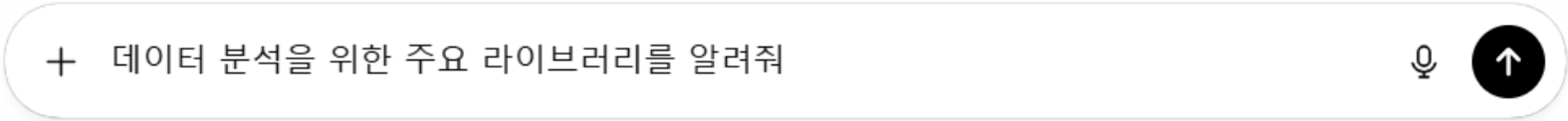
12장. 데이터 분석 및 시각화

판다스(Pandas)

- **데이터 분석용 주요 라이브러리**



라이브러리	주요 역할	핵심 기능
pandas	데이터 분석 핵심	DataFrame, CSV/Excel 처리, 필터링, 그룹화
numpy	수치계산 기반	다차원 배열, 수학/통계 연산, 고속 연산
matplotlib	데이터 시각화	선/막대/산점도 그래프

판다스(Pandas)

- **판다스**
  - 데이터를 수집, 정리하는 데 최적화된 도구로 **표 형태(행·열)의 데이터**를 빠르고 편리하게 처리하기 위한 라이브러리입니다.
  - 가장 인기있는 프로그래밍 언어인 **파이썬(python)**을 기반으로 한다.
  - 시리즈(Series)와 데이터프레임(DataFrame)이라는 구조화된 데이터 형식을 제공한다.
  - 판다스 설치 - **아나콘다 플랫폼**에는 이미 설치되어 있습니다.

    (pip install pandas)

판다스(Pandas)

- **판다스 자료구조**
  - **시리즈(Series)** – 데이터가 순차적으로 나열된 1차원 배열의 형태를 가지며, 인덱스(Index)는 데이터 값(value)과 일대일 대응이 된다.
  - **데이터프레임(DataFrame)** – 행(Row)과 열(Column)로 만들어지는 2차원 배열 구조로 표(Table) 형태이다.

	제품명
0	키보드
1	마우스
2	USB 메모리

인덱스 (index)

[시리즈]

	제품명	가격
0	키보드	20,000
1	마우스	32,000
2	USB 메모리	10,000

칼럼 (column)

[데이터프레임]

판다스(Pandas)

- Series(시리즈)

import pandas as pd(별칭)

pd.Series(데이터, 인덱스)

속성	기능
index	- 시리즈의 인덱스 (생략하면 기본 0, 1, 2)
values	- 시리즈의 값(요소)
shape	- 시리즈의 크기 (요소의수,)

판다스(Pandas)

- Series(시리즈)

시리즈(Series) 만들기

```
import pandas as pd

data = ['키보드', '마우스', 'USB 메모리']

s1 = pd.Series(data, index=[1, 2, 3])
print(s1)
print(type(s1))

# Series의 속성
print(s1.index)
print(s1.values)
print(s1.shape)

# 요소 선택 - 리스트 처럼 순서가 아닌 라벨 인덱스
print(s1[1])
print(s1[2])
```

```
1         키보드
2         마우스
3    USB 메모리
dtype: object
<class 'pandas.core.series.Series'>
Index([1, 2, 3], dtype='int64')
['키보드' '마우스' 'USB 메모리']
(3,)
키보드
마우스
```

판다스(Pandas)

- Series(시리즈)

```
# 두번째 시리즈
idx = [1, 2, 3]

s2 = pd.Series([20000, 32000, 10000], index = idx)
print(s2)
print(s2.shape)
print(s2.dtype)
```

```
1    20000
2    32000
3    10000
dtype: int64
(3,)
int64
```

판다스(Pandas)

- **DataFrame(데이터프레임)**

import pandas as pd(별칭)

pd.DataFrame(데이터, 인덱스)

속성	기능
index	- 데이터프레임의 인덱스 (생략하면 기본 0, 1, 2)
columns	- 데이터프레임의 칼럼
shape	- 데이터프레임의 크기 (행, 열)

판다스(Pandas)

- **DataFrame(데이터프레임)**

```python
import pandas as pd

df = pd.DataFrame({
    "name": s1,
    "price": s2
})

print(df)

# 데이터프레임의 속성
print(df.columns)
print(df.shape)  #(3, 2)
print(type(df))
```

```
      name  price
1      키보드  20000
2      마우스  32000
3  USB 메모리  10000
Index(['name', 'price'], dtype='object')
(3, 2)
<class 'pandas.core.frame.DataFrame'>
```

데이터프레임

- **DataFrame(데이터프레임)**

데이터프레임 만들기

```python
import pandas as pd

# 2차원 리스트
data = [
    ['키보드', 20000],
    ['마우스', 32000],
    ['USB 메모리', 11000]
]

# 데이터프레임 생성
df = pd.DataFrame(data, columns=['name', 'price'])
print(df)

# 칼럼 검색
print(df['name'])
print(df[['name', 'price']])
```

```
       name  price
0      키보드  20000
1      마우스  32000
2  USB 메모리  11000
0        키보드
1        마우스
2    USB 메모리
Name: name, dtype: object
       name  price
0      키보드  20000
1      마우스  32000
2  USB 메모리  11000
```

데이터프레임

- **DataFrame(데이터프레임)**

칼럼(Column) 다루기

칼럼 추가 및 칼럼명 변경

```
import pandas as pd

df = pd.DataFrame({
    "name": ['키보드', '마우스', 'USB 메모리'],
    "price": [20000, 32000, 11000]
})

# 칼럼 추가
df['spec'] = ['유선', '무선', '128GB']

df['할인가'] = df['price'] * 0.8
print(df)
print("-----------------------------------------")

# 수정후 반드시 df에 다시 저장.
df = df.rename(columns={"name": "품명", "price":"가격", "spec":"제품상세"})
print(df)
```

```
      name  price   spec      할인가
0     키보드  20000     유선  16000.0
1     마우스  32000     무선  25600.0
2  USB 메모리  11000  128GB   8800.0
-----------------------------------------
        품명     가격  제품상세      할인가
0     키보드  20000     유선  16000.0
1     마우스  32000     무선  25600.0
2  USB 메모리  11000  128GB   8800.0
```

CSV 파일 만들기

- **데이터 프레임을 CSV 파일로 변환**

  CSV(Comma Seperated Value) 파일: 쉼표(,)로 데이터를 구분하여 저장하는 텍스트 파일 형식이다. 주로 표형태의 데이터를 저장하는 데 사용하며 엑셀에서 쉽고 열고 편집할 수 있음

☞ CSV 파일로 변환

df.to_csv("computer.csv", index=False)

☞ CSV 파일 읽기

pd.read_csv("computer.csv")

CSV 파일 만들기

- **데이터 프레임을 CSV 파일로 변환**

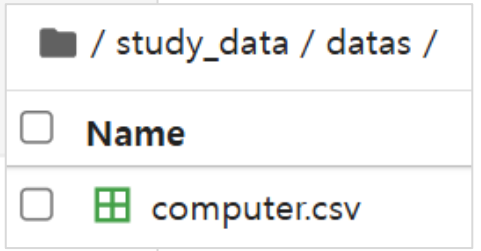
CSV 파일이란?

쉼표(,)로 데이터를 구분하여 저장하는 텍스트 파일 형식이다.

엑셀에서 .csv 형식으로 저장하고 불러올 수 있다.

```python
import pandas as pd

# csv 파일 만들기 - index 제외
df.to_csv("datas/computer.csv", index=False)

# csv 파일 읽기
print(pd.read_csv("datas/computer.csv"))
```

```
        품명     가격   제품상세      할인가
0       키보드  20000     유선   16000.0
1       마우스  32000     무선   25600.0
2  USB 메모리  11000  128GB    8800.0
```



행 / 열 선택

- **행과 열 선택**

속성	범위	예시
loc[인덱스명, 칼럼명] (location)	끝 인덱스 포함	인덱스 - 라벨인덱스
iloc[인덱스번호, 칼럼번호] (integer location)	끝 인덱스 - 1	인덱스 – 순서(0부터 시작)

행 / 열 선택

- **행/열 선택 : loc[행이름, 열이름] 사용**

```
# csv 파일 읽어서 df 객체에 저장
df = pd.read_csv("datas/computer.csv")

# 행 선택 - loc[행(행이름)]
print(df.loc[0:1])
print("========================================")

# 특정 요소 선택 - loc[행이름, 열(칼럼명)]
print(df.loc[0, "품명"])  #키보드
print(df.loc[1, "가격"])  #32000
print(df.loc[0:1, "품명"])
print(df.loc[0:1, ["품명", "가격"]])
```

```
    품명     가격 제품상세      할인가
0  키보드  20000   유선  16000.0
1  마우스  32000   무선  25600.0
========================================
키보드
32000
0    키보드
1    마우스
Name: 품명, dtype: object
    품명     가격
0  키보드  20000
1  마우스  32000
```

행 / 열 선택

- **행/열 선택 : iloc[행번호, 열번호] 사용**

```
# 행 선택 - iloc[행(인덱스번호)]
print(df.iloc[0:2]) # 끝인덱스-1
print("=========================================")

# 특정 요소 선택 - loc[행(인덱스번호), 열(인덱스번호)]
print(df.iloc[0, 0]) #키보드
print(df.iloc[1, 1]) #32000
print(df.iloc[0:2, 0])
print(df.iloc[0:2, 0:2])
```

```
    품명     가격 제품상세      할인가
0  키보드  20000   유선  16000.0
1  마우스  32000   무선  25600.0
=========================================
키보드
32000
0    키보드
1    마우스
Name: 품명, dtype: object
    품명     가격
0  키보드  20000
1  마우스  32000
```

데이터프레임 관리

- 데이터 프레임 - 요소값 변경(업데이트)

df.loc[행이름, 열이름] = “변경할 값 "

df.iloc[행번호, 열번호] = “변경할 값 "

- 데이터 프레임 - 요소값 추가

df[열] = “추가할 칼럼“ => 열 추가

df.loc[행] = “추가할 값“ => 행 추가

데이터프레임 관리

- **데이터 프레임 - 행/열 요소 변경 및 추가**

```
# 요소값 변경 및 추가
df2 = df.copy() #원본 유지 복사
df2

# 품명 '마우스'를 '무선마우스'로 변경
df2.loc[1, "품명"] = "무선마우스"
# 무선 마우스의 가격을 35000으로 변경
df2.loc[1, "가격"] = 35000
# 무선 마우스의 할인가 수정
df2.loc[1, "할인가"] = df2.loc[1, "가격"] * 0.8
df2

# 행 추가
df2.loc[3] = ["키보드", 40000, "무선", 0.0]
df2.loc[3, "할인가"] = df2.loc[3, "가격"] * 0.8
df2
```

	품명	가격	제품상세	할인가
0	키보드	20000	유선	16000.0
1	무선마우스	35000	무선	28000.0
2	USB 메모리	11000	128GB	8800.0
3	키보드	40000	무선	32000.0

데이터프레임 관리

- 데이터 프레임 - 행/열 삭제

df.drop(행, axis=0) => **행 삭제**

df.drop(열, axis=1) => **열 삭제**

```
# 행 삭제 - drop(행번호, axis=0)
# 열 삭제 - drop(칼럼명, axis=1)
df3 = df2.copy()
df3

# 3행 삭제
df3 = df3.drop(3, axis=0)
df3

# 열 삭제
# df3 = df3.drop("할인가", axis=1)
df3 = df3.drop(["제품상세", "할인가"], axis=1)
df3
```

	품명	가격
0	키보드	20000
1	무선마우스	35000
2	USB 메모리	11000

데이터프레임 관리

- 데이터 프레임 – 인덱스 설정(변경)

df.copy() – 데이터프레임 복사

df.set_index("칼럼명") – 인덱스 설정

```
가격          20000
제품상세          유선
할인가       16000.0
Name: 키보드, dtype: object
        가격 제품상세      할인가
품명
키보드  20000   유선  16000.0
마우스  32000   무선  25600.0
             가격    제품상세       할인가
품명
키보드        20000      유선   16000.0
마우스        32000      무선   25600.0
USB 메모리   11000  128GB    8800.0
```

데이터프레임 관리

- **데이터 프레임 – 인덱스 설정(변경)**

```
# 인덱스 설정 - set_index(칼럼명)
df4 = df.copy()
df4

df4 = df4.set_index("품명")
df4

# 행, 열 선택 - loc[ ]
print(df4.loc["키보드"])
print(df4.loc[["키보드", "마우스"]])
print(df4.loc["키보드":"USB 메모리"])
print("=========================================")

# 행, 열 선택 - iloc[ ]
print(df4.iloc[0])
print(df4.iloc[0:2])
print(df4.iloc[0:3])
```

KBO 팀순위 스크랩핑

- KBO 프로야구 순위

KBO 팀순위 스크랩핑

- KBO 프로야구 팀 순위

```python
import requests
from bs4 import BeautifulSoup

# KBO 팀순위 사이트
url = 'https://www.koreabaseball.com/Record/TeamRank/TeamRank.aspx'
response = requests.get(url)
html = BeautifulSoup(response.text, 'html.parser')

# 첫번째 테이블 검색
first_table = html.select_one('table > tbody')
# print(first_table)

# nth-of-type(특정 열 선택)
순위 = first_table.select('tr > td:nth-of-type(1)')
print(순위)
# for i in 순위:
#   print(i.text)
[int(i.text) for i in 순위] #순위를 리스트에 저장
```

```
[<td>1</td>, <td>2</td>, <td>3</td>, <td>4</td>, <td>5</td>, <td>6</td>, <td>7</td>, <td>8</td>, <td>9</td>, <td>10</td>]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

KBO 팀순위 스크랩핑

- KBO 프로야구 팀 순위

```
팀 = first_table.select('tr > td:nth-of-type(2)')
[i.text for i in 팀]
# 경기는 제외
승 = first_table.select('tr > td:nth-of-type(4)')
[int(i.text) for i in 승]

패 = first_table.select('tr > td:nth-of-type(5)')
[int(i.text) for i in 패]

무 = first_table.select('tr > td:nth-of-type(6)')
[int(i.text) for i in 무]

승률 = first_table.select('tr > td:nth-of-type(7)')
[float(i.text) for i in 승률] #승률은 실수
```

```
[0.603, 0.593, 0.536, 0.521, 0.514, 0.511, 0.478, 0.464, 0.442, 0.336]
```

KBO 팀순위 스크랩핑

- 텍스트 파일에 쓰고 읽기

```
# 텍스트 파일 쓰기

with open("datas/2025_kbo.txt", 'w') as f:
    f.write("========== 2025 한국 프로야구 성적표 ==========\n")
    head = "순위\t팀\t승\t패\t무\t승률"
    f.write(head)
    f.write('\n')
    for i in range(len(팀)): # 10번 반복
        txt = f'{순위[i].text}\t{팀[i].text}\t{승[i].text}\t{패[i].text}\t \
{무[i].text}\t{승률[i].text}\n'
        f.write(txt)
```

KBO 팀순위 스크랩핑

- 텍스트 파일에 쓰고 읽기

```python
# 텍스트 파일 읽기
with open("datas/2025_kbo.txt", 'r') as f:
    data = f.read()
    print(data)
```

```
========== 2025 한국 프로야구 성적표 ==========
순위    팀      승    패    무    승률
1       LG      85    56    3     0.603
2       한화    83    57    4     0.593
3       SSG     75    65    4     0.536
4       삼성    74    68    2     0.521
5       NC      71    67    6     0.514
6       KT      71    68    5     0.511
7       롯데    66    72    6     0.478
8       KIA     65    75    4     0.464
9       두산    61    77    6     0.442
10      키움    47    93    4     0.336
```

실습 문제

- 데이터 프레임 만들고 csv 파일로 변환하기

```
import pandas as pd

data = {
    "순위": [int(i.text) for i in 순위],
    "팀": [i.text for i in 팀],
    "승": [int(i.text) for i in 승],
    "패": [int(i.text) for i in 패],
    "무": [int(i.text) for i in 무],
    "승률": [float(i.text) for i in 승률]
}

kbo = pd.DataFrame(data)
kbo

kbo.to_csv("./datas/kbo_2025.csv", index=False)

df = pd.read_csv("./datas/kbo_2025.csv")
df
```

데이터 탐색

- **데이터 탐색을 위한 주요 함수**

함수	기능
head(n) / tail(n)	- 상위, 하위 n개의 행을 보여줌 (생략되면 기본 5개)
info()	- 데이터의 정보 (데이터 개수, 칼럼, 자료형 등)
value_counts()	- 칼럼의 고유한 값의 개수
describe(include='number') describe(include='object')	- 수치형 칼럼 통계 요약 - 문자형 칼럼 통계 요약
astype(자료형)	- 칼럼의 자료형 변환 (int – 정수형, float-실수형)

데이터 탐색

- 데이터 프레임을 CSV 파일로 변환

```
# 데이터프레임 만들기
food = pd.DataFrame({
    "메뉴" : ["김밥", "비빔밥", "자장면", "우동", "김밥", "자장면", "김밥"],
    "가격" : [3000, 7000, 5500, 5500, 3500, 6000, 4000],
    "분류" : ["한식", "한식", "중식", "일식", "한식", "중식", "한식"]
})
# print(food)

# csv 파일로 변환
food.to_csv("food.csv", index=False)

# csv 파일 읽어오기
df = pd.read_csv("food.csv")
print(df)
```

데이터 탐색

- 데이터 탐색을 위한 주요 함수

```
# 데이터 출력 - head(), tail() -> 기본 5개
print(df.head())
print(df.tail())

# 데이터 정보 - info()
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   메뉴      7 non-null      object
 1   가격      7 non-null      int64
 2   분류      7 non-null      object
dtypes: int64(1), object(2)
memory usage: 300.0+ bytes
```

데이터 탐색

- 데이터 탐색을 위한 주요 함수

```
# 고유한 값의 개수 - value_counts()
print(df["메뉴"].value_counts())
print(df["분류"].value_counts())

# 통계 요약 - describe()
# print(df.describe())
print(df.describe(include='number'))
print(df.describe(include='object'))
```

```
메뉴
김밥      3
자장면     2
비빔밥     1
우동      1
Name: count, dtype: int64
분류
한식     4
중식     2
일식     1
Name: count, dtype: int64
```

	가격
count	7.000000
mean	4928.571429
std	1455.694892
min	3000.000000
25%	3750.000000
50%	5500.000000
75%	5750.000000
max	7000.000000

데이터 탐색

- 데이터 탐색을 위한 주요 함수

```
# 칼럼 추가 - "할인가"(10%)
discount = 0.1
df["할인가"] = df["가격"] * (1 - discount)
print(df)
print(df.info())

# 자료형 변환 - astype()
df["할인가"] = df["할인가"].astype("int64")
print(df)
print(df.info())
```

	메뉴	가격	분류	할인가
0	김밥	3000	한식	2700
1	비빔밥	7000	한식	6300
2	자장면	5500	중식	4950
3	우동	5500	일식	4950
4	김밥	3500	한식	3150
5	자장면	6000	중식	5400
6	김밥	4000	한식	3600

데이터 탐색

- **조건 검색(필터링)**

비교 연산 기호	설명
>, >=	크다, 크거나 같다.
<, <=	작다, 작거나 같다
==	두 항이 같다.
!=	두 항이 같지 않다.

논리 연산	연산 기호	설명
교집합(AND)	&	두 조건이 모두 참일때 True 반환
합집합(OR)	|	두 조건중 하나 이상이 참일때 True 반환
부정(NOT)	~	조건이 참이면 False, 거짓이면 True를 반환

데이터 탐색

- **데이터 탐색 – 조건 검색(필터링)**

```python
# 조건 검색(필터링)
# 메뉴가 '자장면'인 음식의 정보
result1 = (df["메뉴"] == '자장면')
print(df[result1])

# 메뉴가 '김밥'이 아닌 음식의 정보
result2 = (df["메뉴"] != '김밥')
# result2 = ~(df["메뉴"] == '김밥')
print(df[result2])

# 가격이 4000원 미만인 음식의 정보
result3 = (df["가격"] < 4000)
print(df[result3])

# 메뉴가 '자장면'이고, 가격이 6000원 이상인 음식의 정보
# result4 = (df["메뉴"] == '자장면') & (df["가격"] >= 6000)
result4 = (df["메뉴"] == '자장면') | (df["가격"] >= 6000)
print(df[result4])
```

데이터 탐색

- **데이터 탐색 - 정렬**

정렬 분류	함수	파라미터
칼럼 기준	sort_values(칼럼, 파라미터)	오름차순: ascending=True 내림차순: ascending=False
인덱스 기준	sort_index(파라미터)	

	메뉴	가격	분류
1	비빔밥	7000	한식
5	자장면	6000	중식
2	자장면	5500	중식
3	우동	5500	일식
6	김밥	4000	한식
4	김밥	3500	한식
0	김밥	3000	한식

	메뉴	가격	분류
6	김밥	4000	한식
4	김밥	3500	한식
0	김밥	3000	한식
1	비빔밥	7000	한식
3	우동	5500	일식
5	자장면	6000	중식
2	자장면	5500	중식

데이터 탐색

- **데이터 탐색 - 정렬**

```
df = pd.read_csv("datas/food.csv")
# print(df)
# print("=========================================")

# 가격을 내림차순으로 정렬하기
df.sort_values("가격", ascending=False, inplace=True)
print(df)
print("=========================================")

# 메뉴를 오름차순으로 정렬, 메뉴가 같으면 가격을 내림차순 정렬
df.sort_values(["메뉴", "가격"], ascending=[True, False], inplace=True)
print(df)
print("=========================================")

# 인덱스 기준 정렬(초기화)
df.sort_index(ascending=True)
print(df)
```

판다스의 주요 함수

- **판다스의 통계, 수학 , 그룹핑 함수**

함수	기능
count() / sum() / mean()	개수 / 합계 / 평균
var() / std()	분산 / 표준편차
max() / min()	최대값 / 최소값
idxmax() / idxmin()	최대값 위치 / 최소값 위치
quantile(.25) / quantile(.75)	1사분위수 / 3사분위수
mode()	빈도값
round(수, 자리수)	반올림
groupby(칼럼명)	그룹화

판다스의 주요 함수

- 판다스의 통계, 수학 함수

```
print("개수:", df["가격"].count())
print("합계:", df["가격"].sum())
print("평균:", round(df["가격"].mean(), 2))
print("분산:", round(df["가격"].var(), 2))
print("표준편차:", round(df["가격"].std(), 2))
print("최대값:", df["가격"].max())
print("최소값:", df["가격"].min())
print("최대값의 위치:", df["가격"].idxmax())
print("최소값의 위치:", df["가격"].idxmin())
```

판다스의 주요 함수

- 판다스의 통계, 수학 함수

```
# 사분위수 - quantile()
print("1사분위수:", df["가격"].quantile(.25))
print("2사분위수:", df["가격"].quantile(.50))
print("3사분위수:", df["가격"].quantile(.75))
print("4사분위수:", df["가격"].quantile(1.0))

# 조건 - 1사분위수 보다 작은 가격을 출력
result = df["가격"] < df["가격"].quantile(.25)
print(df[result])

# 최빈값 - mode()
print("최빈값:", df["가격"].mode()[0])
```

판다스의 주요 함수

- 그룹핑 - groupby()

```
import pandas as pd

# "수량" 칼럼 추가
df["수량"] = [10, 10, 15, 5, 7, 7, 8]
df

# csv 파일로 변환
df.to_csv("./datas/food2.csv", index=False)

# csv 파일 읽기
food = pd.read_csv("./datas/food2.csv")
food
```

```
메뉴,가격,분류,수량
김밥,3000,한식,10
비빔밥,7000,한식,10
자장면,5500,중식,15
우동,5500,일식,5
김밥,3500,한식,7
자장면,6000,중식,7
김밥,4000,한식,8
```

판다스의 주요 함수

- 그룹핑 - groupby()

```
food = pd.read_csv("food2.csv")
# print(food)

# 그룹화 - groupby()
df["분류"].value_counts()

food.groupby("메뉴").mean(numeric_only=True)
food.groupby("분류").mean(numeric_only=True) #모든 수치 칼럼 평균

food.groupby("분류")["가격"].mean()  #분류별 가격의 평균
food.groupby("분류")["수량"].mean()  #분류별 수량의 평균
```

데이터 전처리

- **결측치 삭제 및 대체**

함수	설명
drop(행, axis=0) drop(열, axis=1)	NaN이 포함된 행 삭제 NaN이 포함된 열 삭제
dropna()	NaN이 포함된 모든 행 삭제
fillna()	- 수치형: 평균(mean()), 중간값(median()) - 문자형: 빈도값(mode())
isnull() isnull().sum()	- 결측치 확인(True / False로 반환) - 결측치 개수(True(1))

데이터 전처리

- **결측 데이터 프레임 생성**

```python
import pandas as pd
import numpy as np #수치 계산 라이브러리

# 데이터프레임 만들기
df = pd.DataFrame({
    'A': [1, None, 3],
    'B': [4, 5, None]
})
df

# 칼럼 추가
df['C'] = np.nan
df

# csv 파일로 변환
df.to_csv("./datas/data_nan.csv", index=False)
```

data_nan.csv

	A	B	C
1	1.0	4.0	
2		5.0	
3	3.0		

✓ **None**은 파이썬에서 NULL을 의미(공백)

✓ **np.nan**은 넘파이가 제공하는 데이터 누락을 의미함

데이터 전처리

- **결측치 삭제**

```
# csv 파일 읽기
df_nan = pd.read_csv("./datas/data_nan.csv")
df_nan

# 데이터 탐색
# df_nan.info()
# df_nan.isna().sum()

# 결측치가 있는 1행 삭제
# inplace=True (실행중 즉시 삭제)
df_nan.drop(1, axis=0, inplace=True)
df_nan

# 결측치가 있는 열 삭제
df_nan = df_nan.drop('C', axis=1)
df_nan
```

	A	B
0	1.0	4.0
2	3.0	NaN

데이터 전처리

- **결측치 대체**

```
data = {
    "메뉴": ["김밥", "비빔밥", "자장면", "우동", None, "자장면", "김밥"],
    "가격": [3000, 7000, 5500, None, 3500, 6000, 4000],
    "분류": ["한식", "한식", "중식", "일식", "한식", "중식", "한식"]
}

food = pd.DataFrame(data)
food
food.info()

# 결측치 확인 - isnull()
print(food.isnull())
print(food.isnull().sum()) #True(1) / False(0)
```

```
메뉴    1
가격    1
분류    0
```

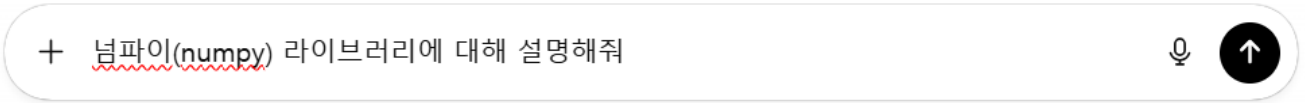
데이터 전처리

- **결측치 대체**

```
# 수치형 - 중간값으로 대체
median = food["가격"].median()
median

# fillna()
food["가격"] = food["가격"].fillna(median)
food

# 문자형 - 최빈값
frequency = food["메뉴"].mode()[0]
frequency
food["메뉴"] = food["메뉴"].fillna(frequency)
food

print(food.isnull().sum())
```

Numpy(넘파이)

- **Numpy(Numeric Python)**



  - **대규모 수치 계산과 배열 연산을 빠르게 처리**하기 위한 파이썬 라이브러리입니다.
  - 과학 계산, 데이터 분석, 머신러닝(딥러닝)에 주로 활용됩니다.

수학 관련 함수	설명
np.sum([1, 2, 3])	합계(리스트, 튜플)
np.mean([1, 2, 3])	평균(리스트, 튜플)
np.square(n)	n의 제곱수
np.sqrt(n)	n의 제곱근

Numpy(넘파이)

- **수학 관련 함수**

```
import numpy as np

# 수학 관련 함수
n1 = np.sum([1, 2, 3])  #합계
print(n1)
n2 = np.mean([1, 2, 3, 4]) #평균
print(n2)
n3 = np.square(9) #제곱수
print(n3)
n4 = np.sqrt(4) #제곱근
print(n4)
```

Numpy(넘파이)

- **Numpy(Numeric Python)**

배열 관련 함수	설명
np.array([1, 2, 3])	1차원 배열
np.array([[1, 2], [3,4]])	2차원 배열
np.arange(n)	1차원 배열로 0부터 n-1까지 생성
a.reshape(x, y)	1차원 배열(a)을 2차원 배열로 변환
b.flatten()	2차원 배열(b)을 1차원 배열로 변환
np.zeros()	0으로 만들어진 배열 생성
np.ones()	1로 만들어진 배열 생성

Numpy(넘파이)

- **Numpy(Numeric Python)**

```
# 1차원 배열
x = np.array([1, 2, 3])
print(x)
print(type(x)) # 타입
print(x.shape) # 크기

# 인덱싱 및 슬라이싱
print(x[0])
print(x[2])
print(x[-1])
print(x[0:3])
print(x[:])
print(x[:-1])
```

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

Numpy(넘파이)

- **Numpy(Numeric Python)**

```
# 2차원 배열
d2 = np.array([
    [1, 2, 3],
    [4, 5, 6]
])
print(d2)
print(d2.shape)

# 인덱싱 및 슬라이싱
print(d2[0, 0])
print(d2[1, 1])
print(d2[0:, 0:])
print(d2[1:, 1:])
```

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

Numpy(넘파이)

- **Numpy(Numeric Python)**

```
# arange(시작값, 종료값, 증감값) - (종료값-1)
a = np.arange(10)
print(a)
b = np.arange(0, 10, 1)
print(b)
c = np.arange(0, 10, 2)
print(c)

# 1차원 배열을 2차원 배열
a2 = a.reshape(2,5)
print(a2)

# 2차원을 1차원 배열로
a3 = a2.flatten()
print(a3)
```

```
[0 1 2 3 4 5 6 7 8 9]
[0 1 2 3 4 5 6 7 8 9]
[0 2 4 6 8]
[[0 1 2 3 4]
 [5 6 7 8 9]]
[0 1 2 3 4 5 6 7 8 9]
```

Numpy(넘파이)

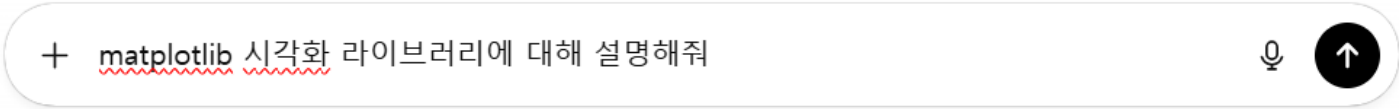
- **Numpy(Numeric Python)**

```
# 결측치(nan)
a = np.nan
print(a)
print(type(a))

# 0 생성
a1 = np.zeros((3, 3))
print(a1)

# 1 생성
a2 = np.ones((2, 3))
print(a2)

a3 = np.ones((2, 3), dtype=int)
print(a3)
```

```
nan
<class 'float'>
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]]
[[1 1 1]
 [1 1 1]]
```

시각화 도구 - matplotlib

- **matplotlib 라이브러리**



  - 데이터 분석, 과학 시각화, 보고서 작성 등 다양한 분야에서 그래프를 그릴 때 사용됩니다.
  - 주요 그래프 종류

종류	함수명	예시
선 그래프	plot()	plt.plot(x, y)
막대 그래프(수직/수직)	bar() / barh()	plt.bar(x, y) / plt.barh(x, y)
산점도	scatter()	plt.scatter(x, y)
히스토그램	hist()	plt.hist(data)
파이차트	pie()	plt.pie(data)

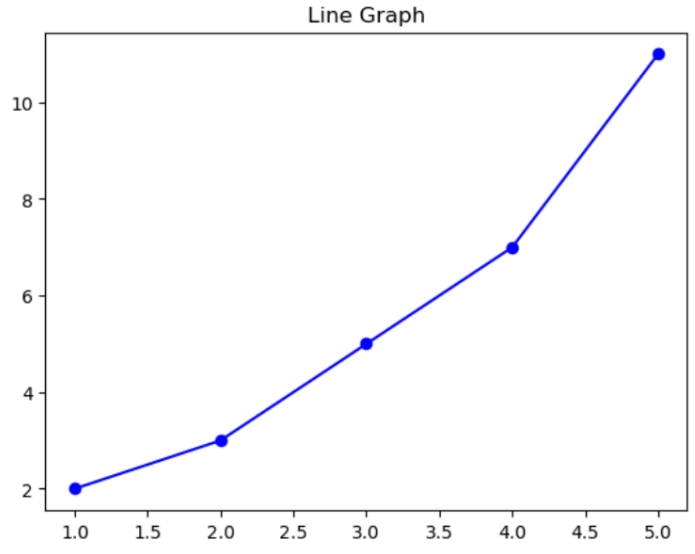
시각화 도구 – matplotlib

- **차트 그리기 – 기본 구조**

요소	설명
`plt`	pyplot 모듈 (그래프 그리기 담당)
`plot()`	그래프 생성
`show()`	화면에 출력

시각화 도구 - matplotlib

- 선 그래프

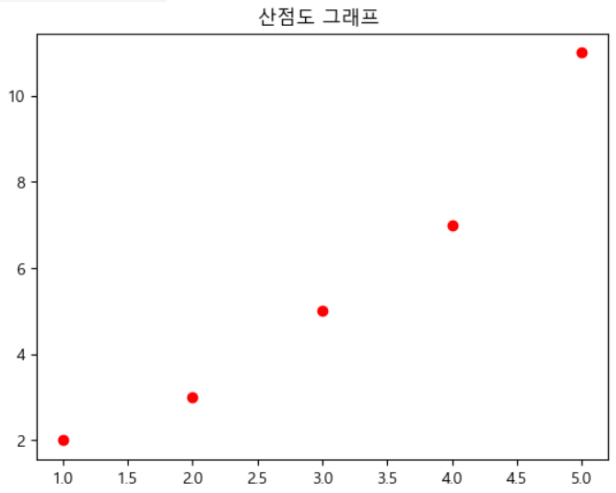



```
import matplotlib.pyplot as plt

# 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

- 산점도 그래프

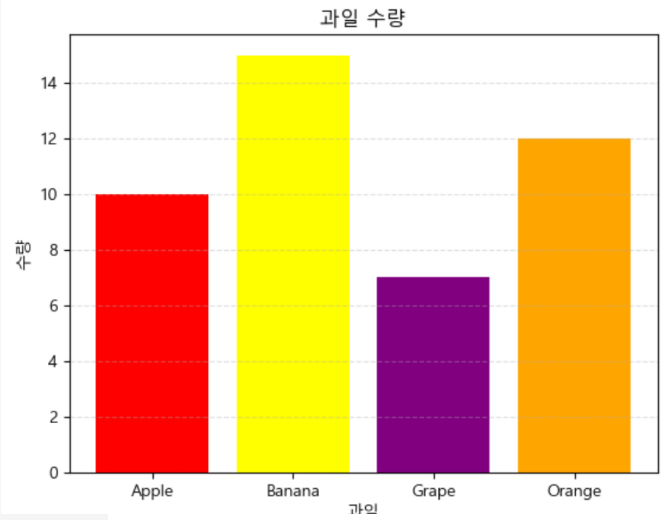
```
import matplotlib.pyplot as plt

# 데이터 준비
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]  #소수

plt.rc('font', family='Malgun Gothic')
plt.title('산점도 그래프 ')
plt.scatter(x, y, color='red')
plt.show()
```



시각화 도구 - matplotlib

- 막대 그래프

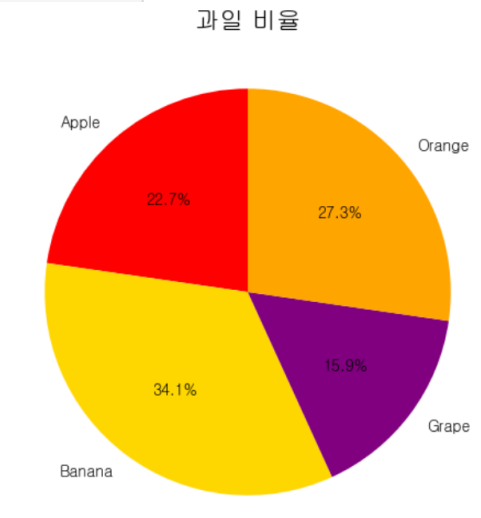
```python
import matplotlib.pyplot as plt

# 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()
```

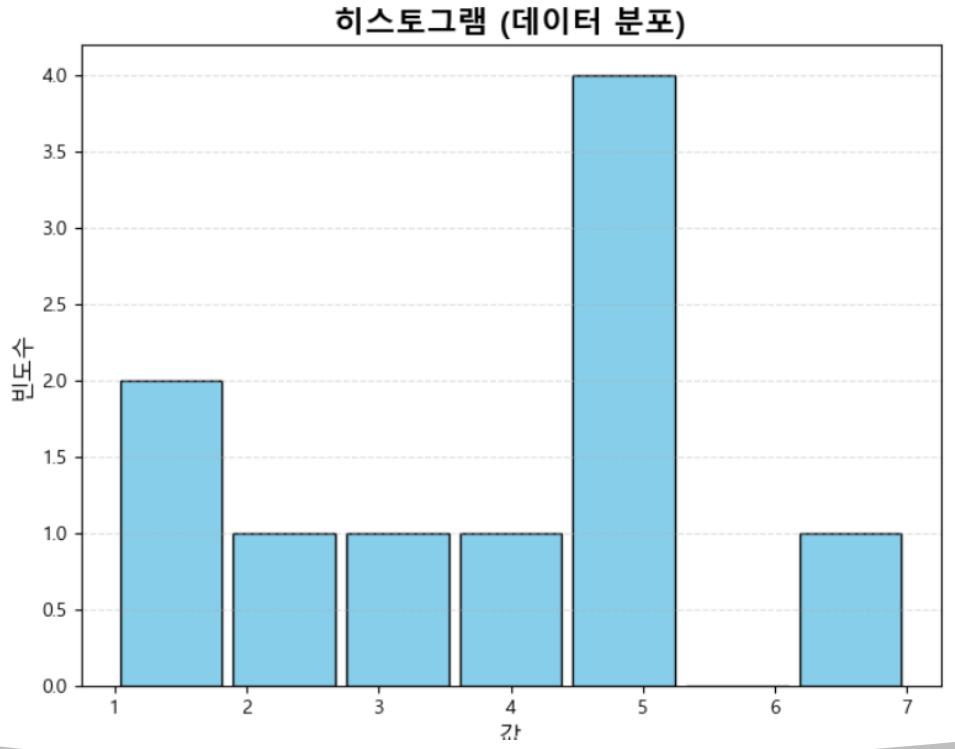


시각화 도구 - matplotlib

- 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()
```

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

```python
# 결측치 처리
# df.isnull().sum()
print(df.isna().sum())

# age, embarked, embark_town 칼럼 - 결측치 대체
# deck 칼럼 - 결측치 삭제
df2 = df.copy() #원본 복사

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

# 평균값으로 채우기
df2['age'] = df2['age'].fillna(mean_age)

# embarked - 최빈값 계산
most_freq = df2['embarked'].mode()[0]
print(most_freq) #S

df2['embarked'] = df2['embarked'].fillna(most_freq)
print(df2.isna().sum())
```

```
survived          0
pclass            0
sex               0
age               0
sibsp             0
parch             0
fare              0
embarked          0
class             0
who               0
adult_male        0
deck            688
embark_town       2
alive             0
alone             0
dtype: int64
```

타이타닉호 데이터 분석 및 처리

- 타이타닉호 데이터 분석 및 처리

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

most_freq = df2['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])

# deck열 삭제
df2 = df2.drop('deck', axis=1)
df2.info()
print(df2.isna().sum())
df2.columns # 칼럼 확인
```

```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
embark_town    0
alive          0
alone          0
dtype: int64
```

seaborn 라이브러리

- **타이타닉호 - 성별에 따른 생존자 수 그래프**

```python
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**