

Domain DX Enabler (Level 3)

- 통계적추정과 검정 -

학습목표

- ❖ 과정명: Domain DX Enabler (Level 3) – 통계적 검정과 추정
- ❖ 대 상 : Level 2 이수자
- ❖ 학습목표
 - (1) 확증적 데이터분석(CDA)의 의미 및 다양한 분석방법을 이해한다.
 - (2) Python의 기초 사용법 및 데이터 처리방법을 이해한다.
 - (3) 범주형 및 수치형 자료를 분석하는 방법을 이해한다.
 - (4) t-test와 ANOVA를 이용한 차이검정을 이해한다.
 - (5) Regression을 이용한 관계검정을 이해한다.
 - (6) 통계적 분석방법과 기계학습 분석방법의 차이를 이해한다.

강의내용

일정	Module	Lesson	강의내용
1일차	Python 기초	Python 기초 사용법	Python 기초 실습, 연습문제 실습
		Python 데이터 처리	넘파이, 판다스, 그래프 그리기, 연습문제 실습
	탐색적 자료분석	확증적 데이터분석(CDA)의 이해	탐색적 단계와 학인적 데이터 분석의 이해
		범주형 자료 분석	범주형 자료 요약 및 그래프 그리기, 연습문제 실습
		수치형 자료 분석	수치형 자료 요약(평균, 표준편차) 및 그래프 그리기, 연습문제 실습
2일차	확률분포와 통계적 검정	표본추출과 표본오차	확률표본추출 방법, 표본오차
		확률분포에 대한 이해	이항분포, 포아송분포, 지수분포, 정규분포, t-분포, F분포, 카이제곱분포
		추정과 가설검정	평균, 비율, 산포의 구간 추정, 가설검정 절차, 검정력
	t-test	One Sample T-test	평균, 비율, 산포의 차이검정, 산포의 차이검정, 비모수 통계, 연습문제 실습
		Independent Sample T-test	두집단 평균, 비율, 산포의 차이검정, 비모수 통계, 연습문제 실습
		Paired Sample T-test	반복측정 평균 차이검정, 비모수 통계, 연습문제 실습
		Equivalence test	One sample, independent, paired 동등성검정
		Sample size	One sample, independent, paired, proportion 샘플 수
3일차	ANOVA	One Way ANOVA	One Way ANOVA, 비모수 통계, 연습문제 실습
		Repeated Measures ANOVA	Repeated Measures ANOVA, 비모수 통계, 연습문제 실습
		Two Way ANOVA	Two Way ANOVA, 연습문제 실습
		Two Way Repeated Measures ANOVA	Two Way Repeated Measures ANOVA, 연습문제 실습
	Regression	Chi-Square test	Chi-Square test, 동질성, 독립성 검정, 연습문제 실습
		AB test	AB test에 대한 이해
		Regression	Correlation, Regression(예측), Regression(설명), 연습문제 실습
		Logistic Regression	Logistic Regression, 연습문제 실습
		통계적 분석과 기계학습 분석 방법 비교	통계적 분석과 기계학습 분석방법의 차이, 기계학습 분석 절차

Module1

Python 기초



◆ 학습목표

Python에 대한 기초 및 데이터 처리 방법을 학습하고 그래프 그리는 방법을 학습한다.

-
- I. Python 기초 사용법
 - II. Python 데이터 처리
 - III. Python 그래프 그리기
-

I. Python 기초 사용법

Python이란

Python

❖ 개발자

- 네델란드 귀도 반 로섬(Guido van Rossum)
- 1989년 발표

❖ 버전

- 파이썬 2: 2000~
- 파이썬 3: 2008~
 - 2.0과 호환되지 않음
 - 2008: Pandas 추가
 - 2010: sklearn(SCIKIT-Learn) 추가

❖ Jupyter Notebook

- Python을 좀 더 편하게 사용하기 위해 개발된 통합 개발환경 프로그램(IDE, Integrated Development Environment)
- IDE: 공통된 개발자 툴을 하나의 그래픽 사용자 인터페이스(GUI)로 결합하는 애플리케이션을 구축하기 위한 소프트웨어

출처: <https://ko.wikipedia.org/wiki/파이썬>

Python: Engine



Jupyter Notebook: Dashboard



Analogy of difference between Python and IDE Tools

출처: <https://moderndive.com/1-getting-started.html>

Python

❖ python에서 필요한 라이브러리

- 아나콘다(Anaconda): 배포판
- 주피터 노트북: IDE 프로그램 코드를 브라우저에서 실행
- Colab: 구글에서 만든 클라우드 기반 무료 주피터 노트북
- Numpy(넘파이): 과학 계산을 위한 패키지, 다차원 배열, 선형 대수 연산 등
- Matplotlib (맷플롯립): 그래프 패키지, 선 그래프, 히스토그램 등
- Pandas (판다스)
 - 데이터 처리와 분석, R의 data.frame과 유사한 DataFrame 사용
 - 엑셀 파일, CSV 파일 같은 다양한 파일 처리
- 통계 패키지: pingouin, scipy.stats, statsmodels
- 기계학습 패키지: Sklearn(싸이킷런), Tensorflow(텐서플로), Keras(케라스)

Python

❖ 특징

- 고급프로그래밍 언어
- 컴파일, 인터프리터, 웹 프로그래밍 지원
- 객체지향 프로그램

❖ 문법 특징

- 들여쓰기를 이용한 블록 구조
- 들여쓰기 4칸: 동일해야지만 인식
- C : {} 이용

파이썬

```
def factorial(x):  
    if x == 0:  
        return 1  
    else:  
        return x * factorial(x - 1)
```

C

```
int factorial(int x) {  
    if(x == 0) {  
        return 1;  
    } else {  
        return x * factorial(x - 1);  
    }  
}
```

출처: <https://ko.wikipedia.org/wiki/파이썬>

R과 Python 비교

	Python	R
개발자	<ul style="list-style-type: none"> Guido van Rossum 	<ul style="list-style-type: none"> Robert Gentleman Ross Ihaka
활용	<ul style="list-style-type: none"> 일반 프로그래밍 + 기계학습 + 딥러닝 	<ul style="list-style-type: none"> 통계분석 + 기계학습
IDE	<ul style="list-style-type: none"> 아나콘다(Anaconda) 주피터 노트북 Colab 	<ul style="list-style-type: none"> Rstudio
기계학습 패키지	<ul style="list-style-type: none"> Sklearn Tensorflow/Keras 	<ul style="list-style-type: none"> Tidymodel

Python 기초 사용법

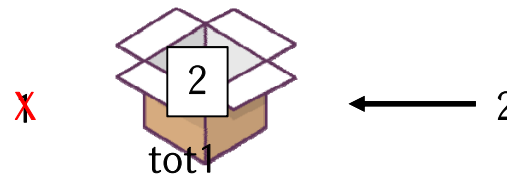
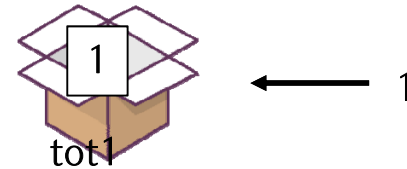
프로그래밍의 기본원칙

- ❖ 변수선언
- ❖ 데이터형
 - 숫자, 문자, 자료
 - 자료구조
- ❖ 연산자
 - 산술연산자, 증가감소연산자, 대입연산자
 - 비교연산자, 논리연산자, 연결연산자
- ❖ 제어문
 - If
- ❖ 반복문
 - for
 - while
- ❖ 함수

변수명

❖ 변수선언

- 변수명 = 값



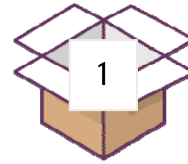
❖ 변수명 규칙

- # 제외
- 첫 문자를 영문자로 시작해서 영문 대문자(A~Z)와 소문자(a~z), 숫자(0~9), 밑줄(_)을 사용하여 작성
- 특수기호(!, @, # ...)는 사용할 수 없음
- 대/소문자를 구분
- 변수의 값이 무엇을 나타내는지 쉽게 표기

데이터형

❖ 기본 자료형

- 정수형, 실수형, 부울형, 문자형



1

숫자형

❖ 집합형 자료형

- 리스트형, 튜플형, 사전형



"남자"

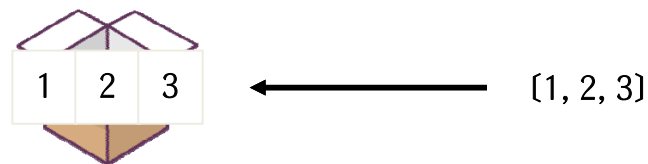
문자형

❖ type() : 자료형 확인

종류	종류	문법 예
정수형	int	42, 0, -41
실수형	float	3.141
불울형	bool	True False
문자형	str	'홍길동' "홍길동"

❖ Python 데이터형

	형식	데이터 수정, 삭제	index 이용
리스트(list)	num = [1, 2, 3]	O	O
튜플(tuple)	num = (1, 2, 3)	X	O
세트(set)	num = {1, 2, 3}	O	X
딕셔너리(dictionary)	num = {a: 1, 2, 3}	O	X



Index	num[0]	num[1]	num[2]	num[3]	num[4]
data	1	2	3	4	5

산술 및 관계 연산자

❖ 산술연산자

산술연산자	설명	예제	사례
+	더하기 연산	$A + B$	$5 + 10 = 15$
-	빼기 연산	$A - B$	$5 - 10 = -5$
*	곱하기 연산	$A * B$	$5 * 10 = 50$
/	나누기 연산	A / B	$10 / 5 = 2$
**	승수	$A ** B$	$5 ** 2 = 25$
%	나머지	$A \% B$	$13 \% 5 = 3$

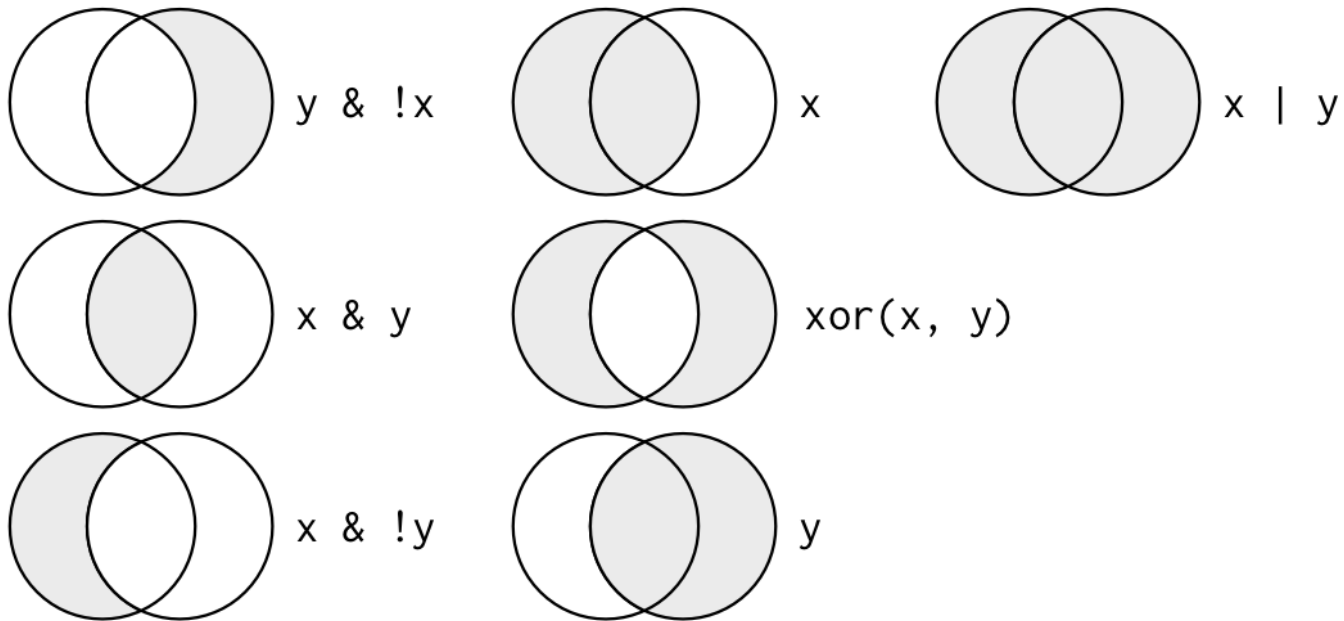
❖ 관계연산자

관계 연산자	설명	예제	사례
==	A, B가 같다.	변수 == B	Name == "홍길동"
!=	A, B가 같지 않다.	변수 != B	Name != "홍길동"
<	A가 B보다 작다.	변수 < B	Weight < 40
<=	A가 B보다 작거나 같다.	변수 <= B	Weight <= 40
>	A가 B보다 크다.	변수 > B	Weight > 40
>=	A가 B보다 크거나 같다.	변수 >= B	Weight >= 40

논리 연산자

❖ 논리연산자

관계 연산자	설명	예제
&	A와 B가 모두 참이면 참	A & B
	A와 B가 중 하나만 참이면 참	A B
!	부정	!A



❖ 문자열 포맷코드

코드	설명
%s	문자열(String)
%d	정수(Integer)
%f	부동소수(floating-point) Ex)0.3f 소수점 3자리까지 표기

1. Python 기초

▼ 00.Python 기초 및 데이터 처리

▼ 1.Python 기초

▼ 1.1 변수선언 및 데이터 형

```
✓ 0초 [1] tot1 = 5  
      tot2 = 3.14  
      name = '홍길동'
```

```
✓ 0초 [2] print(tot1)  
      name  
  
      5  
      '홍길동'
```

```
✓ 0초 [3] type(tot1)  
  
      int
```

```
✓ 0초 [4] type(tot2)  
  
      float
```

```
✓ 0초 [5] type(name)  
  
      str
```

```
✓ 0초 [6] # 여러변수를 동시에 입력  
      start, end, step = 1, 10, 2
```

✓ 2초 오후 3:06에 완료됨



1. Python 기초

```

str

[6] # 여러변수를 동시에 입력
    start, end, step = 1, 10, 2

[7] print(start, end, step)

1 10 2

▼ 1.2 데이터 포매팅

[8] print('이름은 ' + name + "입니다")

이름은 홍길동입니다

[9] # 문자형 형태로 변환 필요
    print('tot1: ' + tot1)

-----
TypeError                                Traceback (most recent call last)
<ipython-input-9-e107612258ca> in <cell line: 2>()
      1 # 문자형 형태로 변환 필요
----> 2 print('tot1: ' + tot1)

TypeError: can only concatenate str (not "int") to str

[10] print('tot1: ' + str(tot1))

tot1: 5

[11] # 기본 출력
      # ,로 연결하면 변환없이 출력 가능
      print('tot1:', tot1) # 1칸 띄어쓰기 처리
      print('이름은', name, '입니다.')
  
```

✓ 2초 오후 3:06에 완료됨

1. Python 기초

```
✓ 1초 [11] # 기본 출력
# ,로 연결하면 변환없이 출력 가능
print('tot1:', tot1) # 1칸 띄어쓰기 처리
print('이름은', name, '입니다.')
```

```
tot1: 5
이름은 홍길동 입니다.
```

```
✓ 0초 [12] # 문자열 포맷 이용
print('tot1: %d' % tot1)
print('tot2: %0.2f' % tot2)
print('이름: %s' % name)
print('tot1의 값: %0.2f, 이름: %s' % (tot1, name))
```

```
tot1: 5
tot2: 3.14
이름: 홍길동
tot1의 값: 5.00, 이름: 홍길동
```

```
✓ 0초 [13] # format 이용
print('tot1의 값: {}'.format(tot1))
print('tot2의 값: {:.2f}, 이름: {}'.format(tot2, name))
```

```
tot1의 값: 5
tot2의 값: 3.14, 이름: 홍길동
```

2. 데이터구조

2.1 리스트, 튜플, 집합, 사전

- 리스트(list): 순서를 가지는 객체 집합
- 튜플(tuple): 자료 추가 X, 속도 빠름
- 집합(set): 중복 X, index X, 자료추가 O
- 사전(dictionary): key, value 형태로 저장 - index(X)

✓ 2초 오후 3:06에 완료됨

연습문제

연습문제1

❖ 1. 변수와 데이터 포매팅을 이용해 본인을 소개해 보세요

- 이름(name)
- 나이(age)
- 소속(dep)
- 직위(pos)
- 취미(hob)

❖ 결과화면

안녕하세요.
저의 이름은 이상철 이라고 합니다.
간단하게 제 소개를 하고자 합니다.

이름: 이상철
나이: 54
소속: 강서대학교 빅데이터경영학과
직위: 교수
취미: 코딩

II. 데이터 처리

2.데이터구조

2.데이터구조

2.1 리스트, 튜플, 집합, 사전

- 리스트(list): 순서를 가지는 객체 집합
- 튜플(tuple): 자료 추가 X, 속도 빠름
- 집합(set): 중복 X, index X, 자료추가 O)
- 사전(dictionary): key, value 형태로 저장 - index(X)

```
[14] # 리스트
num = [1, 2, 3, 4]
num

[1, 2, 3, 4]
```

```
[15] # 튜플(tuple)
num = (1, 2, 3, 4)
num

(1, 2, 3, 4)
```

```
[16] # 인덱싱(0부터 시작)
num[0]

1
```

```
[17] num[1:3]

(2, 3)
```

```
[18] # 집합(set)
num = {1, 2, 3, 4}
num
```

✓ 2초 오후 3:06에 완료됨

2.데이터구조

```

(2, 3)

✓ [18] # 집합(set)
0초 num = {1, 2, 3, 4}
num

{1, 2, 3, 4}

✓ [19] # 사전
0초 student_dc = {1:"김길동", 10:"박길동"}
student_dc

{1: '김길동', 10: '박길동'}
```

▼ 2.2 넘파이 배열

```

✓ [20] num1 = [1, 2, 3, 4]
0초 num2 = [5, 6, 7, 8]
num = [num1, num2]
num

[[1, 2, 3, 4], [5, 6, 7, 8]]

✓ [21] # 계산이 안됨
0초 num1 + num2

[1, 2, 3, 4, 5, 6, 7, 8]

✓ [22] import numpy as np
0초

num1 = np.array(num1)
num2 = np.array(num2)
num1

array([1, 2, 3, 4])
```

✓ 2초 오후 3:06에 완료됨

2.데이터구조

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
[22] import numpy as np

num1 = np.array(num1)
num2 = np.array(num2)
num1

array([1, 2, 3, 4])
```

```
[23] num = num1 + num2
num

array([ 6,  8, 10, 12])
```

```
[24] # 열기준으로 합계
num.sum(axis=0)

36
```

```
[25] num1.sum()

10
```

```
[26] num = np.array([num1, num2])
num

array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

```
[27] # 배열의 행, 열 갯수 확인
num.shape

(2, 4)
```

3.데이터 처리(판다스 배열)

2초 오후 3:06에 완료됨

- https://pandas.pydata.org/docs/user_guide/index.html

```
[28] #구글 드라이브 이용시
      #from google.colab import drive
      #drive.mount('/content/drive')
```

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	flight	origin	dest
0	1	9	813.0	-9.0	1014.0	-5.0	EV	4691	EWR	DAY
1	9	4	2031.0	-9.0	2225.0	-34.0	EV	4085	EWR	OMA
2	3	8	2248.0	128.0	134.0	111.0	B6	629	JFK	HOU
3	7	23	743.0	0.0	1106.0	3.0	UA	1668	EWR	SFO
4	3	15	754.0	-5.0	916.0	-20.0	UA	393	EWR	ORD

```
[30] # DataFrame: Series 집합
type(flight_df)
```

✓ 2초 오후 3:06에 완료됨

3.데이터 처리(판다스 배열)

✓ [30] # DataFrame: Series 집합
type(flight_df)

pandas.core.frame.DataFrame

def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype | None=None, copy: bool | None=None) -> None

</usr/local/lib/python3.10/dist-packages/pandas/core/frame.py>

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary

✓ [31] # pd.Series: 한개 열로 구성
type(flight_df["month"])

pandas.core.series.Series

def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool=False, fastpath: bool=False) -> None

</usr/local/lib/python3.10/dist-packages/pandas/core/series.py>

One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object supports both integer- and label-based indexing and provides a host of methods for performing operations involving the index. Statistical

✓ [32] # pd.Series: 한개 열로 구성
type(flight_df["month"])

pandas.core.series.Series

def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool=False, fastpath: bool=False) -> None

and index is None, then the keys in the data are used as the index. If the index is not None, the resulting Series is reindexed with the index values.

dtype : str, numpy.dtype, or ExtensionDtype, optional

Data type for the output Series. If not specified, this will be inferred from `data`.

See the :ref:`user guide <basics.dtypes>` for more usages.

name : str, optional

✓ 2초 오후 3:06에 완료됨

3.데이터 처리(판다스 배열)

LGE Internal Use Only

```
3.2 data type

[33] flight_df.shape

(10000, 10)

[34] flight_df.dtypes

month      int64
day        int64
dep_time   float64
dep_delay  float64
arr_time   float64
arr_delay  float64
carrier    object
flight     int64
origin     object
dest       object
dtype: object

[35] flight_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 10 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   month      10000 non-null  int64
1   day        10000 non-null  int64
2   dep_time   9748 non-null   float64
3   dep_delay  9748 non-null   float64
4   arr_time   9736 non-null   float64
5   arr_delay  9715 non-null   float64
6   carrier    10000 non-null  object
7   flight     10000 non-null  int64
8   origin     10000 non-null  object
9   dest       10000 non-null  object
dtypes: float64(4), int64(3), object(3)
memory usage: 781.4+ KB

✓ 2초  오후 3:06에 완료됨
```

3.데이터 처리(판다스 배열)

LGE Internal Use Only

```
3.3 data 속성

[36] # array형식으로 추출
flight_df.columns

Index(['month', 'day', 'dep_time', 'dep_delay', 'arr_time', 'arr_delay',
      'carrier', 'flight', 'origin', 'dest'],
      dtype='object')

[37] # 변수명만 추출
flight_df.columns.tolist()

['month',
 'day',
 'dep_time',
 'dep_delay',
 'arr_time',
 'arr_delay',
 'carrier',
 'flight',
 'origin',
 'dest']

[38] flight_df.values

array([[1, 9, 813.0, ..., 4691, 'EWR', 'DAY'],
      [9, 4, 2031.0, ..., 4085, 'EWR', 'OMA'],
      [3, 8, 2248.0, ..., 629, 'JFK', 'HOU'],
      ...,
      [5, 9, 1359.0, ..., 209, 'JFK', 'LGB'],
      [3, 6, nan, ..., 5325, 'LGA', 'CHO'],
      [4, 29, 1644.0, ..., 586, 'EWR', 'CLE']], dtype=object)

3.4 column(열), row(행) 추출

[39] flight_df.head(10)
```

✓ 2초 오후 3:06에 완료됨



3.데이터 처리(판다스 배열)

LGE Internal Use Only

```
[40] 9999 4
      Name: month, Length: 10000, dtype: int64

[41] # 1개 열 추출
      flight_df["month"]

0      1
1      9
2      3
3      7
4      3
...
9995    8
9996   11
9997    5
9998    3
9999    4
      Name: month, Length: 10000, dtype: int64

[42] # 2개 열 추출
      # loc(컬럼명으로 추출)
      flight_df.loc[:, ['month', 'day']]

   month  day
0      1    9
1      9    4
2      3    8
3      7   23
4      3   15
...     ...
9995    8    3
9996   11   21
9997    5    9
9998    3    6
9999    4   29
```

✓ 2초 오후 3:06에 완료됨

3.데이터 처리(판다스 배열)

✓ 0초 [43] # 여러개 열을 연속으로 추출
iloc(index로 추출)
index= 0으로 시작, 마지막은 +1
flight_df.iloc[:, 0:4]

	month	day	dep_time	dep_delay
0	1	9	813.0	-9.0
1	9	4	2031.0	-9.0
2	3	8	2248.0	128.0
3	7	23	743.0	0.0
4	3	15	754.0	-5.0
...
9995	8	3	1737.0	2.0
9996	11	21	743.0	-2.0
9997	5	9	1359.0	1.0
9998	3	6	NaN	NaN
9999	4	29	1644.0	14.0

10000 rows × 4 columns

✓ 0초 [44] # dataframe에서 행 선택하기
flight_df.iloc[0:5, :]

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	flight	origin	dest
0	1	9	813.0	-9.0	1014.0	-5.0	EV	4691	EWR	DAY
1	9	4	2031.0	-9.0	2225.0	-34.0	EV	4085	EWR	OMA
2	3	8	2248.0	128.0	134.0	111.0	B6	629	JFK	HOU
3	7	23	743.0	0.0	1106.0	3.0	UA	1668	EWR	SFO
4	3	15	754.0	-5.0	916.0	-20.0	UA	393	EWR	ORD

✓ 2초 오후 3:06에 완료됨

3.데이터 처리(판다스 배열)

	4	3	15	754.0	-5.0	916.0	-20.0	UA	393	EWR	ORD	
✓ 1초	[45] # column 삭제											
	flight_df = flight_df.drop('flight', axis=1)											
	flight_df.head()											

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
0	1	9	813.0	-9.0	1014.0	-5.0	EV	EWR	DAY
1	9	4	2031.0	-9.0	2225.0	-34.0	EV	EWR	OMA
2	3	8	2248.0	128.0	134.0	111.0	B6	JFK	HOU
3	7	23	743.0	0.0	1106.0	3.0	UA	EWR	SFO
4	3	15	754.0	-5.0	916.0	-20.0	UA	EWR	ORD

Next steps: [View recommended plots](#)

3.5 filtering

✓ 0초

[46] flight_df[(flight_df.month == 1)].head()

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
0	1	9	813.0	-9.0	1014.0	-5.0	EV	EWR	DAY
14	1	26	555.0	-6.0	931.0	10.0	UA	JFK	LAX
29	1	25	1723.0	-6.0	2111.0	25.0	DL	EWR	SLC
72	1	4	1128.0	-5.0	1304.0	-5.0	EV	EWR	RDU
73	1	9	2019.0	-11.0	2159.0	-7.0	FL	LGA	CAK

✓ 0초

[47] flight_df[(flight_df.month == 1) & (flight_df.day == 31)].head()

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
--	-------	-----	----------	-----------	----------	-----------	---------	--------	------

✓ 2초 오후 3:06에 완료됨

3.데이터 처리(판다스 배열)

```
[47] flight_df[(flight_df.month == 1) & (flight_df.day == 31)].head()
```

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
494	1	31	1124.0	-6.0	1443.0	13.0	DL	JFK	MCO
1070	1	31	804.0	5.0	1026.0	1.0	UA	LGA	DEN
1343	1	31	651.0	17.0	1028.0	53.0	UA	EWR	MIA
1846	1	31	1533.0	-2.0	1843.0	-7.0	AA	LGA	DFW
1959	1	31	1534.0	-6.0	1655.0	-21.0	9E	JFK	ROC

```
[48] flight_df[(flight_df.month == 1) | (flight_df.month == 2)].head() # pd에서는 or |> |
```

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
0	1	9	813.0	-9.0	1014.0	-5.0	EV	EWR	DAY
5	2	22	1723.0	8.0	1830.0	5.0	EV	EWR	DCA
12	2	16	1857.0	-1.0	2201.0	5.0	UA	EWR	PBI
14	1	26	555.0	-6.0	931.0	10.0	UA	JFK	LAX
29	1	25	1723.0	-6.0	2111.0	25.0	DL	EWR	SLC

```
[49] flight_df[flight_df['dep_delay'] < 0].head()
```

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
0	1	9	813.0	-9.0	1014.0	-5.0	EV	EWR	DAY
1	9	4	2031.0	-9.0	2225.0	-34.0	EV	EWR	OMA
4	3	15	754.0	-5.0	916.0	-20.0	UA	EWR	ORD
6	10	23	919.0	-10.0	1234.0	0.0	AA	EWR	DFW
8	9	12	1011.0	-4.0	1233.0	21.0	US	JFK	CLT

✓ 2초 오후 3:06에 완료됨

3.데이터 처리(판다스 배열)

✓ [50] flight_df.query('dep_delay < 0').head()

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
0	1	9	813.0	-9.0	1014.0	-5.0	EV	EWR	DAY
1	9	4	2031.0	-9.0	2225.0	-34.0	EV	EWR	OMA
4	3	15	754.0	-5.0	916.0	-20.0	UA	EWR	ORD
6	10	23	919.0	-10.0	1234.0	0.0	AA	EWR	DFW
8	9	12	1011.0	-4.0	1233.0	21.0	US	JFK	CLT

✓ [51] flight_df[(flight_df['dep_delay'] < 0) & (flight_df['origin'] == 'EWR')].head()

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
0	1	9	813.0	-9.0	1014.0	-5.0	EV	EWR	DAY
1	9	4	2031.0	-9.0	2225.0	-34.0	EV	EWR	OMA
4	3	15	754.0	-5.0	916.0	-20.0	UA	EWR	ORD
6	10	23	919.0	-10.0	1234.0	0.0	AA	EWR	DFW
12	2	16	1857.0	-1.0	2201.0	5.0	UA	EWR	PBI

✓ [52] filter = (flight_df['dep_delay'] < 0) & (flight_df['origin'] == 'EWR')
flight_df.loc[filter, ['dep_delay', 'origin']]

	dep_delay	origin
0	-9.0	EWR
1	-9.0	EWR
4	-5.0	EWR
6	-10.0	EWR
12	-1.0	EWR

✓ 2초 오후 3:06에 완료됨

3.데이터 처리(판다스 배열)

3.6 sorting

```
[53] # sorting
flight_df.sort_values(by="dep_delay", ascending = False)
```

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
9980	8	6	1846.0	411.0	2110.0	410.0	WN	LGA	DEN
5686	5	22	146.0	406.0	255.0	378.0	UA	LGA	ORD
5281	9	12	2125.0	386.0	2258.0	396.0	B6	JFK	BTV
2652	3	8	1746.0	376.0	1938.0	359.0	DL	LGA	MSP
2679	12	5	1548.0	373.0	1712.0	357.0	WN	LGA	MDW
...
9748	3	6	NaN	NaN	NaN	NaN	AA	LGA	ORD
9768	4	10	NaN	NaN	NaN	NaN	US	LGA	BOS
9805	1	30	NaN	NaN	NaN	NaN	EV	EWR	PIT
9971	3	8	NaN	NaN	NaN	NaN	EV	EWR	BTV
9998	3	6	NaN	NaN	NaN	NaN	EV	LGA	CHO

10000 rows x 9 columns

```
[54] # sorting
flight_df[(flight_df.month == 12)].sort_values(by="day", ascending = True)
```

	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	origin	dest
6400	12	1	1930.0	30.0	2124.0	0.0	9E	LGA	DSM
3445	12	1	1712.0	8.0	1915.0	7.0	US	LGA	CLT
6924	12	1	1942.0	30.0	2112.0	36.0	EV	EWR	PWM
4738	12	1	1655.0	-5.0	1807.0	-12.0	US	LGA	DCA

✓ 2초 오후 3:06에 완료됨

```
[55] flight_df['gain'] = flight_df['dep_delay'] - flight_df['arr_delay']
flight_df
```

10000 rows x 10 columns

- 3.8 groupby

```
origin
EWR    15.034483
JFK    11.202257
LGA    10.524015
```

✓ 2초 오후 3:06에 완료됨

3.8 groupby

✓ 0초

[56] flight_df.groupby('origin')['dep_delay'].mean()

origin
EWR 15.034483
JFK 11.202257
LGA 10.524015
Name: dep_delay, dtype: float64

4.그래프 그리기

4.1 코랩에 한글폰트 설치

• <https://teddylee777.github.io/colab/colab-korean/>

✓ 10초

[57] !sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf

*** 세션 다시 시작 * * * *

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
 fonts-nanum
0 upgraded, 1 newly installed, 0 to remove and 35 not upgraded.
Need to get 10.3 MB of archives.
After this operation, 34.1 MB of additional disk space will be used.
Get:1 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 fonts-nanum all 20200506-1 [10.3 MB]
Fetched 10.3 MB in 1s (10.2 MB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78, <> line 1.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
dpkg-dev: (this frontend requires a controlling tty.)

연습문제

연습문제2

- ❖ 1. 00_02.weather를 가져오기
- ❖ 2. origin(출발공항)이 EWR 또는 JFK만 가져오기
- ❖ 3. origin에서 temp까지 변수를 모두 가져오기
- ❖ 4. 화씨 온도를 섭씨온도로 변환하기 $(F-32)/1.8$

$$C = \frac{(\text{온도} - 32)}{1.8}$$

- ❖ 5. 월별 평균 온도(섭씨) 구하기
- ❖ 6. 평균이 높은 값으로 sorting

```
month
7      27.177465
8      24.454902
6      20.916000
9      20.144186
5      16.675000
Name: temp_c, dtype: float64
```

III. 그래프 그리기

그래프의 중요성

❖ 그래프

- 인간이 지닌 시각적인 인지능력을 활용하여 직관적으로 그 현상을 쉽게 인식하도록 하는 방법
- 통계적인 데이터를 요약하여 시각적으로 그 특징을 나타내는 것

❖ 그래프의 문제점

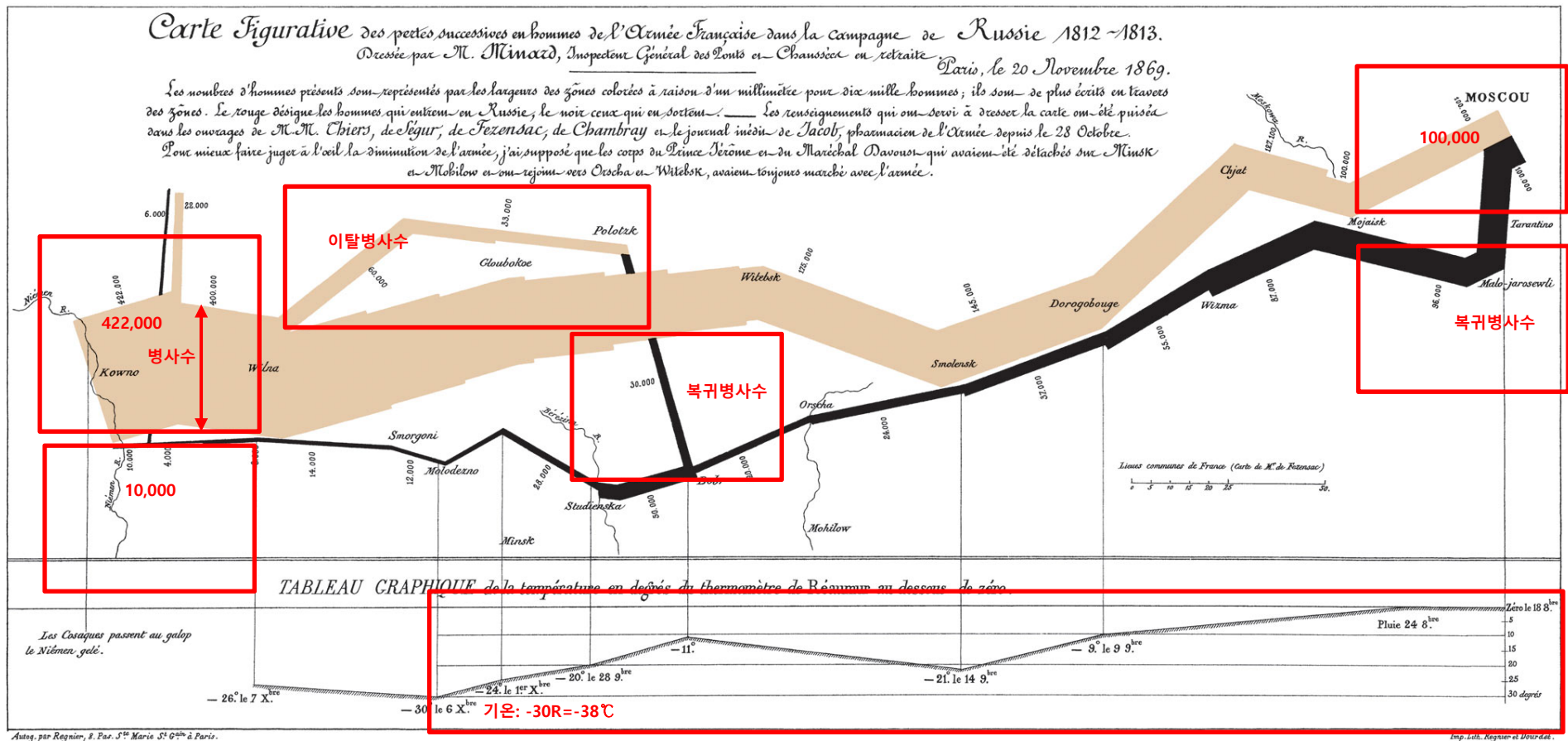
- 그래프는 자료가 가지고 있는 속성뿐만 아니라 강렬한 인상을 주게 되어 확대해석의 오류를 범할 위험이 있음
-

❖ 활용

- 그래프를 통해 데이터의 특징 및 자료의 이상치를 점검하여 이후의 통계분석에 대비함

그래프의 중요성

- ❖ Minard's graphic diagram(1868)
 - 나폴레옹 러시아 원정(1812.06~1813.1)

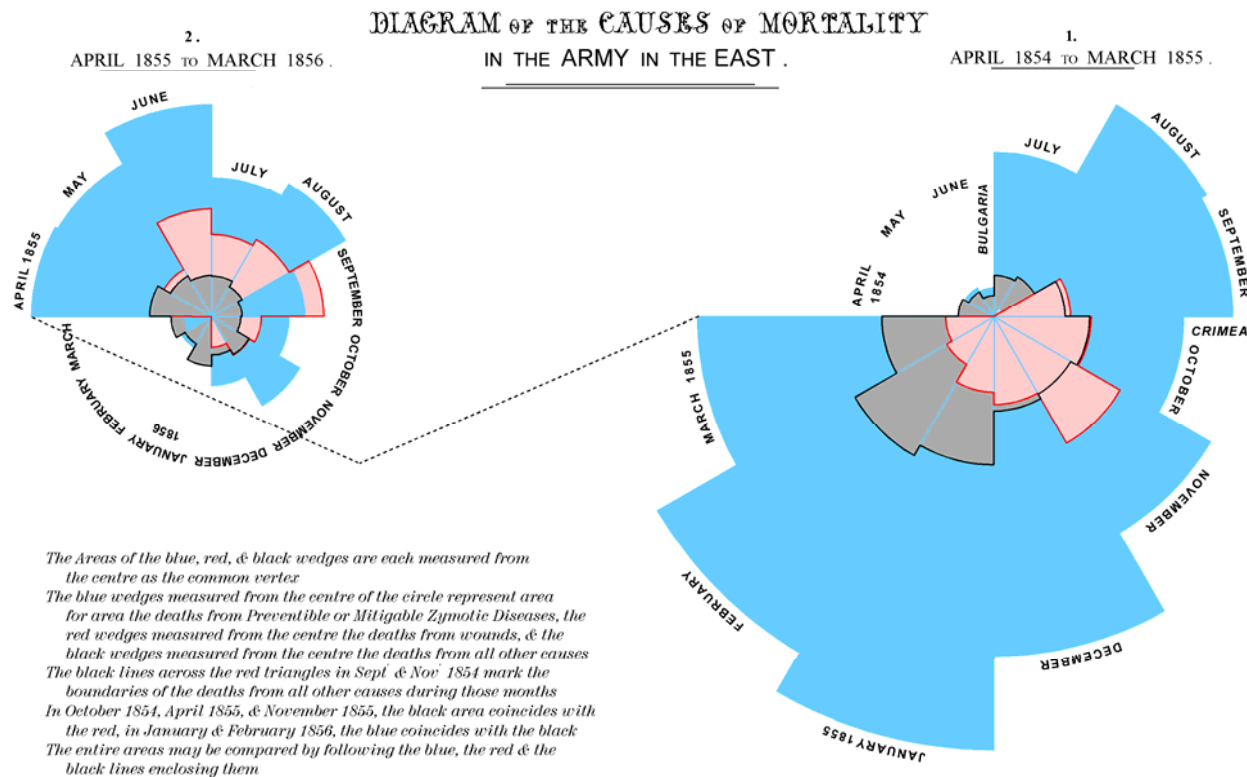


출처: <https://en.wikipedia.org/wiki/File:Minard.png>

- ❖ 나이팅게일의 로즈 다이어그램(원도표)
 - 1854년 ~ 1856년 크림리아 전쟁
 - 군인 1000당 사망률과 사망원인 파악

<http://www.Florence-Nightingale-Avenging-Angel.co.uk/Coxcomb.htm>

Diagram by Florence Nightingale, corrected by Hugh Small



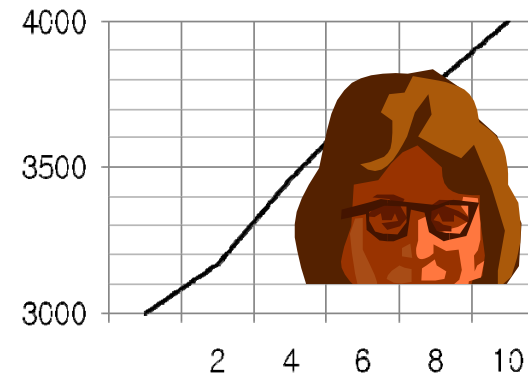
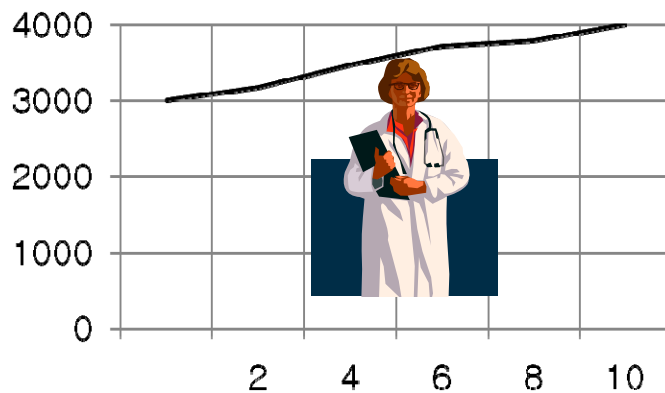
Red: 부상
Blue: 질병
Black: 기타

출처: <http://www.florence-nightingale-avenging-angel.co.uk/correct.tif>

올바른 그래프 그리기

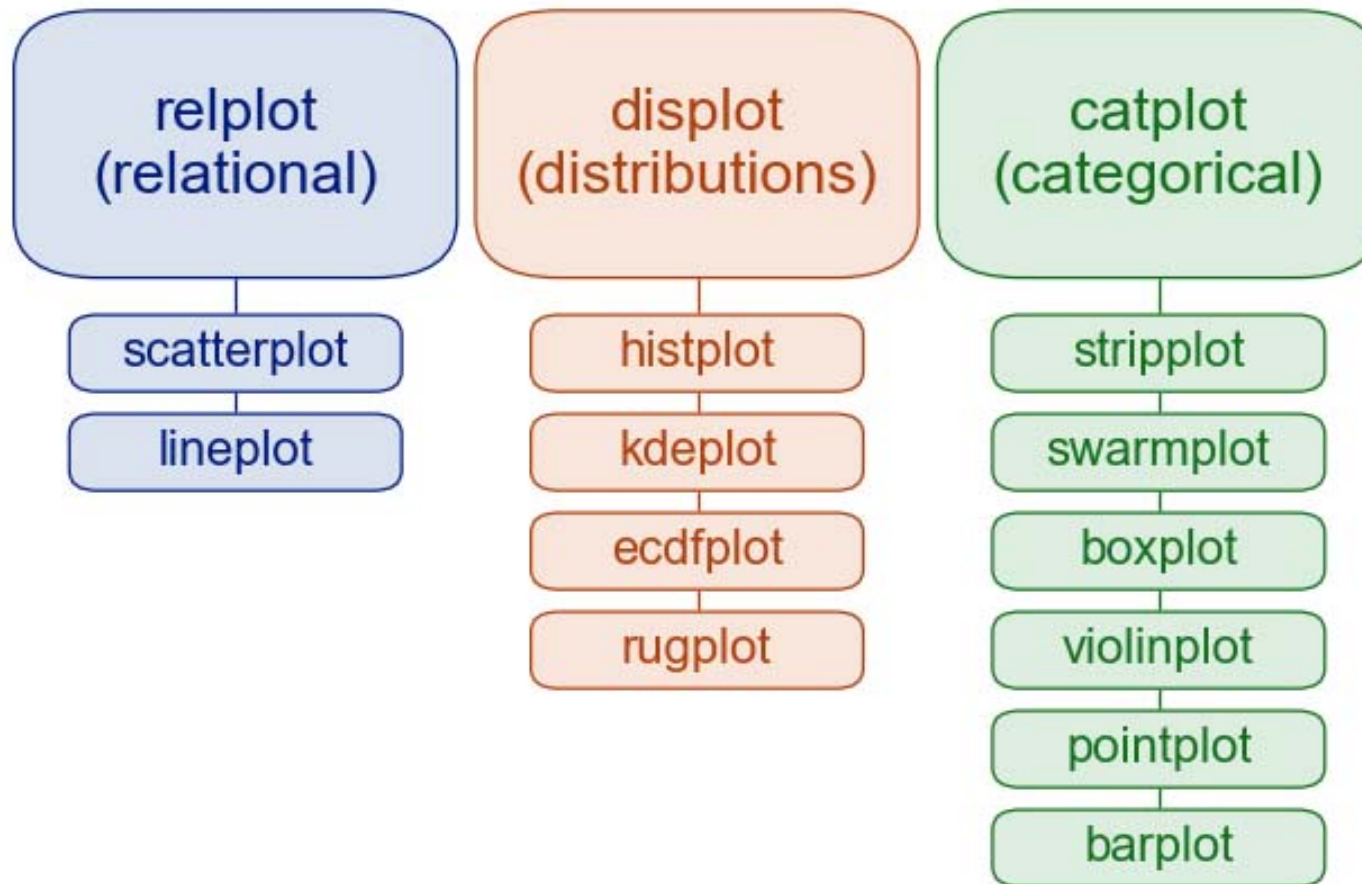
❖ 올바른 그래프의 작성

- 그래프에 적합한 제목
- 자료의 출처, 표본의 크기, 수집방법을 나타냄
- 축에 대한 제목을 명확히 함
- 도수비율, 퍼센트 등이 0에서 시작하는지 점검
- 변수의 측정단위가 표시되어야 함
- Y축 단위가 0이 아닐 때: 비교되는 자료를 동시에 포함



Seaborn 그래프 종류

❖ Seaborn 그래프 종류



출처: https://seaborn.pydata.org/tutorial/function_overview.html

4.그래프 그리기

4.그래프 그리기

4.1 코랩에 한글폰트 설치

<https://teddylee777.github.io/colab/colab-korean/>

```

[57] | sudo apt-get install -y fonts-nanum
      | sudo fc-cache -fv
      | rm ~/.cache/matplotlib -rf


# *** 세션 다시 시작 ***

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  fonts-nanum
0 upgraded, 1 newly installed, 0 to remove and 35 not upgraded.
Need to get 10.3 MB of archives.
After this operation, 34.1 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-nanum all 20200506-1 [10.3 MB]
Fetched 10.3 MB in 1s (10.2 MB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78, <> line 1.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package fonts-nanum.
(Reading database ... 121749 files and directories currently installed.)
Preparing to unpack .../fonts-nanum_20200506-1_all.deb ...
Unpacking fonts-nanum (20200506-1) ...
Setting up fonts-nanum (20200506-1) ...
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) ...
/usr/share/fonts: caching, new cache contents: 0 fonts, 1 dirs
/usr/share/fonts/truetype: caching, new cache contents: 0 fonts, 3 dirs
/usr/share/fonts/truetype/humor-sans: caching, new cache contents: 1 fonts, 0 dirs
/usr/share/fonts/truetype/liberation: caching, new cache contents: 16 fonts, 0 dirs
/usr/share/fonts/truetype/nanum: caching, new cache contents: 12 fonts, 0 dirs

```

2초

오후 3:06에 완료됨



49/862

4.그래프 그리기

```

/ root/.fontconfig: not creating non-existent cache directory
fc-cache: succeeded

✓ 0초 [58] import matplotlib.pyplot as plt
import seaborn as sns

# 테마 설정
sns.set_theme(style = "darkgrid")

# 한글 인식
plt.rc('font', family='NanumBarunGothic')
plt.rcParams['axes.unicode_minus'] = False # -인식

▼ 4.2 seaborn

• 그래프 종류
• relplot(relational): scatterplot, lineplot
• catplot(categorical): barplot, pointplot, boxplot, stripplot, swarmplot, violinplot
• displot(distributions): histplot, kdeplot, ecdfplot, rugplot

✓ 0초 [59] tips = sns.load_dataset("tips")
tips

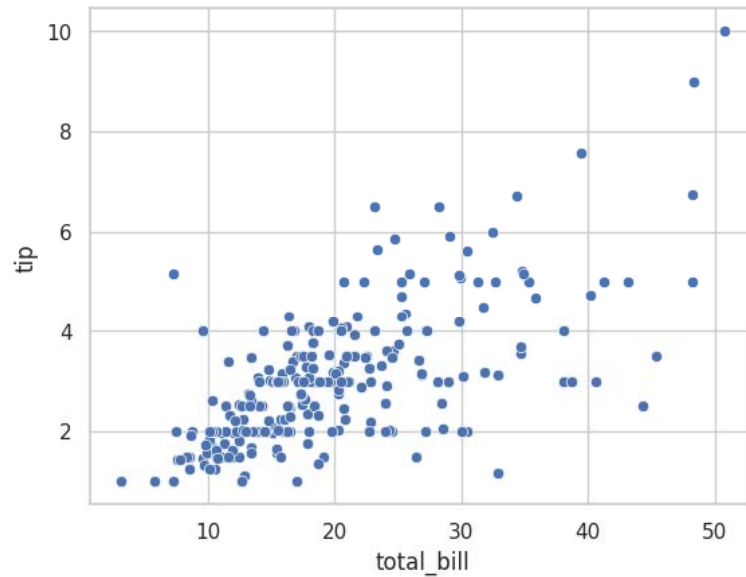
   total_bill  tip  sex  smoker  day  time  size
0      16.99  1.01 Female     No  Sun  Dinner     2
1      10.34  1.66  Male     No  Sun  Dinner     3
2      21.01  3.50  Male     No  Sun  Dinner     3
3      23.68  3.31  Male     No  Sun  Dinner     2
4      24.59  3.61 Female     No  Sun  Dinner     4
...         ...   ...     ...   ...   ...     ...
239     29.03  5.92  Male     No  Sat  Dinner     3
240     27.18  2.00 Female    Yes  Sat  Dinner     2
  
```

✓ 2초 오후 3:06에 완료됨

4. 그래프 그리기

```
[60] # 방법1
sns.set_theme(style="whitegrid")
sns.scatterplot(x = "total_bill",
               y = "tip",
               data = tips)

plt.show()
```

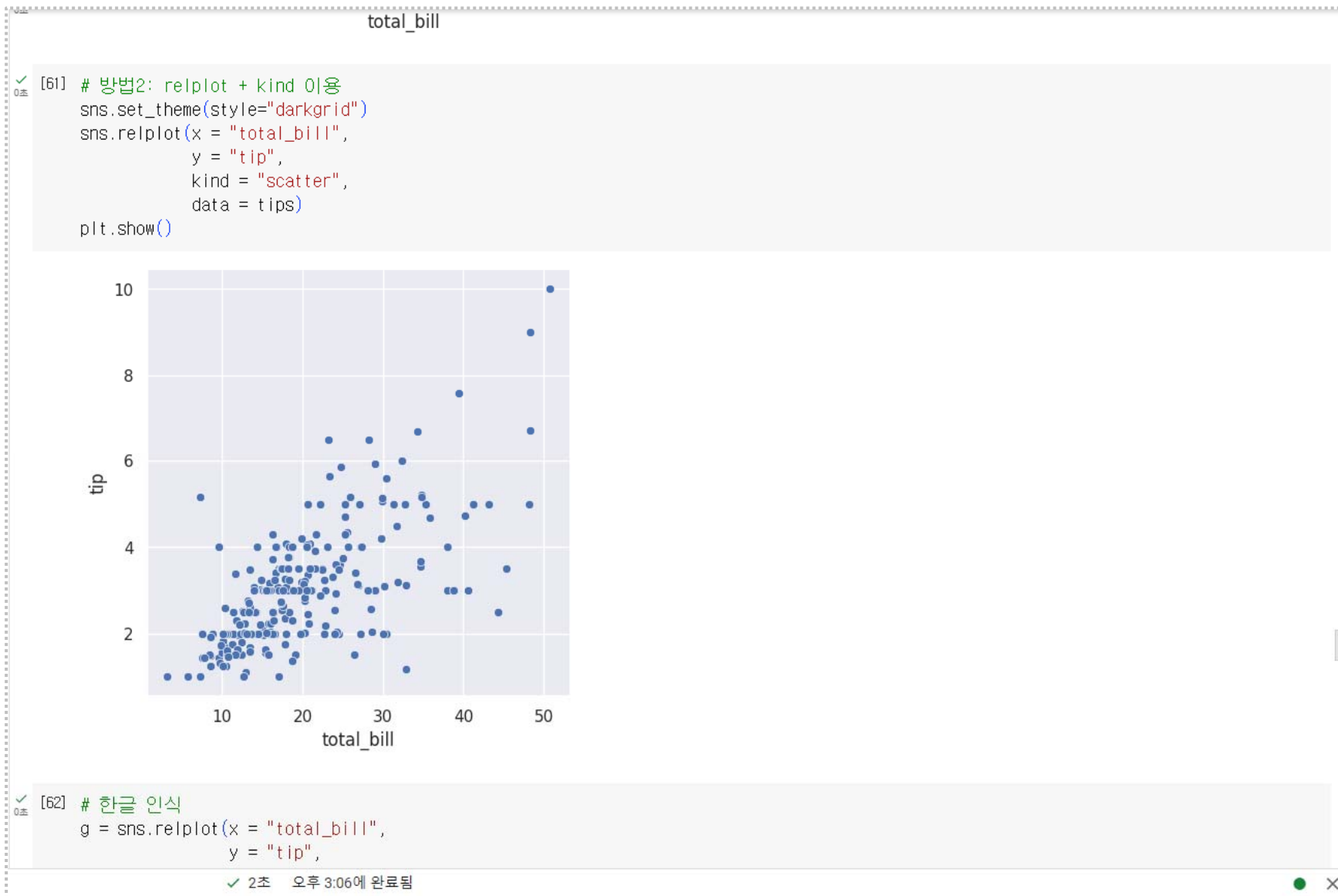


```
[61] # 방법2: relplot + kind 이용
sns.set_theme(style="darkgrid")
sns.relplot(x = "total_bill",
            y = "tip",
            kind = "scatter",
            data = tips)

plt.show()
```

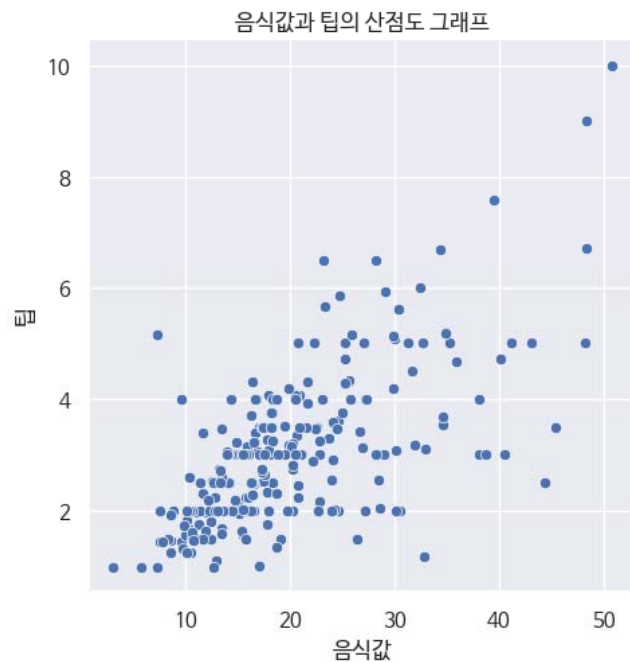
✓ 2초 오후 3:06에 완료됨

4.그래프 그리기



4.그래프 그리기

```
[3] # 한글 인식
g = sns.relplot(x = "total_bill",
                y = "tip",
                kind = "scatter",
                data = tips)
g.set(title = "음식값과 팁의 산점도 그래프",
      xlabel = "음식값",
      ylabel = "팁")
plt.show()
```



4.3 matplotlib

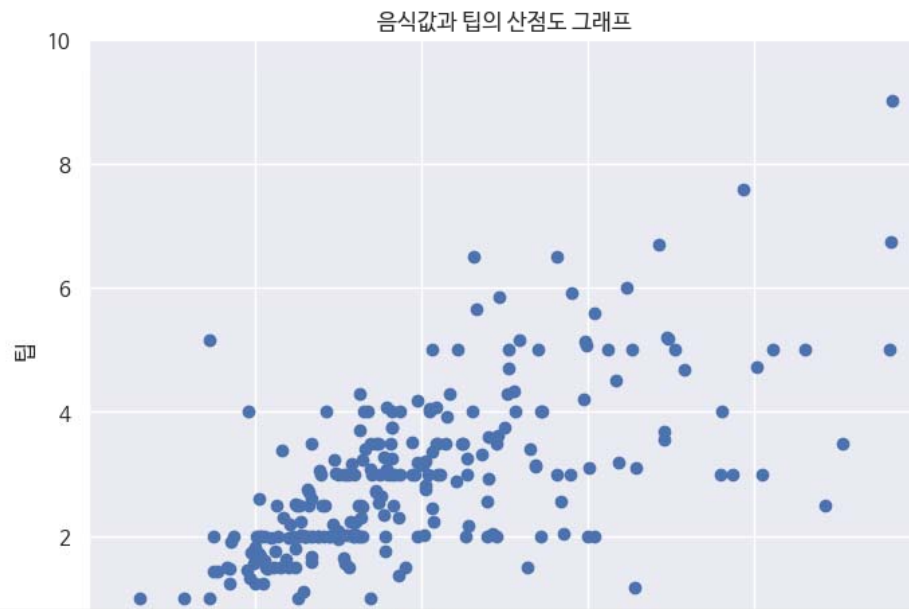
✓ 2초 오후 3:22에 완료됨

4.그래프 그리기

4.3 matplotlib

```
[4] x = np.array(tips.total_bill)
    y = np.array(tips.tip)

    fig, ax = plt.subplots(figsize = (8, 6))
    ax.scatter(x, y)
    ax.set(title = '음식값과 팁의 산점도 그래프',
            xlabel = '음식값',
            ylabel = '팁',
            xlim = (0, 50),
            ylim = (0, 10))
    plt.show()
```



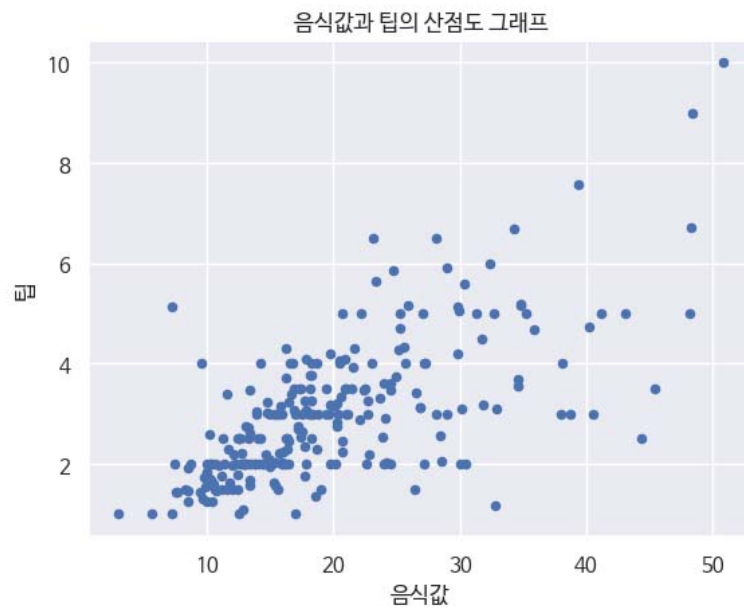
✓ 2초 오후 3:22에 완료됨

4.그래프 그리기

4.4.pdplot

- https://pandas.pydata.org/docs/user_guide/visualization.html#scatter-plot

```
[5] ax = tips.plot.scatter(x = "total_bill",
                          y = "tip")
ax.set(title = "음식값과 팁의 산점도 그래프",
       xlabel = "음식값",
       ylabel = "팁")
plt.show()
```



4.5 seaborn 활용 예제

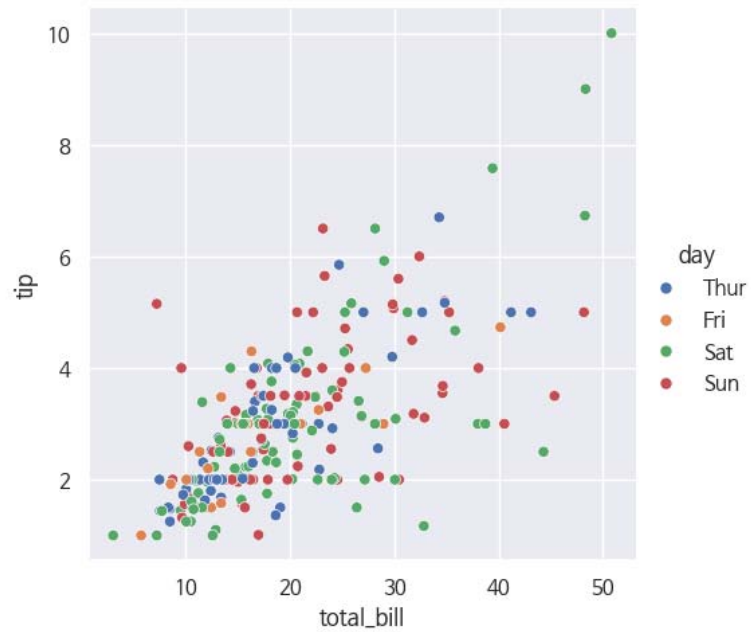
✓ 2초 오후 3:22에 완료됨

4.그래프 그리기

4.5 seaborn 활용 예제

```
[6] # hue: 색으로 분리
sns.relplot(x = "total_bill",
            y = "tip",
            hue = "day",
            data = tips)

plt.show()
```



```
[7] # 2개로 분리 (column)
sns.relplot(x = "total_bill",
            y = "tip",
            hue = "day",
            data = tips,
            col = "day")
```

✓ 2초 오후 3:22에 완료됨

4.그래프 그리기



4.그래프 그리기



연습문제

연습문제3

- ❖ 내장데이터(penguins)을 이용해서 그래프를 그리세요
- ❖ island : 서식지
- ❖ species : 펭귄 종
- ❖ bill_length_mm : 부리 길이
- ❖ bill_depth_mm : 부리 두께
- ❖ flipper_length_mm: 날개 길이
- ❖ body_mass_g: 몸무게
- ❖ sex: 성별

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

연습문제3

❖ 1. 데이터 불러오기, 한글 인식 및 기본 세팅하기

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

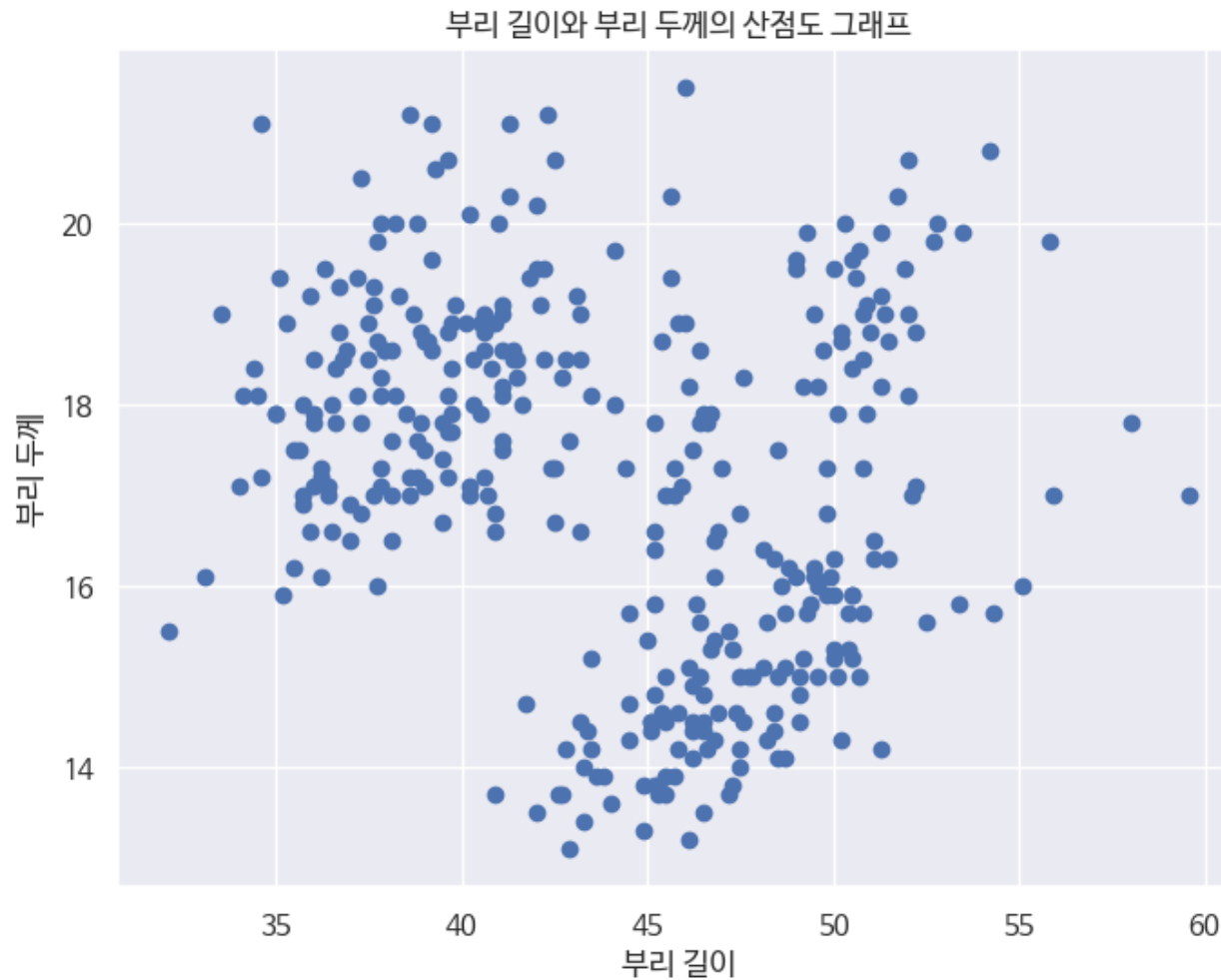
344 rows × 7 columns

- ❖ 2.seaborn을 이용해 bill_length_mm, bill_depth_mm의 산점도 그래프

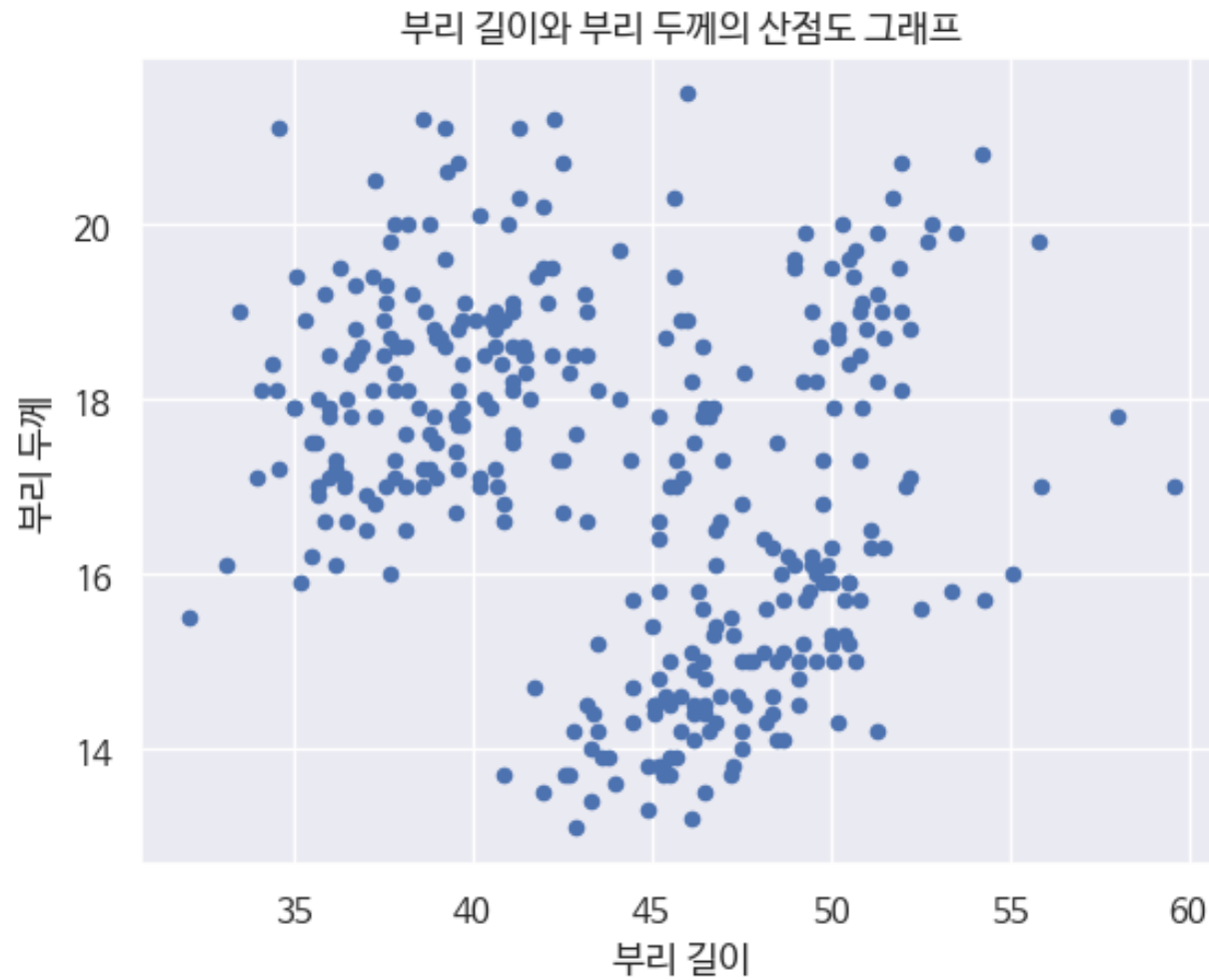


연습문제3

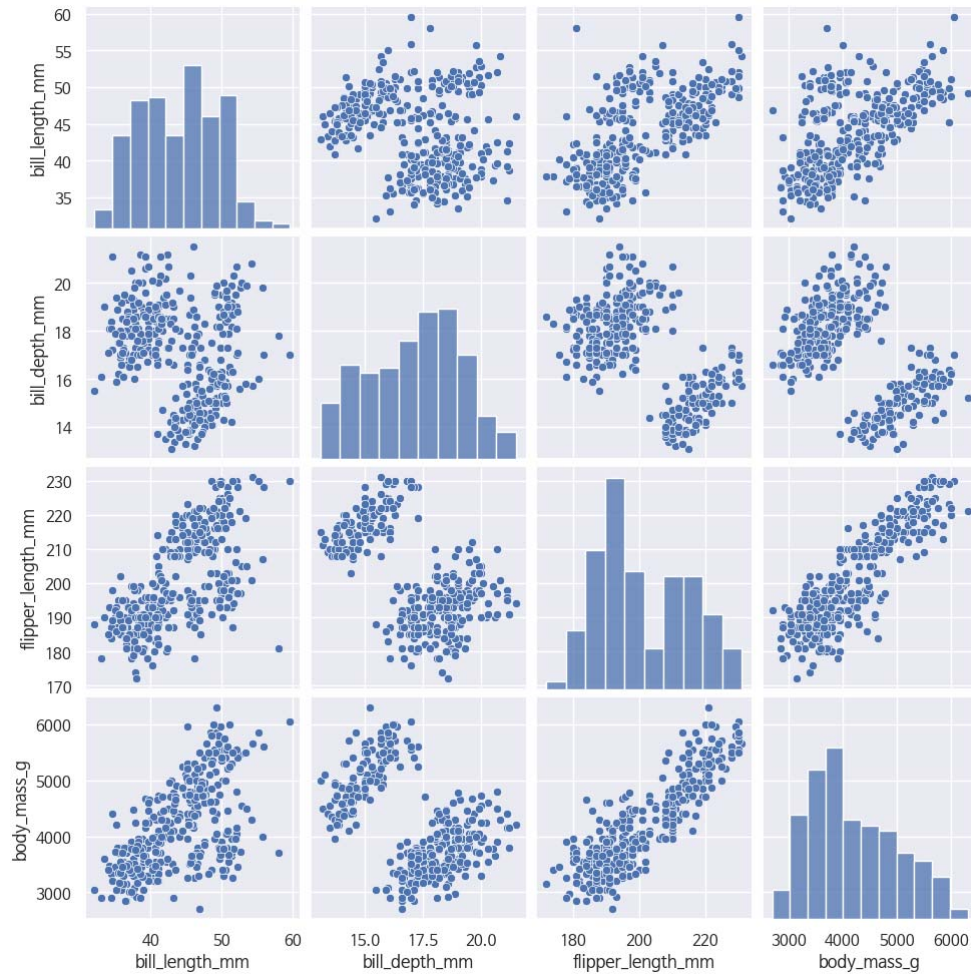
- ❖ 3.matplotlib을 이용해 bill_length_mm, bill_depth_mm의 산점도 그래프



- ❖ 4.pdplot을 이용해 bill_length_mm, bill_depth_mm의 산점도 그래프



❖ 5. 수치형 자료 관계 파악



연습문제3

- ❖ 6.bill_length_mm, bill_depth_mm의 산점도 그래프
 - Species, island로 구분

