'SQLQuery1.sql' file, selecting all rows from the 'Customers' table in the 'TutorialDB' database. The results are displayed in a table with columns: CustomerId, Name, Location, and Email. The results show four rows of customer data.

CustomerId	Name	Location	Email
1	Orlando	Australia	
2	Keith	India	keith0@adventure-works.com
3	Donna	Germany	donna0@adventure-works.com
4	Janet	United States	janet1@adventure-works.com

# Data Import

- 최근에는 물리적 파일, 데이터베이스 외에 웹 API를 이용한 Data Import 방식도 증가하는 추세
- 웹 API를 이용할 경우 파일 업로드/다운로드 등 저장환경을 신경 쓰지 않아도 되는 장점이 있지만, API 환경을 이해하고 초기환경 설정을 하는데 시간이 걸리는 단점이 있음

API 환경 확인



코드 작성

```
library(httr)
library(dplyr)

url <- URLencode(iconv("http://www.kobis.or.kr/kobisopenapi/webservice/rest/boxoffice", "UTF-8"))

# http://www.kobis.or.kr/kobisopenapi/homepg/apikey/ckUser/findApikeyList.do

# 발급받은 Key
KEY <- 'e95cabd1202a4ffe248c09f1e1268cae'

# 요청할 데이터 날짜 (18.10.01 ~ 18.10.31)
target_date_list <- c(20181001:20181031)

## 10월 한달간 일별 1~3위 영화명 출력
# 결과 정리할 list
list_output <- list()

# wait 1-2 minute.
for(i in 1:length(target_date_list)){
  output <- c()
  target_date <- target_date_list[i]
  search_result <- GET(url, query= list("key" = KEY, "targetDt"= target_date))
  all_contents <- content(search_result)

  box_office_result <- all_contents$boxOfficeResult
```

데이터 불러오기

	date	rank_1	rank_2	rank_3
1	2018.09.27	가	가	가
2	2018.09.28	가	가	가
3	2018.09.29	가	가	가
4	2018.09.30	가	가	가
5	2018.10.01	가	가	가
6	2018.10.02	가	가	가
7	2018.10.03	가	가	가
8	2018.10.04	가	가	가
9	2018.10.05	가	가	가
10	2018.10.06	가	가	가
11	2018.10.07	가	가	가
12	2018.10.08	가	가	가
13	2018.10.09	가	가	가
14	2018.10.10	가	가	가
15	2018.10.11	가	가	가
16	2018.10.12	가	가	가
17	2018.10.13	가	가	가
18	2018.10.14	가	가	가
19	2018.10.15	가	가	가
20	2018.10.16	가	가	가
21	2018.10.17	가	가	가
22	2018.10.18	가	가	가

분석

```
> # 매출액 1~3위
> sales_and_audience_inform %>%
+ arrange(-total_sales) %>%
+ head(3)
  title total_sales total_audience
1   배움 33992735357      3873263
2 암수살인 32592248848      3739184
3   창귀 10826759714      1307862

> # 관객수 1~3위
> sales_and_audience_inform %>%
+ arrange(-total_audience) %>%
+ head(3)
  title total_sales total_audience
1   배움 33992735357      3873263
2 암수살인 32592248848      3739184
3   창귀 10826759714      1307862
```

# DataFrame

- Data Frame이란 Column과 Row로 이루어진 Table(표)
- 분석에 가장 적합한 데이터 형식 중 하나로, 엑셀 형식과 동일하다고 생각하면 이해하기 쉬움

In [3]: credit\_data.head()

Out [3]:

	Creditability	Account.Balance	Duration.of.Credit..month.	Payment.Status.of.Previous.Credit	Purpose	Credit.Amount	Value.Savings.Stocks	Length.of.current
0	1	1	18	4	2	1049	1	
1	1	1	9	4	0	2799	1	
2	1	2	12	2	9	841	2	
3	1	1	12	4	0	2122	1	
4	1	1	12	4	0	2171	1	

Column

Row

Python Code

```
df = pd.DataFrame(  
    {  
        'column_1': [1, 2, 3, 4],  
        'column_2': ['a', 'b', 'c', 'd'],  
        'column_3': [True, True, False, False]  
    })
```



	column_1	column_2	column_3
0	1	a	True
1	2	b	True
2	3	c	False
3	4	d	False

# Subset

- 데이터프레임에서 특정 Row 혹은 특정 Column을 가져오는 것

강의코드	강의명	중간고사 점수	기말고사 점수	학점
class_1	국어	100	100	A+
class_2	수학	70	80	B+
class_3	영어	50	40	C+
...	...	...	...	...
class_n	통계	20	30	C0



강의코드	강의명	중간고사 점수	기말고사 점수	학점
class_2	수학	70	80	B+
...	...	...	...	B+
...	...	...	...	B+

강의코드	강의명	중간고사 점수	기말고사 점수	학점
class_1	국어	100	100	A+
class_2	수학	70	80	B+
class_3	영어	50	40	C+
...	...	...	...	...
class_n	통계	20	30	C0



강의명	학점
국어	A+
수학	B+
영어	C+
...	...
통계	C0

# Subset

- 다음과 같은 수식을 활용하여 원하는 Row를 가져올 수 있음

Row Subset을 위한 Python Code

```
df[df.total_score == 'B+']
```



	class_code	class_name	score_1	score_2	total_score
1	class_2	math	70	80	B+
3	class_4	computer	65	95	B+
4	class_5	chinese	100	50	B+

수식	의미
a < b	Less then
a > b	Greater then
a <= b	Less then or Equal
a >= b	Greater then or Equal
a == b	Equal
a != b	Not Equal
a>5 & a<10	and
a>10   b>10	or

- Column의 경우 원하는 Column 명을 입력하여 가져올 수 있음

Column Subset을 위한 Python Code

```
df[['class_name', 'total_score']]
```



	class_name	total_score
0	korean	A+
1	math	B+
2	english	C+
3	computer	B+
4	chinese	B+
5	stat	C0
6	music	B0
7	art	B0
8	python	A+
9	java	B0

# Column & Row 추가

- 데이터프레임에 새로운 Row 혹은 Column을 생성하는 것

Column 추가

강의코드	강의명	중간고사 점수	기말고사 점수	학점
class_1	국어	100	100	A+
class_2	수학	70	80	B+
class_3	영어	50	40	C+
...	...	...	...	...
class_n	통계	20	30	C0
class_x	체육	35	85	C+



Row 추가

강의코드	강의명	중간고사 점수	기말고사 점수	학점
class_1	국어	100	100	A+
class_2	수학	70	80	B+
class_3	영어	50	40	C+
...	...	...	...	...
class_n	통계	20	30	C0
class_x	체육	35	85	C+





# Column & Row 추가

- Column과 Column간의 연산(+, -, / \*) 및 복잡한 수식을 활용하여 새로운 Column 생성 가능

Column 생성을 위한 Python Code

```
df['score_sum'] = df['score_1'] + df['score_2']
```



	class_code	class_name	score_1	score_2	total_score	score_sum
0	class_1	korean	100	100	A+	200
1	class_2	math	70	80	B+	150
2	class_3	english	50	40	C+	90
3	class_4	computer	65	95	B+	160
4	class_5	chinese	100	50	B+	150
5	class_6	stat	20	30	C0	50

- Row 생성 시, 각 Column에 맞는 데이터를 입력하여 추가

Row 생성을 위한 Python Code

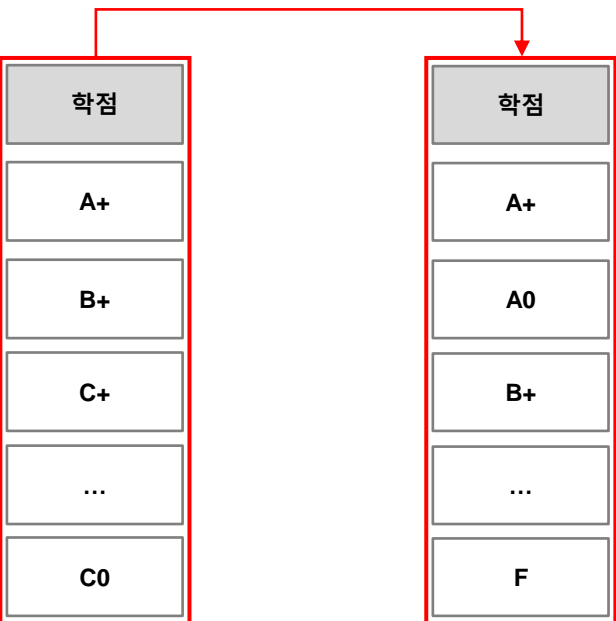
```
df= df.append({  
    'class_code': 'class_11',  
    'class_name': 'kotlin',  
    'score_1': 65,  
    'score_2': 55,  
    'total_score': 'B+'  
},  
ignore_index=True)
```



	class_code	class_name	score_1	score_2	total_score
0	class_1	korean	100	100	A+
1	class_2	math	70	80	B+
2	class_3	english	50	40	C+
3	class_4	computer	65	95	B+
4	class_5	chinese	100	50	B+
5	class_6	stat	20	30	C0
6	class_7	music	60	70	B0
7	class_8	art	90	45	B0
8	class_9	python	85	95	A+
9	class_10	java	90	40	B0
10	class_11	kotlin	65	55	B+

# Group by

- 특정 Column으로 데이터프레임을 묶어(group) 통계량을 구하는 것



강의코드	강의명	중간고사 점수	기말고사 점수	학점
class_1	국어	100	100	A+
class_2	수학	70	80	B+
class_3	영어	50	40	C+
...	...	...	...	...
class_n	통계	20	30	C0

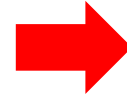
학점	평균 중간고사 점수	평균 기말고사 점수
A+	95	95
A0	90	85
B+	85	80
...	...	...
F	10	20

# Group by

- Group by 이용하여 산출하는 대표적통계량으로는 count(수), sum(합), mean(평균), min(최솟값), max(최대값) 등이 있음

## Python Code

```
df.groupby('total_score')['score_1'].mean()
```



```
total_score
A+    92.5
B+    75.0
B0    80.0
C+    50.0
C0    20.0
Name: score_1, dtype: float64
```

- Group by 결과를 데이터프레임 형식으로 변경가능

## Python Code

```
output = df.groupby('total_score')['score_1'].mean()
pd.DataFrame(output)
```



score_1	
total_score	
A+	92.5
B+	75.0
B0	80.0
C+	50.0
C0	20.0

# Merge

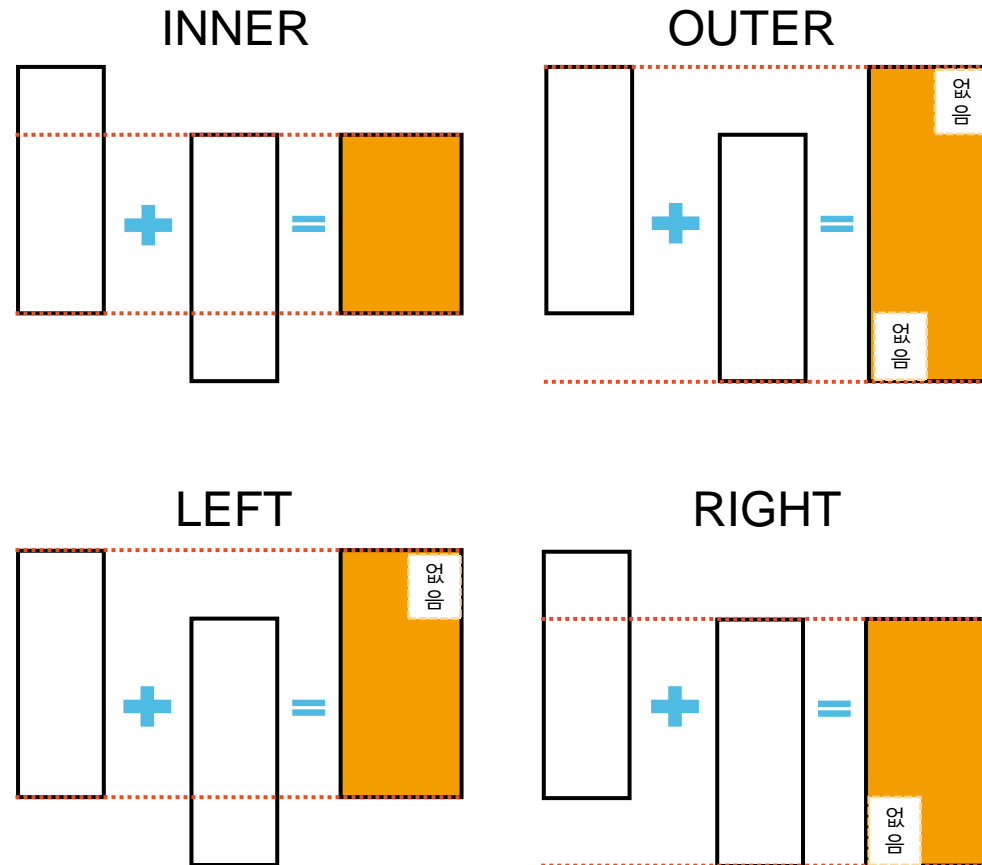
- 두 개의 데이터 프레임을 공통된 Column을 기준으로 합치는 것. SQL의 Join과 동일한 기

능



# Merge

- Inner, Outer, Left, Right 네 가지의 방법이 있으며, 데이터를 처리하고자 하는 방향에 맞게 활용



# Inner, Outer Merge

기준 Column

df\_1

	class_code	class_name	score_1	score_2	total_score
0	class_1	korean	100	100	A+
1	class_2	math	70	80	B+
2	class_3	english	50	40	C+
3	class_4	computer	65	95	B+
4	class_5	chinese	100	50	B+
5	class_6	stat	20	30	C0
6	class_7	music	60	70	B0
7	class_8	art	90	45	B0
8	class_9	python	85	95	A+
9	class_10	java	90	40	B0
10	class_11	kotlin	65	55	B+

df\_2

	class_code	professor	total_student
0	class_1	a	35
1	class_2	b	25
2	class_3	c	21
3	class_4	d	33
4	class_6	e	29
5	class_7	a	46
6	class_8	b	74
7	class_9	c	16
8	class_10	d	37
9	class_11	e	13
10	class_12	a	75

누락된 데이터

Inner merge Python Code

```
pd.merge(df_1, df_2, how='inner', on='class_code')
```

	class_code	class_name	score_1	score_2	total_score	professor	total_student
0	class_1	korean	100	100	A+	a	35
1	class_2	math	70	80	B+	b	25
2	class_3	english	50	40	C+	c	21
3	class_4	computer	65	95	B+	d	33
4	class_6	stat	20	30	C0	e	29
5	class_7	music	60	70	B0	a	46
6	class_8	art	90	45	B0	b	74
7	class_9	python	85	95	A+	c	16
8	class_10	java	90	40	B0	d	37
9	class_11	kotlin	65	55	B+	e	13

Outer merge Python Code

```
pd.merge(df_1, df_2, how='outer', on='class_code')
```

	class_code	class_name	score_1	score_2	total_score	professor	total_student
0	class_1	korean	100.0	100.0	A+	a	35.0
1	class_2	math	70.0	80.0	B+	b	25.0
2	class_3	english	50.0	40.0	C+	c	21.0
3	class_4	computer	65.0	95.0	B+	d	33.0
4	class_5	chinese	100.0	50.0	B+	NaN	NaN
5	class_6	stat	20.0	30.0	C0	e	29.0
6	class_7	music	60.0	70.0	B0	a	46.0
7	class_8	art	90.0	45.0	B0	b	74.0
8	class_9	python	85.0	95.0	A+	c	16.0
9	class_10	java	90.0	40.0	B0	d	37.0
10	class_11	kotlin	65.0	55.0	B+	e	13.0
11	class_12	NaN	NaN	NaN	NaN	a	75.0

# Left, Right Merge

df\_1

	class_code	class_name	score_1	score_2	total_score
0	class_1	korean	100	100	A+
1	class_2	math	70	80	B+
2	class_3	english	50	40	C+
3	class_4	computer	65	95	B+
4	class_5	chinese	100	50	B+
5	class_6	stat	20	30	C0
6	class_7	music	60	70	B0
7	class_8	art	90	45	B0
8	class_9	python	85	95	A+
9	class_10	java	90	40	B0
10	class_11	kotlin	65	55	B+

기준  
Column

df\_2

	class_code	professor	total_student
0	class_1	a	35
1	class_2	b	25
2	class_3	c	21
3	class_4	d	33
4	class_6	e	29
5	class_7	a	46
6	class_8	b	74
7	class_9	c	16
8	class_10	d	37
9	class_11	e	13
10	class_12	a	75

누락된 데이터

Left merge Python Code

```
pd.merge(df_1, df_2, how='left', on='class_code')
```

	class_code	class_name	score_1	score_2	total_score	professor	total_student
0	class_1	korean	100	100	A+	a	35.0
1	class_2	math	70	80	B+	b	25.0
2	class_3	english	50	40	C+	c	21.0
3	class_4	computer	65	95	B+	d	33.0
4	class_5	chinese	100	50	B+	NaN	NaN
5	class_6	stat	20	30	C0	e	29.0
6	class_7	music	60	70	B0	a	46.0
7	class_8	art	90	45	B0	b	74.0
8	class_9	python	85	95	A+	c	16.0
9	class_10	java	90	40	B0	d	37.0
10	class_11	kotlin	65	55	B+	e	13.0

Right merge Python Code

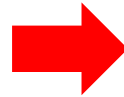
```
pd.merge(df_1, df_2, how='right', on='class_code')
```

	class_code	class_name	score_1	score_2	total_score	professor	total_student
0	class_1	korean	100.0	100.0	A+	a	35
1	class_2	math	70.0	80.0	B+	b	25
2	class_3	english	50.0	40.0	C+	c	21
3	class_4	computer	65.0	95.0	B+	d	33
4	class_6	stat	20.0	30.0	C0	e	29
5	class_7	music	60.0	70.0	B0	a	46
6	class_8	art	90.0	45.0	B0	b	74
7	class_9	python	85.0	95.0	A+	c	16
8	class_10	java	90.0	40.0	B0	d	37
9	class_11	kotlin	65.0	55.0	B+	e	13
10	class_12	NaN	NaN	NaN	NaN	a	75

# Sorting

- 데이터프레임을 특정 Column 기준으로 정렬하는 것
- 여러 개의 컬럼을 선택할 수 있으며 오름차순/내림차순 정의

강의코드	강의명	중간고사 점수	기말고사 점수	학점
class_1	국어	100	100	A+
class_2	수학	70	80	B+
class_3	영어	50	40	C+
...	...	...	...	...
class_n	통계	20	30	C0



강의코드	강의명	중간고사 점수	기말고사 점수	학점
...	...	0	...	F
...	...	10	...	C0
...	...	20	...	C+
...	...	...	...	...
...	...	100	...	A+



# Sorting

- Pandas에서 기본값은 오름차순(0 -> 100, a -> z)

Python Code

```
df.sort_values('score_1')
```

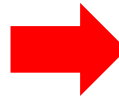


	class_code	class_name	score_1	score_2	total_score
5	class_6	stat	20	30	C0
2	class_3	english	50	40	C+
6	class_7	music	60	70	B0
3	class_4	computer	65	95	B+
1	class_2	math	70	80	B+
8	class_9	python	85	95	A+
7	class_8	art	90	45	B0
9	class_10	java	90	40	B0
0	class_1	korean	100	100	A+
4	class_5	chinese	100	50	B+

- 내림차순(100 -> 0, z->a)의 경우 **ascending=False** 을 선언

Python Code

```
df.sort_values('class_name', ascending=False)
```

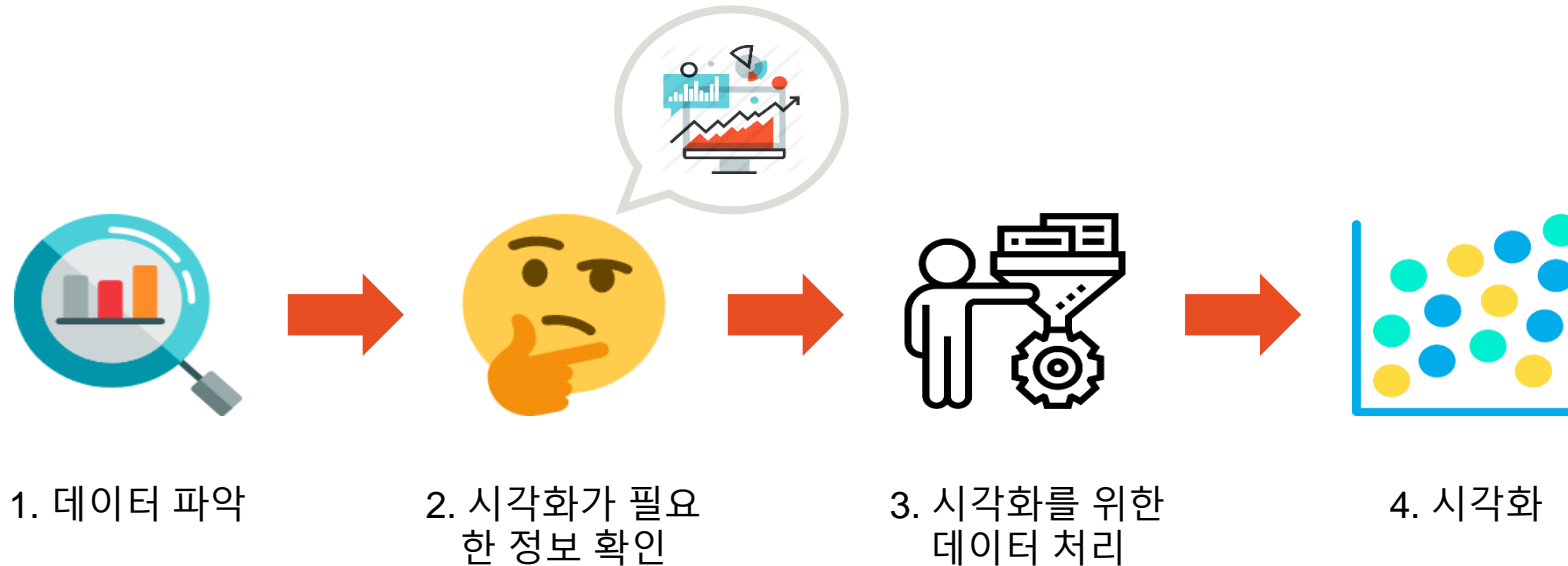


	class_code	class_name	score_1	score_2	total_score
5	class_6	stat	20	30	C0
8	class_9	python	85	95	A+
6	class_7	music	60	70	B0
1	class_2	math	70	80	B+
0	class_1	korean	100	100	A+
9	class_10	java	90	40	B0
2	class_3	english	50	40	C+
3	class_4	computer	65	95	B+
4	class_5	chinese	100	50	B+
7	class_8	art	90	45	B0

# Matplotlib를 활용한 시각화

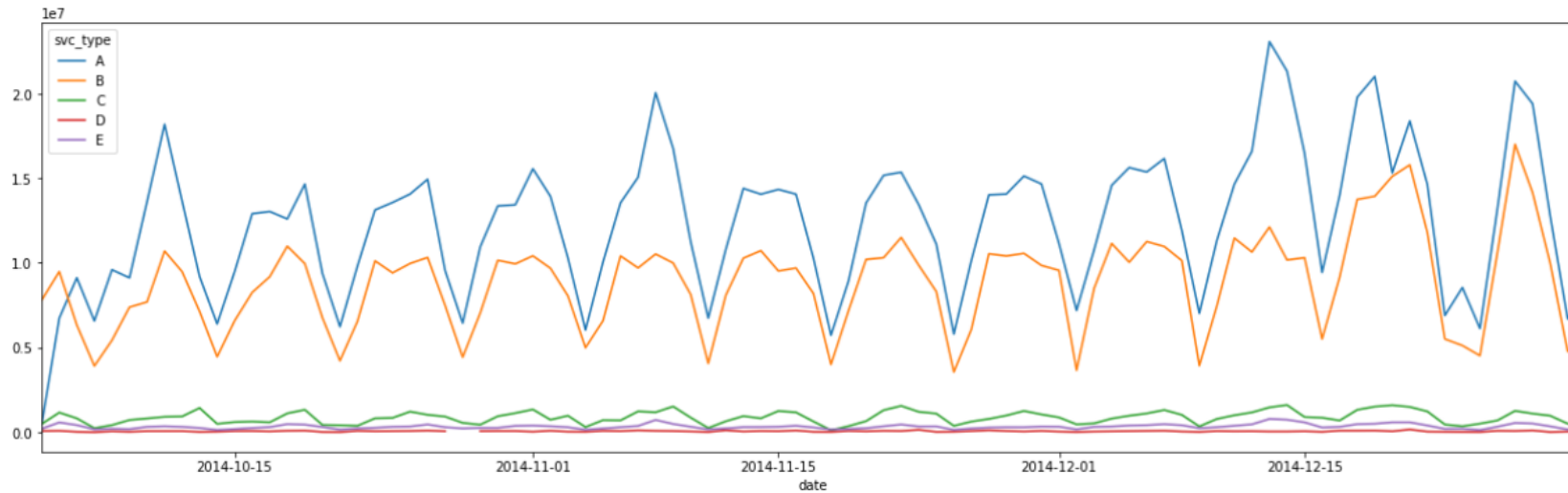
# 시각화에 필요한 과정

1. 데이터 파악: 분석 과정 중 혹은 최종 분석 결과를 파악하여 데이터의 상태를 확인
2. 시각화가 필요한 정보 확인: 데이터를 파악하면서 시각화가 필요하다고 판단되는 정보 확인
3. 시각화를 위한 데이터 처리: 시각화가 필요하다고 판단된 데이터는 시각화 형식(line, bar, pie ...)을 선정하고, 선정된 시각화에 알맞은 형태로 데이터를 처리(재구조화)
4. 시각화



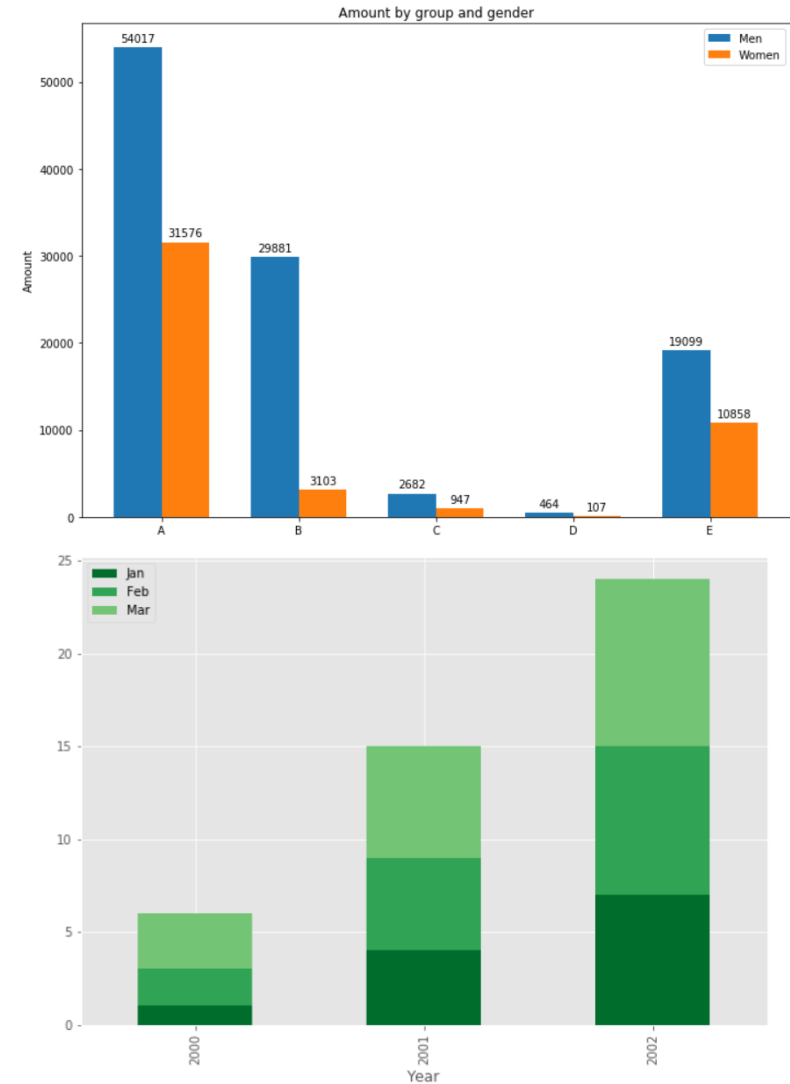
# Line Chart

- 시간이 경과함에 따른 추세(trend)를 파악하는 시각화
- 동일시점 동안 다양한 카테고리들의 변화 추세를 파악



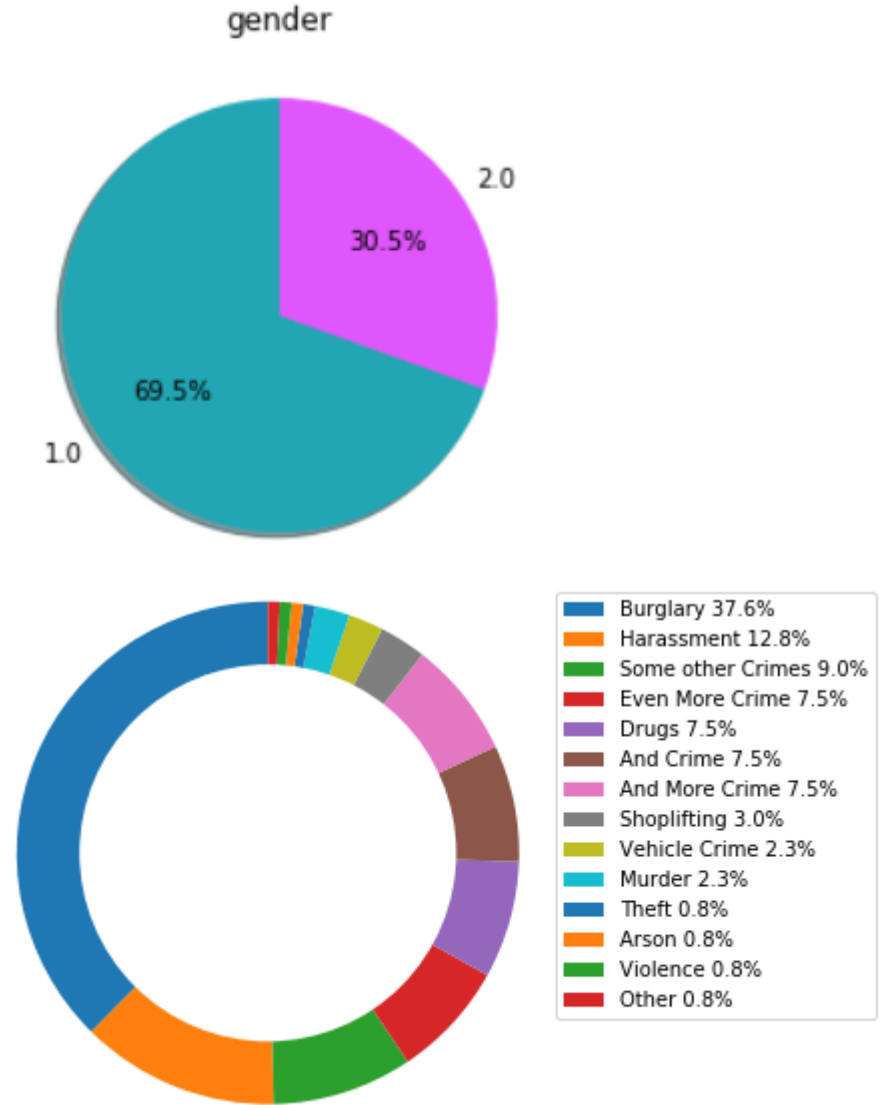
# Bar Chart

- 카테고리별로 순위를 매기거나 비교하기 적합한 시각화
- 시간의 흐름에 따라 변화하는 데이터를 표시할 수도 있으나, 너무 많은 시간 정보를 포함하기에는 부적절



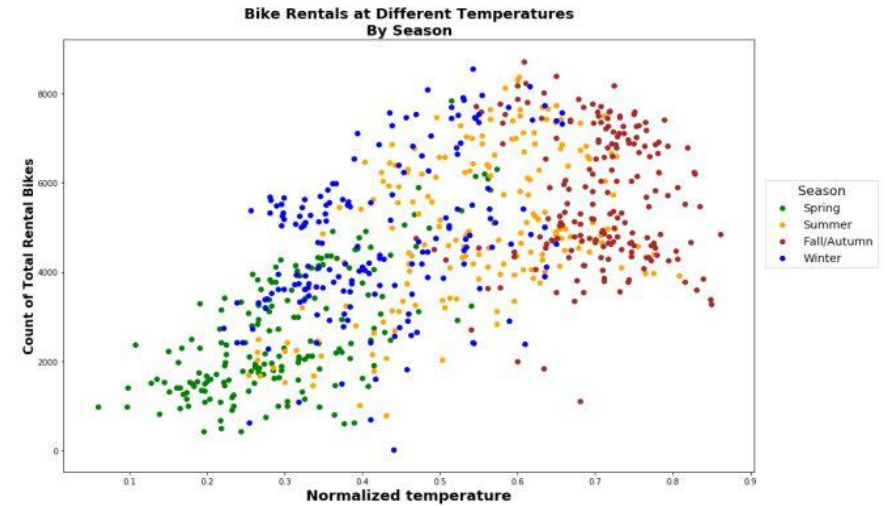
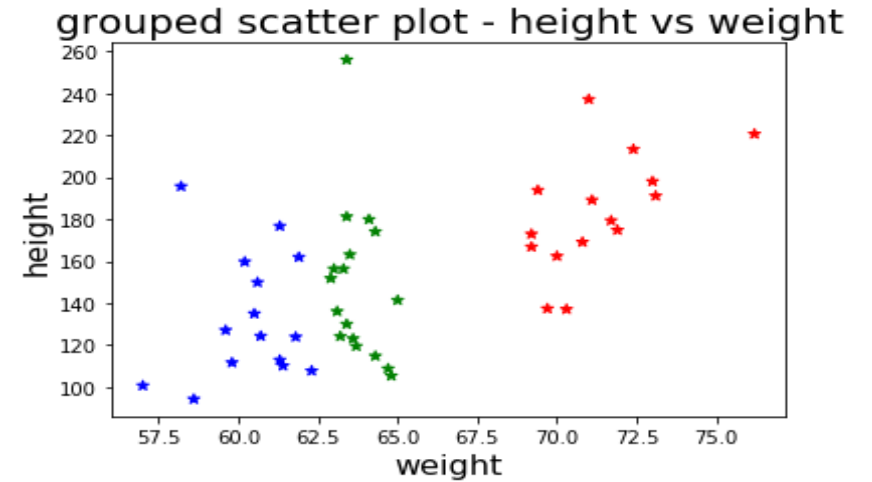
# Pie Chart

- 비율(Percent) 자료를 비교하기 위한 시각화
- 모든 카테고리의 합이 100%가 되었을 때 가장 이해하기도 활용하기도 쉬움
- 너무 많은 카테고리를 비교하기에는 적절하지 않으므로 카테고리 수준을 고려해야 함



# Scatter Plot

- 두 가지 대상 간의 관계를 파악하기 위한 시각화
- 증감의 추세를 파악하기도 하며 특정 영역으로 데이터를 구분 짓기 위한 자료로 활용



# 제주도 인구통계자료를 활용한 Data Handling 및 시각화 실습