

5장. 데이터 시각화



시각화 도구 - matplotlib

● matplotlib 라이브러리

- Python에서 가장 널리 사용되는 시각화 라이브러리이다.
- 데이터 분석, 과학 시각화, 보고서 작성 등 다양한 분야에서 그래프를 그릴 때 사용됨
- 주요 그래프 종류

| 종류 | 함수명 | 예시 |
|------------|--------|----------------|
| 선 그래프 | plot() | plt.plot(x, y) |
| 막대 그래프(수직) | bar() | plt.bar(x, y) |
| 막대 그래프(수평) | barh() | plt.barh(x, y) |
| 히스토그램 | hist() | plt.hist(data) |
| 파이차트 | pie() | plt.pie(data) |

시각화 도구 - matplotlib

- 선 그래프

```
import matplotlib.pyplot as plt
```

```
# 데이터 준비
```

```
x = [1, 2, 3, 4, 5]
```

```
y = [2, 3, 5, 7, 11]
```

```
# 선 그래프 그리기
```

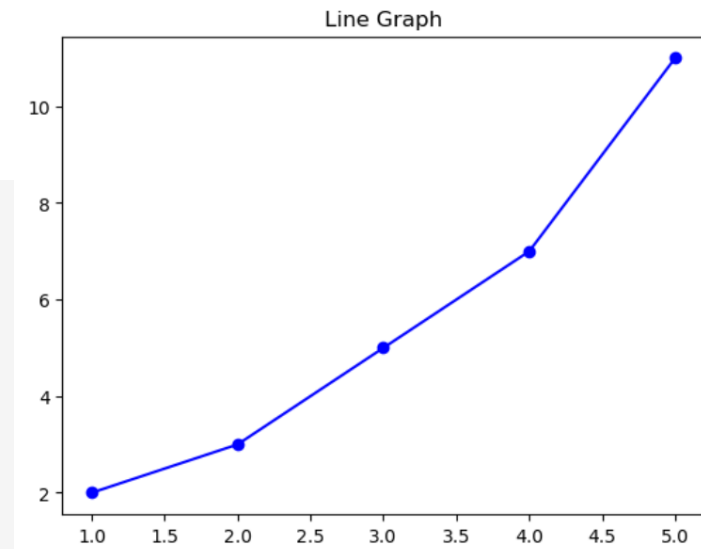
```
plt.plot(x, y, label='Prime numbers', color='blue', marker='o')
```

```
plt.title('Line Graph') #제목
```

```
plt.legend() #범례
```

```
# 그래프 출력
```

```
plt.show()
```



시각화 도구 - matplotlib

● 막대 그래프

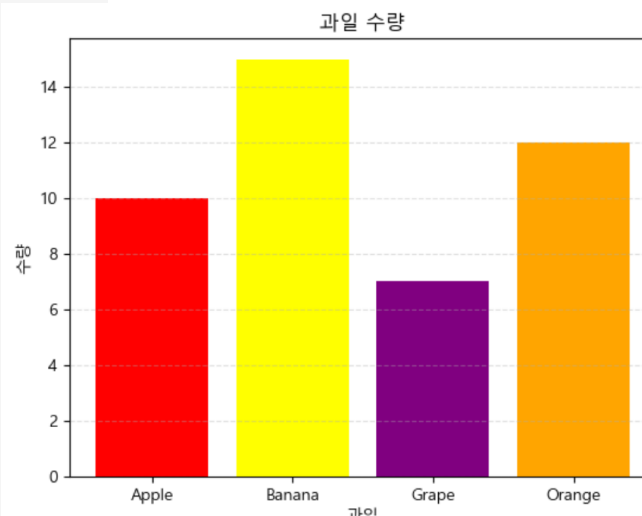
```
import matplotlib.pyplot as plt

# 데이터 준비
fruits = ['Apple', 'Banana', 'Grape', 'Orange']
counts = [10, 15, 7, 12]
color = ['red', 'yellow', 'purple', 'Orange']

# 한글 깨짐 방지 - Batang, Gulim 사용
plt.rc('font', family="Malgun Gothic")

# 막대 그래프
plt.bar(fruits, counts, color=color)
plt.title('과일 수량')
plt.xlabel('과일') # x축 제목
plt.ylabel('수량') # y축 제목

# y축 grid 추가
# plt.grid()
plt.grid(axis='y', linestyle='--', alpha=0.3)
plt.show()
```



시각화 도구 - matplotlib

- 원형 차트(파이 차트)

```
import matplotlib.pyplot as plt

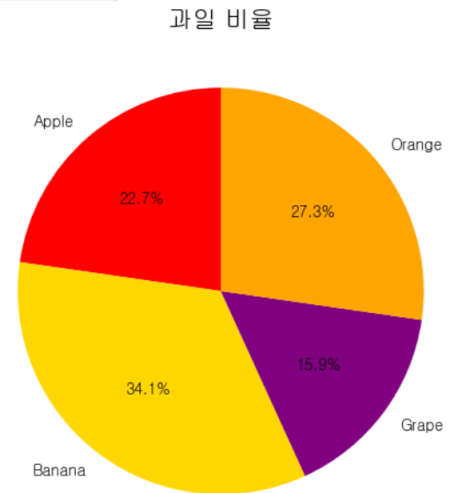
# 데이터
fruits = ['Apple', 'Banana', 'Grape', 'Orange']
counts = [10, 15, 7, 12]
colors = ['red', 'gold', 'purple', 'orange']

plt.rc('font', family="Gulim") # 한글 폰트

plt.figure(figsize=(6,6)) # 그래프 크기(가로-6인치, 세로-6

# 파이 차트
plt.pie(counts, labels=fruits, autopct='%1.1f%%',
        startangle=90, colors=colors)

plt.title('과일 비율', fontsize=16, fontweight='bold')
plt.show()
```

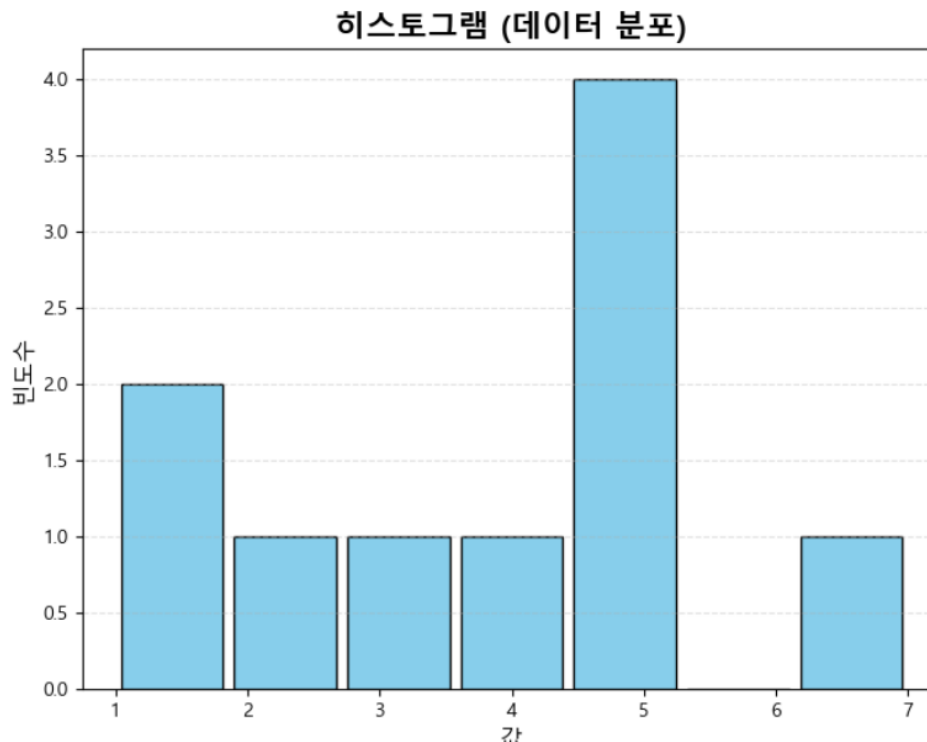


시각화 도구 - matplotlib

- 히스토그램

히스토그램은 데이터를 여러 구간으로 나누고, 각 구간에 속하는 데이터 개수를 세서 막대로 표시합니다.

bins=7 이라면 데이터 범위를 7개의 구간으로 나눠서 막대를 그립니다.



시각화 도구 - matplotlib

- 히스토그램

```
import matplotlib.pyplot as plt

# 데이터
data = [1, 1, 2, 3, 4, 5, 5, 5, 5, 7]

plt.rc('font', family='Batang') # 한글 폰트 설정
plt.figure(figsize=(8,6))      # 그래프 크기

# 히스토그램 그리기
plt.hist(data, bins=7, color='skyblue', edgecolor='black', rwidth=0.9)

# 제목과 축 이름
plt.title('히스토그램 (데이터 분포)', fontsize=16, fontweight='bold')
plt.xlabel('값', fontsize=12)
plt.ylabel('빈도수', fontsize=12)

# y축 그리드 추가
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```

시각화 도구 - matplotlib

● 히스토그램

```
import matplotlib.pyplot as plt
import random

#
numbers = []
for i in range(10):
    dice = random.randint(1, 6)
    numbers.append(dice)

print(numbers)

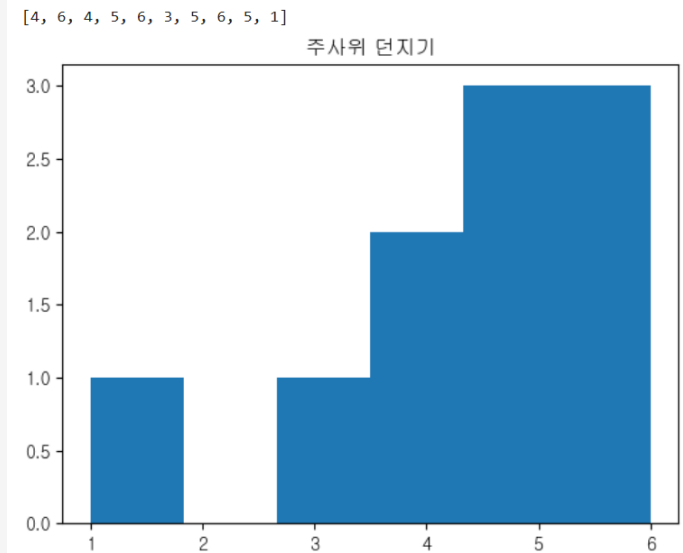
plt.rc('font', family='Malgun Gothic') # 한글 폰트 설정
plt.figure(figsize=(8,6))             # 그래프 크기

# 히스토그램 그리기
plt.hist(numbers, bins=6)

# 제목과 축 이름
plt.title('히스토그램 (주사위 던지기)', fontsize=16, fontweight='bold')
plt.xlabel('값', fontsize=12)
plt.ylabel('빈도수', fontsize=12)

# y축 그리드 추가
plt.grid(axis='y', linestyle='--', alpha=0.4)

plt.show()
```



seaborn 라이브러리

- **seaborn 라이브러리**

- Python의 데이터 시각화 라이브러리로, 통계적 그래프를 쉽게 그릴 수 있도록 matplotlib을 기반으로 만들어졌다
- pandas의 DataFrame과 잘 통합되어 있어, 데이터를 시각적으로 분석하고자 할 때 매우 유용하다

- **설치 방법**

pip install seaborn

타이타닉호 데이터 분석

- 타이타닉호 데이터 준비

```
import seaborn as sns

df = sns.load_dataset('titanic')
df.head()
```

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_male | deck | embark_town | alive |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|-------|------------|------|-------------|-------|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | True | NaN | Southampton | no |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | False | C | Cherbourg | yes |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | False | NaN | Southampton | yes |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | False | C | Southampton | yes |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | True | NaN | Southampton | no |

타이타닉호 데이터 분석

● 타이타닉호 데이터 정보(결측치 확인)

| # | Column | Non-Null Count | Dtype |
|-----|-------------|----------------|----------|
| --- | ----- | ----- | ----- |
| 0 | survived | 891 non-null | int64 |
| 1 | pclass | 891 non-null | int64 |
| 2 | sex | 891 non-null | object |
| 3 | age | 714 non-null | float64 |
| 4 | sibsp | 891 non-null | int64 |
| 5 | parch | 891 non-null | int64 |
| 6 | fare | 891 non-null | float64 |
| 7 | embarked | 889 non-null | object |
| 8 | class | 891 non-null | category |
| 9 | who | 891 non-null | object |
| 10 | adult_male | 891 non-null | bool |
| 11 | deck | 203 non-null | category |
| 12 | embark_town | 891 non-null | object |
| 13 | alive | 891 non-null | object |
| 14 | alone | 891 non-null | bool |



| # | Column | Non-Null Count | Dtype |
|-----|-------------|----------------|----------|
| --- | ----- | ----- | ----- |
| 0 | survived | 891 non-null | int64 |
| 1 | pclass | 891 non-null | int64 |
| 2 | sex | 891 non-null | object |
| 3 | age | 714 non-null | float64 |
| 4 | sibsp | 891 non-null | int64 |
| 5 | parch | 891 non-null | int64 |
| 6 | fare | 891 non-null | float64 |
| 7 | embarked | 889 non-null | object |
| 8 | class | 891 non-null | category |
| 9 | who | 891 non-null | object |
| 10 | adult_male | 891 non-null | bool |
| 11 | embark_town | 891 non-null | object |
| 12 | alive | 891 non-null | object |
| 13 | alone | 891 non-null | bool |

타이타닉호 데이터 분석 및 처리

- 타이타닉호 데이터 분석 및 처리

```
# df.info()
# df.isnull().sum()
print(df.isna().sum())
df['deck'].value_counts(dropna=False)
```

```
# 데이터 전처리
df2 = df.copy() #원본 복사
```

```
# deck 열 삭제
df2 = df2.drop('deck', axis=1)
# df2.info()
df2.columns # 컬럼 확인
```

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'embark_town', 'alive',
       'alone'],
      dtype='object')
```

타이타닉호 데이터 분석 및 처리

- 타이타닉호 데이터 분석 및 처리

```
# # age 열 결측 여부 확인
print(df['age'].isnull())
print(df['age'].head(10))

# 데이터 대체 - 평균값 이용
mean_age = df2['age'].mean()
print(mean_age) #29.699

# 평균값으로 채우기
df2['age'] = df2['age'].fillna(mean_age)
print(df2['age'].head(10))
```

| | | | |
|---|------|---|-----------|
| 0 | 22.0 | 0 | 22.000000 |
| 1 | 38.0 | 1 | 38.000000 |
| 2 | 26.0 | 2 | 26.000000 |
| 3 | 35.0 | 3 | 35.000000 |
| 4 | 35.0 | 4 | 35.000000 |
| 5 | NaN | 5 | 29.699118 |
| 6 | 54.0 | 6 | 54.000000 |

타이타닉호 데이터 분석 및 처리

- 타이타닉호 데이터 분석 및 처리

```
# embark_town 열 데이터 대체 - 최빈값 이용
print(df2['embark_town'][825:830])

most_freq = df['embark_town'].mode()[0]
print(most_freq) #Southampton

# 최빈값으로 채우기
df['embark_town'] = df['embark_town'].fillna(most_freq)
print(df['embark_town'][825:830])
```

| | |
|-----|-------------|
| 825 | Queenstown |
| 826 | Southampton |
| 827 | Cherbourg |
| 828 | Queenstown |
| 829 | NaN |

| | |
|-----|-------------|
| 825 | Queenstown |
| 826 | Southampton |
| 827 | Cherbourg |
| 828 | Queenstown |
| 829 | Southampton |

seaborn 라이브러리

- 타이타닉호 - 성별에 따른 생존자 수 그래프

```
import seaborn as sns
import matplotlib.pyplot as plt

# 예제 데이터셋 로드
df = sns.load_dataset('titanic')

# 데이터 확인
print(df.head())

# 성별에 따른 생존자 수 시각화 (막대그래프)
sns.countplot(data=df, x='sex', hue='survived')
plt.title('성별에 따른 생존자 수')
plt.show()
```

seaborn 라이브러리

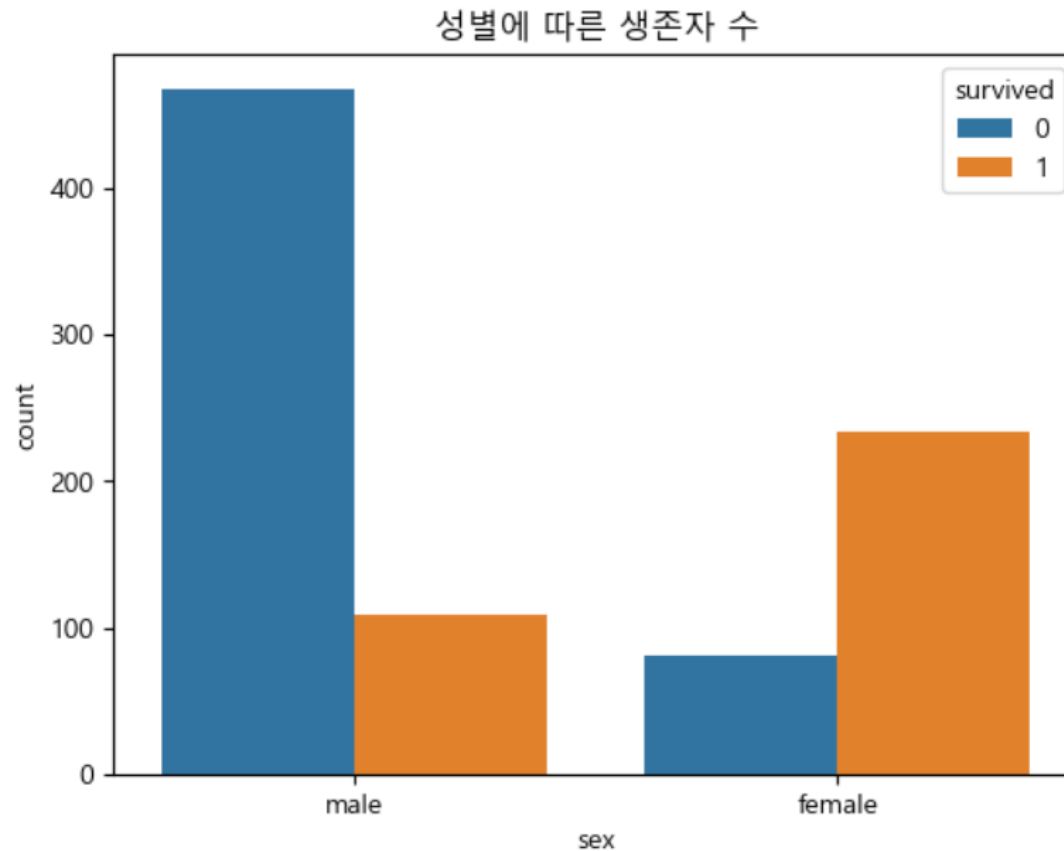
- 타이타닉호 - 성별에 따른 생존자 수 그래프

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | \ |
|---|----------|--------|--------|------|-------|-------|---------|----------|-------|---|
| 0 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | |
| 1 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | |
| 2 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | |
| 3 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | |
| 4 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | |

| | who | adult_male | deck | embark_town | alive | alone |
|---|-------|------------|------|-------------|-------|-------|
| 0 | man | True | NaN | Southampton | no | False |
| 1 | woman | False | C | Cherbourg | yes | False |
| 2 | woman | False | NaN | Southampton | yes | True |
| 3 | woman | False | C | Southampton | yes | False |
| 4 | man | True | NaN | Southampton | no | True |

seaborn 라이브러리

- 타이타닉호 - 성별에 따른 생존자 수 그래프



공공데이터 분석 및 시각화


- 자료 수집
 - 열린 데이터 광장(data.seoul.go.kr)



서울시 운동을 하지 않는 이유 통계

- 자료 수집
 - 운동을 하지 않는 이유 검색

로그인 | 회원가입 | 사이트맵

 서울 열린데이터 광장

공공데이터

통계

서울빅데이터

소식&참여

이용안내

통합검색

Home > 통합검색

☐ 결과 내 재검색

'운동을 하지 않는 이유'의 검색결과 1건을 찾았습니다.

전체(1)

공공데이터(0)

통계표(1)

메뉴(0)

게시물(0)

통계표(1)

서울시 운동을 하지 않는 이유 통계

서울시 통계정보시스템에서 제공하는 운동을 하지 않는 이유에 대한 통계정보 입니다.서울서베이의 서울시민 운동을 하지 않는 이유를 제공하는 일반 · 조사통계
공개일자: 2017-12-26 수정일자: 2020-05-21 제공기관: 서울특별시 제공부서: 디지털정책관 빅데이터담당관 담당자: 최성용(02-2133-4365)

서울시 운동을 하지 않는 이유 통계

● 자료 수집

- excel 파일 다운로드

1) 운동을 하지 않는 이유

자료갱신일 : 2022-06-30 / 수록기간 : 년 2009 ~ 2019
출처 : 서울특별시, 서울시도시정책정보조사

[항목[1/1]] [구분별[62/62]] [이유별[7/7]] [시점[1/7]]

(단위 : %)

| 시점 | 구분별(1) | 서울시 성별 | 연령별 | 학력별 | 소득별 |
|------|--------|-----------|-----|-----|-----|
| 2019 | | | | | |

다운로드

메타자료받기 (TXT)

파일형태

☐ 통계부호 ☐ 코드포함

☒ EXCEL(xlsx) ☐ EXCEL(xls) ☒ 셀 병합)
☐ CSV
☐ TXT

시점정렬

☒ 오름차순 ☐ 내림차순

소수점

☐ 수록자료형식과 동일 ☒ 조화면과 동일

다운로드

서울시 운동을 하지 않는 이유 통계

- 판다스로 엑셀 파일 읽어 오기
 - `pd.read_excel(파일명)`

```
import pandas as pd
import warnings
warnings.simplefilter('ignore')

# 제목행의 1번 인덱스만 출력(0번은 숨김)
not_exercise = pd.read_excel("./datas/exercise.xlsx", header=1)
not_exercise.head()
```

| | 시점 | 구분별 (1) | 구분별 (2) | 운동을 할 충분한 시간이 없어서 | 함께 운동을 할 사람이 없어서 | 운동을 할 만한 장소가 없어서 |
|---|--------|------------|------------|-------------------|------------------|------------------|
| 0 | 2019.0 | 서울시 | 소계 | 46.8 | 5.0 | 4.3 |
| 1 | NaN | 성별 | 남자 | 52.4 | 4.4 | 4.9 |
| 2 | NaN | NaN | 여자 | 42.5 | 5.6 | 3.9 |
| 3 | NaN | 연령별 | 10대 | 55.3 | 4.8 | 3.9 |
| 4 | NaN | NaN | 20대 | 46.0 | 4.2 | 4.5 |

서울시 운동을 하지 않는 이유 통계

- 데이터 전처리
 - '시점' 칼럼 삭제

```
# '시점' 칼럼 확인
print(not_exercise.columns[0])

# 칼럼 삭제 - 객체.drop()
not_exercise = not_exercise.drop('시점', axis=1)
not_exercise
```

시점

| | 구분별(1) | 구분별(2) | 운동을 할 충분한 시간이 없어서 | 함께 운동을 할 사람이 없어서 |
|---|--------|--------|-------------------|------------------|
| 0 | 서울시 | 소계 | 46.8 | 5.0 |
| 1 | 성별 | 남자 | 52.4 | 4.4 |

서울시 운동을 하지 않는 이유 통계

- 데이터 전처리
 - 칼럼명 변경

```
# 칼럼명 변경 - 객체.rename(), inplace=True(즉시 저장 및 실행)
not_exercise.rename(columns={'구분별(1)': '대분류', '구분별(2)': '분류'}, inplace=True)
not_exercise
```

| | 대분류 | 분류 | 운동을 할 충분한 시간이 없어서 | 함께 운동을 할 사람이 없어서 | 운동을 |
|---|-----|-----|-------------------|------------------|-----|
| 0 | 서울시 | 소계 | 46.8 | 5.0 | |
| 1 | 성별 | 남자 | 52.4 | 4.4 | |
| 2 | NaN | 여자 | 42.5 | 5.6 | |
| 3 | 연령별 | 10대 | 55.3 | 4.8 | |
| 4 | NaN | 20대 | 46.0 | 4.2 | |

서울시 운동을 하지 않는 이유 통계

- 데이터 전처리
 - 행 삭제

| | | | |
|----|-----|------|------|
| 45 | NaN | 영등포구 | 50.2 |
| 46 | NaN | 동작구 | 39.1 |
| 47 | NaN | 관악구 | 44.4 |
| 48 | NaN | 서초구 | 49.9 |
| 49 | NaN | 강남구 | 40.7 |
| 50 | NaN | 송파구 | 47.6 |
| 51 | NaN | 강동구 | 47.0 |

```
: # 행 삭제 - 객체.drop(index=range(시작행, 끝행-1))
not_exercise.drop(index=range(22, 52), inplace=True)
not_exercise
```


서울시 운동을 하지 않는 이유 통계

- 데이터 전처리
 - 행 수정

```
# 2행의 NaN을 '성별'로 변경 - '대분류' 칼럼의 2행을 수정
not_exercise.loc[2, '대분류'] = '성별'
not_exercise
```

| | 대분류 | 분류 | 운동을 할 충분한 시간이 없어서 | 함께 운동을 할 사람이 없어서 |
|---|-----|----|-------------------|------------------|
| 0 | 서울시 | 소계 | 46.8 | 5.0 |
| 1 | 성별 | 남자 | 52.4 | 4.4 |
| 2 | 성별 | 여자 | 42.5 | 5.6 |

서울시 운동을 하지 않는 이유 통계

- 데이터 전처리
 - 조건 검색

```
# 대분류가 성별인 항목만 검색
```

```
not_exercise['대분류'] == '성별'
```

```
not_exercise[not_exercise['대분류'] == '성별']
```

| | 대분류 | 분류 | 운동을 할 충분한 시간이 없어서 | 함께 운동을 할 사람이 없어서 |
|---|-----|----|-------------------|------------------|
| 1 | 성별 | 남자 | 52.4 | 4.4 |
| 2 | 성별 | 여자 | 42.5 | 5.6 |

서울시 운동을 하지 않는 이유 통계

- 데이터 전처리
 - 깊은 복사 & 얕은 복사

원본 유지 - 깊은 복사

```
not_ex_gender = not_exercise[not_exercise['대분류'] == '성별'].copy() #깊은 복사
not_ex_gender
```

얕은 복사, 깊은 복사

얕은 복사

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

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

```
b = a
b
```

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

```
b[1] = 10
b
```

```
[1, 10, 3, 4]
```

a # b리스트를 변경하면 a(원본)도 똑같이 변경됨

```
[1, 10, 3, 4]
```

깊은 복사

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

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

```
b = a.copy()
b
```

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

```
b[1] = 10
b
```

```
[1, 10, 3, 4]
```

a # b리스트를 변경해도 a는 원본을 유지함

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

서울시 운동을 하지 않는 이유 통계

- 데이터 전처리
 - 칼럼 삭제

```
# 대분류 칼럼 삭제
# not_ex_gender.drop("대분류", axis=1, inplace=True)
not_ex_gender.drop(columns='대분류', inplace=True)
not_ex_gender
```

| | 분류 | 운동을 할 충분한 시간이 없어서 | 함께 운동을 할 사람이 없어서 |
|---|----|-------------------|------------------|
| 1 | 남자 | 52.4 | 4.4 |
| 2 | 여자 | 42.5 | 5.6 |

서울시 운동을 하지 않는 이유 통계

- 데이터 전처리
 - 인덱스 설정

```
# index를 '분류'로 세팅  
not_ex_gender.set_index('분류', inplace=True)  
not_ex_gender
```

| 운동을 할 충분한 시간이 없어서 함께 운동을 할 사람이 없어서 | | |
|---------------------------------------|------|-----|
| 분류 | | |
| 남자 | 52.4 | 4.4 |
| 여자 | 42.5 | 5.6 |

서울시 운동을 하지 않는 이유 통계

- 시각화
 - 파이 차트 그리기

fig(figure) : 그래프를 그릴 공간

ax(axis) : 그 공간중 사용할 부분

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 2) # subplots() - 작은 그래프(1행 2열)
plt.rc('font', family='Malgun Gothic')

not_ex_gender['운동을 할 충분한 시간이 없어서'].plot.pie(ax=ax[0],
                                                            autopct='%.1f%%', startangle=90)
ax[0].set_title('운동을 할 충분한 시간이 없어서')
ax[0].set_ylabel('')

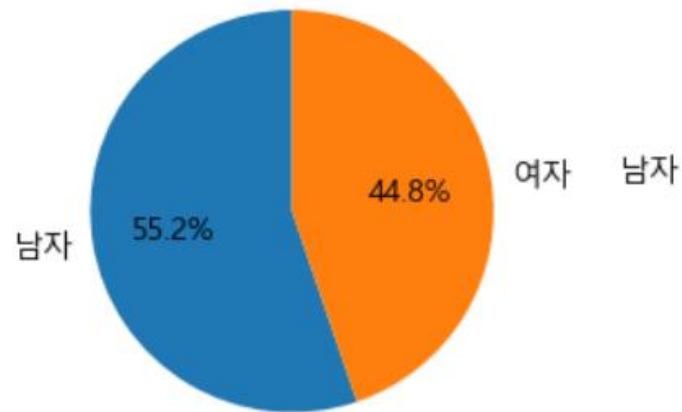
not_ex_gender['운동을 싫어해서'].plot.pie(ax=ax[1],
                                            autopct='%.1f%%', startangle=90)
ax[1].set_title('운동을 싫어해서')
ax[1].set_ylabel('')
```

서울시 운동을 하지 않는 이유 통계

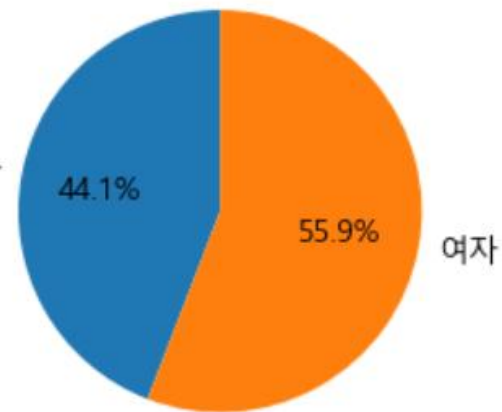
- 시각화
 - 파이 차트 그리기

Text(0, 0.5, '')

운동을 할 충분한 시간이 없어서



운동을 싫어해서



OpenCV

- OpenCV 사용법

- 영상 처리와 컴퓨터 비전을 위한 오픈 소스 라이브러리이다.
- 공장에서 제품 검사, 의료 영상 처리 및 보정, CCTV, 로봇틱스 등에 활용
- 설치

```
pip install opencv-python
```

- 사용

```
import cv2
```


OpenCV

- OpenCV 사용법

- 이미지 읽기.

cv2.imread(file, flag) //IMREAD_COLOR(컬러), IMREAD_GRAYSCALE(회색)

- 이미지 출력

cv2.imshow(이미지제목, 이미지파일)

- 키보드 입력을 처리하는 함수

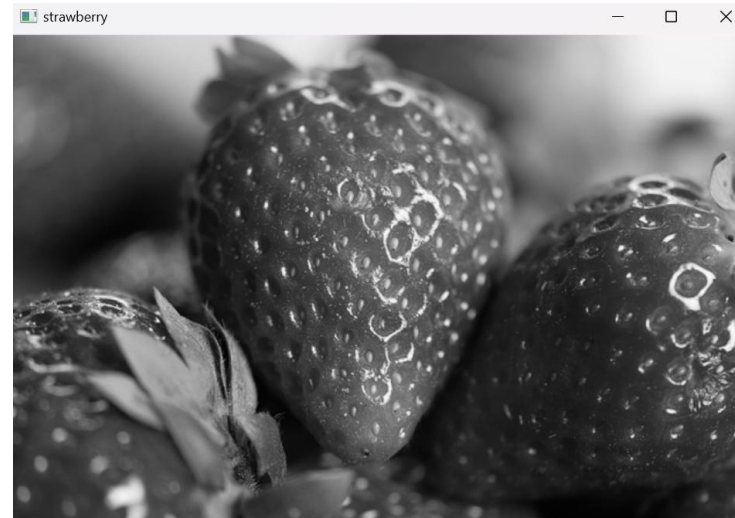
cv2.waitKey(0)

(0 - 계속 대기, 1000 - 1초)

- 윈도우(창) 닫기

cv2.destroyAllWindows()

이미지 출력



이미지 출력

- 이미지를 읽어서 창 띄우기

```
import cv2

img = cv2.imread('source/strawberry.jpg', cv2.IMREAD_COLOR)

# 이미지를 회색으로 변경
# img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imshow('strawberry', img) #새 창으로 띄우기

# cv2.waitKey(2000) #2초 대기
cv2.waitKey(0) # 계속 대기(아무키나 눌러 종료)

cv2.destroyAllWindows() #모든 창 닫기
```

이미지 출력

- 이미지를 읽어서 저장(쓰기)하기

```
import cv2

# 이미지를 흑백으로 읽기
img = cv2.imread('source/strawberry.jpg', cv2.IMREAD_GRAYSCALE)

cv2.imshow('strawberry', img) #창 열기

cv2.waitKey(0) #무한 대기

# output 폴더에 저장하기
cv2.imwrite("output/strawberry.jpg", img)

cv2.destroyAllWindows() #창 닫기
```