Pandas의 데이터 구조와 데이터 타입

한국기술교육대학교
온라인평생교육원

학습목표

- Pandas의 DataFrame의 구조와 데이터 타입을 이해할 수 있다.
- Pandas의 DataFrame이 제공하는 API와 통계 함수를 활용할 수 있다.

학습내용

- Pandas의 DataFrame 구조와 데이터 타입
- Pandas의 DataFrame API

Pandas의 DataFrame 구조와 데이터 타입

1 DataFrame

1 개요

- Pandas가 제공하는 데이터 타입

형태	▪ 2차원 배열 형태 ▪ 각 열이 각기 다른 타입으로 구성
장점	▪ 별도의 구조체나 클래스를 만들지 않고도 사용자 데이터를 취급하기 쉬움 ▪ 많은 통계 관련 함수 제공 ▪ 각 열과 행에 대해서 자유롭게 접근 가능 ▪ 외부 파일을 읽어 DataFrame으로 손쉽게 전환

Pandas의 DataFrame 구조와 데이터 타입

1 DataFrame

2 Pandas가 제공하는 파일 형식

파일 형식	설명	Read	Write
csv	✓ text 형태로 데이터 저장 ✓ 데이터와 데이터 사이에 구분자를 이용해 저장 ✓ 메모장에서도 작성 가능	read_csv	to_csv
excel	✓ 엑셀 프로그램 필요	read_excel	to_excel
json	✓ Javascript 객체 저장 형식으로 데이터 저장 ✓ 웹 등을 이용해 데이터를 주고받기 위해 사용하는 형식	read_json	to_json
html	✓ 웹 페이지 파일 형식	read_html	to_html
hd5	✓ 딥러닝에서의 모델 저장 시 사용하는 형식	read_hd5	to_hd5

Pandas의 DataFrame 구조와 데이터 타입

1 DataFrame

3 csv파일 읽고 쓰기

- 데이터를 쉼표(,)로 구분
- 텍스트 파일의 형식
- 빅데이터에서 가장 많이 사용하는 데이터 저장 형태

장점

- 엑셀에서 수정 작업 진행 후 다시 저장 가능
- 특정 프로그램 필요 없이 보통의 에디터로 작성 용이

- Pandas를 이용한 csv파일 읽기

```
Import pandas as pd
data = pd.read_csv("./data/score.csv")
```

자주 사용하는 옵션 : 헤더(Header)

- 제목 줄이 없는 경우 : None
- 제목 줄이 첫 줄에 없고 다른 줄에 있는 경우 : 그 줄을 명시

Pandas의 DataFrame 구조와 데이터 타입

1 DataFrame

4 경로 표현

- 파일의 경로는 절대 경로와 상대 경로 모두 가능

경로 기술 시 주의사항

- 경로 기술 시 c:₩pandas_workspace₩data₩score.csv처럼 모든 경로를 써줘야 함
  - ➡ 파이썬에서 ₩는 다른 문자와 결합하여 경로를 제대로 인식하지 못하는 문제 발생

문제 해결 방법

- ₩를 하나 더 붙여서 **₩₩로 작성**

```
c:₩₩pandas_workspace₩₩data₩₩score.csv
```

- **경로 앞에 r**을 붙여 ₩문자의 기능을 무력화시킴

```
r"c:₩pandas_workspace₩data₩score.csv"
```

- ₩(역슬래쉬) 대신 **/(슬래쉬) 사용**
  - ➡ 윈도우 OS는 경로 표현 시 ₩를 사용, 리눅스 OS는 /를 사용했으나 요즘은 윈도우 OS도 /를 지원

Pandas의 DataFrame 구조와 데이터 타입

1 DataFrame

4 경로 표현

1 상대 경로

- 현재 애플리케이션이 가동 중인 폴더를 기준으로 경로를 배정

- . (도트 한 개) : 현재 어플리케이션이 가동 중인 폴더
- ..(도트 두 개) : 자신보다 하나 위의 폴더

uni_10 data

파이썬 파일 존재

./data/score.csv 형태로 기술

Pandas의 DataFrame 구조와 데이터 타입

1 DataFrame

4 경로 표현

2 절대 경로

- - /(루트)부터 모든 경로를 기술

c:/pandas_workspace/uni_10/data/score.csv 형태로 기술

↳ 생략 가능

절대 경로

- 폴더 이동 시 모든 경로를 다시 작성해야 함

상대 경로

- 폴더 이동 시 경로를 수정할 필요가 없음

➡ 절대 경로보다는 상대 경로를 사용하는 것이 더 좋음

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

1 형태

/data/score.csv

	A	B	C	D
1	name	kor	eng	mat
2	홍길동	90	90	90
3	임꺽정	80	80	80
4	장길산	70	70	70
5	홍경래	90	80	70
6	이징옥	60	50	50

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

2 외부파일(csv형식)읽기 예제

```
#파일명 : exam10_1.py

import pandas as pd

data = pd.read_csv("./data/score.csv")
#상대 경로로 파일을 읽었다면 현재 폴더 아래에 data 폴더가 있어야 하고 그 폴더 아래에 score.csv파일이 있어야 함

print( " 컬럼명 :  " , data.columns)
print( " 인덱스 :  " , data.index)

#총점, 평균 구하기 : 새로운 필드 추가
data[ ' total ' ] = data[ ' kor ' ] + data[ ' eng ' ]+data[ ' mat ' ]
data[ ' avg ' ] = data[ ' total ' ]/3

print(data )
```

```
컬럼명 :  Index(['name', 'kor', 'eng', 'mat'], dtype='object')
인덱스 :  RangeIndex(start=0, stop=5, step=1)
  name  kor  eng  mat  total        avg
0  홍길동   90   90   90    270  90.000000
1  임꺽정   80   80   80    240  80.000000
2  장길산   70   70   70    210  70.000000
3  홍경래   90   80   70    240  80.000000
4  이징옥   60   50   50    160  53.333333
```

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

3 제목줄이 없는 경우

/data/score_noheader.csv

- 파일을 읽을 때 header=None을 주고 별도로 컬럼명을 부여

	A	B	C	D
1	홍길동	90	90	90
2	임꺽정	80	80	80
3	장길산	70	70	70
4	홍경래	90	80	70
5	이징옥	60	50	50

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

3 제목줄이 없는 경우

```
#파일명 : exam10_2.py

import pandas as pd

#제목줄이 없는 경우, 헤더 옵션을 None으로 줌
data = pd.read_csv( " ./data/score_noheader.csv " ,
header=None)

print( " 컬럼명 :  " , data.columns)
print( " 인덱스 :  " , data.index)

#직접 컬럼 부여
print( " 컬럼부여후 ---------- " )
data.columns = [ ' name ' ,  ' kor ' ,  ' eng ' ,  ' mat ' ]
print( " 컬럼명 :  " , data.columns)
print(data)

#총점, 평균 구하기
print("필드추가후")
data['total'] = data['kor'] + data['eng']+data['mat']
data['avg'] = data['total']/3

print(data )
```

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

3 제목줄이 없는 경우

```
컬럼명 :  Int64Index([0, 1, 2, 3], dtype='int64')
인덱스 :  RangeIndex(start=0, stop=5, step=1)
컬럼부여후 ----------
컬럼명 :  Index(['name', 'kor', 'eng', 'mat'], dtype='object')
  name  kor  eng  mat
0  홍길동   90   90   90
1  임꺽정   80   80   80
2  장길산   70   70   70
3  홍경래   90   80   70
4  이징옥   60   50   50
필드추가후
  name  kor  eng  mat  total        avg
0  홍길동   90   90   90    270  90.000000
1  임꺽정   80   80   80    240  80.000000
2  장길산   70   70   70    210  70.000000
3  홍경래   90   80   70    240  80.000000
4  이징옥   60   50   50    160  53.333333
```

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

4 데이터 가공 결과 저장

/data/score_header.csv

- 실제로 제목줄이 있는 라인 수 지정
  header= 3

저장

- DataFrame에서 제공하는 to_csv함수를 사용
- csv파일을 엑셀에서 보고 싶다면 별도의 인코딩 작업 필요(문자셋 cp949 사용)

	A	B	C	D
1	성적파일입니다			
2				
3				
4	name	kor	eng	mat
5	홍길동	90	90	90
6	임꺽정	80	80	80
7	장길산	70	70	70
8	홍경래	90	80	70
9	이징옥	60	50	50

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

4 데이터 가공 결과 저장

```
#파일명 : exam10_3.py

import pandas as pd

#헤더가 3번째 줄에 있음
data = pd.read_csv("./data/score_header.csv", header=3)
print(data)

print("컬럼명 : ", data.columns)
print("인덱스 : ", data.index)

#총점, 평균 구하기
data['total'] = data['kor'] + data['eng']+data['mat']
data['avg'] = data['total']/3

print(data )

#엑셀이 인코딩 cp949를 사용하므로 한글이 깨지지 않도록 하려면 cp949로 인코딩을 해야 함
index = False 옵션이 없으면 index까지 저장되면서 필드가 늘어남
data.to_csv("score_result.csv", mode='w', encoding="cp949", index=False)
```

(계속)

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

4 데이터 가공 결과 저장

#현재 작업 폴더에서 직접 score_result.csv파일 열어보기

파일명 : score_result.csv

	A	B	C	D	E	F
1	name	kor	eng	mat	total	avg
2	홍길동	90	90	90	270	90
3	임꺽정	80	80	80	240	80
4	장길산	70	70	70	210	70
5	홍경래	90	80	70	240	80
6	이징옥	60	50	50	160	53.33333

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

5 엑셀 파일 읽기

- 엑셀 파일도 손쉽게 처리할 수 있음
- 별도의 Win32 Component 라이브러리나 다른 라이브러리를 설치할 필요가 없음

→ Openpyxl 등

MS 제공,
32비트에서만 사용 가능

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

5 엑셀 파일 읽기

```
#파일명 : exam10_4.py

import pandas as pd

data = pd.read_excel( " ./data/score.xlsx " )

data[ ' total ' ] = data[ ' kor ' ] + data[ ' eng ' ]+data[ ' mat ' ]
data[ ' avg ' ] = data[ ' total ' ]/3
print(data)

data.to_excel( " score_result1.xlsx " )
data.to_excel( " score_result2.xlsx " , index=False)
```

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

5 엑셀 파일 읽기

- score_result1.xlsx

- index=False 옵션을 주지 않을 경우 index 필드도 출력됨

	A	B	C	D	E	F	G
1		name	kor	eng	mat	total	avg
2	0	홍길동	90	90	90	270	90
3	1	임꺽정	80	80	80	240	80
4	2	장길산	70	70	70	210	70
5	3	홍경래	90	80	70	240	80
6	4	이징옥	60	50	50	160	53.33333
7	5	일지매	90	90	0	180	60

- score_result2.xlsx

- index=False 옵션을 줄 경우 index 필드는 출력되지 않음

	A	B	C	D	E	F	G
1	name	kor	eng	mat	total	avg	
2	홍길동	90	90	90	270	90	
3	임꺽정	80	80	80	240	80	
4	장길산	70	70	70	210	70	
5	홍경래	90	80	70	240	80	
6	이징옥	60	50	50	160	53.33333	
7	일지매	90	90	0	180	60	

Pandas의 DataFrame 구조와 데이터 타입

2 데이터 타입

6 DataFrame 파일 읽기 옵션

옵션	설명
path	✓ 파일의 위치(파일명 포함) url
sep(delimiter)	✓ csv 파일의 구분자가 ,(콤마)가 아니라 다른 문자일 경우 다른 문자 지정 가능
header	✓ 열 이름으로 사용할 행의 번호
index_col	✓ 행 인덱스로 사용할 열의 번호 또는 열의 이름
names	✓ 열 이름으로 사용할 문자열 리스트
skiprows	✓ 처음 몇 줄을 건너뛸지 지정 ✓ 제목줄이 첫 번째 줄에 없을 때, 헤더도 사용 가능

Pandas의 DataFrame API

1 DataFrame이 제공하는 API의 활용

DataFrame

- 데이터 분석을 위해 많은 API를 제공
- 파이썬 클래스로 만들어짐

DataFrame 크기 | 데이터 구성 항목 | 자료형 | 통계 수치

실습용 데이터 정보

- 한 대학의 머신러닝 데이터

자동차 연비 | 실린더 수 | 배기량 | 출력 | 차중

가속 능력 | 출시 년도 | 제조국 | 모델명

- 예제 파일은 data 폴더 안에 있음

Pandas의 DataFrame API

1 DataFrame이 제공하는 API의 활용

head()

- DataFrame의 앞에서부터 5개의 데이터를 보여줌
- 인수를 직접 지정하여 n개의 데이터를 확인할 수 있음
- 데이터 분석용 파일은 데이터 개수가 커서, 모든 내용을 한 번에 확인하기 어려움 ➡ head 함수 등을 이용해 데이터의 대략적 성격 파악 가능

```
data.head(3)
```

tail()

- DataFrame의 뒤에서부터 5개의 데이터를 보여줌

```
data.tail(3)
```

shape

- DataFrame 행과 열의 크기를 알려줌
- tuple 타입을 전달함

```
row , col = data.shape
```

Pandas의 DataFrame API

1 DataFrame이 제공하는 API의 활용

```
#파일명 : exam10_5.py

import pandas as pd

#헤더가 3번째 줄에 있음
data = pd.read_csv("./data/auto-mpg.csv")

print("앞에서부터 5개만 미리 보기")
print(data.head() )

print("뒤에서부터 5개만 미리 보기")
print(data.tail() )

print("앞에서부터 10개만 미리 보기")
print(data.head(10))
print(data.shape )  #DataFrame의 차원 행, 열의 개수 확인 가능

row, col = data.shape
#tuple 타입으로 행과 열에 대한 정보를 모두 가지고 있음
print("행의 개수 ", row)
print("열의 개수 ", col)
```

Pandas의 DataFrame API

1 DataFrame이 제공하는 API의 활용

- 결과 화면

```
앞에서 부터 5개만 미리 보기
    mpg  cylinders  displacement  horsepower  weight  acceleration  model-year
0  18.0          8         307.0       130.0    3504          12.0          70
1  15.0          8         350.0       165.0    3693          11.5          70
2  18.0          8         318.0       150.0    3436          11.0          70
3  16.0          8         304.0       150.0    3433          12.0          70
4  17.0          8         302.0       140.0    3449          10.5          70
뒤에서 부터 5개만 미리 보기
      mpg  cylinders  displacement  horsepower  weight  acceleration  model-year
393  27.0          4         140.0        86.0    2790          15.6          82
394  44.0          4          97.0        52.0    2130          24.6          82
395  32.0          4         135.0        84.0    2295          11.6          82
396  28.0          4         120.0        79.0    2625          18.6          82
397  31.0          4         119.0        82.0    2720          19.4          82
앞에서 부터 10개만 미리 보기
    mpg  cylinders  displacement  horsepower  weight  acceleration  model-year
0  18.0          8         307.0       130.0    3504          12.0          70
1  15.0          8         350.0       165.0    3693          11.5          70
2  18.0          8         318.0       150.0    3436          11.0          70
3  16.0          8         304.0       150.0    3433          12.0          70
4  17.0          8         302.0       140.0    3449          10.5          70
5  15.0          8         429.0       198.0    4341          10.0          70
6  14.0          8         454.0       220.0    4354           9.0          70
7  14.0          8         440.0       215.0    4312           8.5          70
8  14.0          8         455.0       225.0    4425          10.0          70
9  15.0          8         390.0       190.0    3850           8.5          70
(398, 7)
행의 개수  398
열의 개수  7
```

Pandas의 DataFrame API

1 DataFrame이 제공하는 API의 활용

info ()

- DataFrame 객체 안에 어떤 필드를 가졌는지 구조 파악 가능
- 데이터 개수, 인덱스, 각 컬럼별 데이터 타입 확인 가능

```
데이터의 기본 구조
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 7 columns):
mpg             398 non-null float64
cylinders       398 non-null int64
displacement    398 non-null float64
horsepower      396 non-null float64
weight          398 non-null int64
acceleration    398 non-null float64
model-year      398 non-null int64
dtypes: float64(4), int64(3)
memory usage: 21.8 KB
None
```

〈 auto-mpg.csv 파일의 예 〉

Pandas의 DataFrame API

1 DataFrame이 제공하는 API의 활용

describe

- 필드별 카운트, 평균, 최솟값, 최댓값, 표준편차, 4분위 수(1/4, 2/4, 3/4), 확인 가능
- 파이썬을 이용한 데이터 분석 시, 대략적인 데이터 파악을 도와주는 함수
- 필드별로 통계에서 자주 사용되는 통계량을 보여줌

```
데이터의 요약정보 확인
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model-year
count	398.000000	398.000000	398.000000	396.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.189394	2970.424623	15.568090	76.010050
std	7.815984	1.701004	104.269838	38.402030	846.841774	2.757689	3.697627
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000
50%	23.000000	4.000000	148.500000	92.000000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	262.000000	125.000000	3608.000000	17.175000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

Pandas의 DataFrame API

1 DataFrame이 제공하는 API의 활용

```
#파일명 : exam10_6.py

import pandas as pd

#헤더가 3번째 줄에 있음
data = pd.read_csv("./data/auto-mpg.csv")

print("데이터의 기본 구조")
print(data.info() )

print("데이터의 요약 정보 확인")
print(data.describe())
```

Pandas의 DataFrame API

2 조건 부여하기

1 조건연산자 사용하기

- DataFrame 객체는 데이터를 검색할 때 특정 조건을 부여하기 쉬움
- 별도의 for문을 필요로 하지 않음

data[조건식]	▪ 조건식의 결과가 참인 경우의 데이터만 반환

data[data['필드명'] == "값"] 또는
data [data.필드명 == "값"]

- 조건식은 == 뿐만 아니라 관계연산자(==, !=, >, <, >=, <=) 모두 사용 가능

- 조건이 두 개 이상 결합할 때는 파이썬이 제공하는 논리연산자인 and(&) 또는 or(|) 연산자를 사용하면 안 됨

- 파이썬 논리연산자를 사용할 경우 다음 에러 발생

```
ValueError: The truth value of a Series is ambiguous.
Use a.empty, a.bool(), a.item(), a.any() or a.all().
```

Pandas의 DataFrame API

2 조건 부여하기

1 조건연산자 사용하기

- 조건식을 결합해야 할 경우 Numpy 라이브러리 사용
  ➡ logical_and 메서드와 logical_or 메서드를 대신 사용

벡터 연산을
지원하기 위한
수학 라이브러리

```
import numpy as np
data [ np.logical_and(data['model-year']==70,
data['mpg']>=25) ]
```

Pandas의 DataFrame API

2 조건 부여하기

2 주의할 점

파이썬 사용이 가장 많은 영역

머신러닝

딥러닝

- 머신러닝, 딥러닝 라이브러리 ➡ C언어로 되어 있음

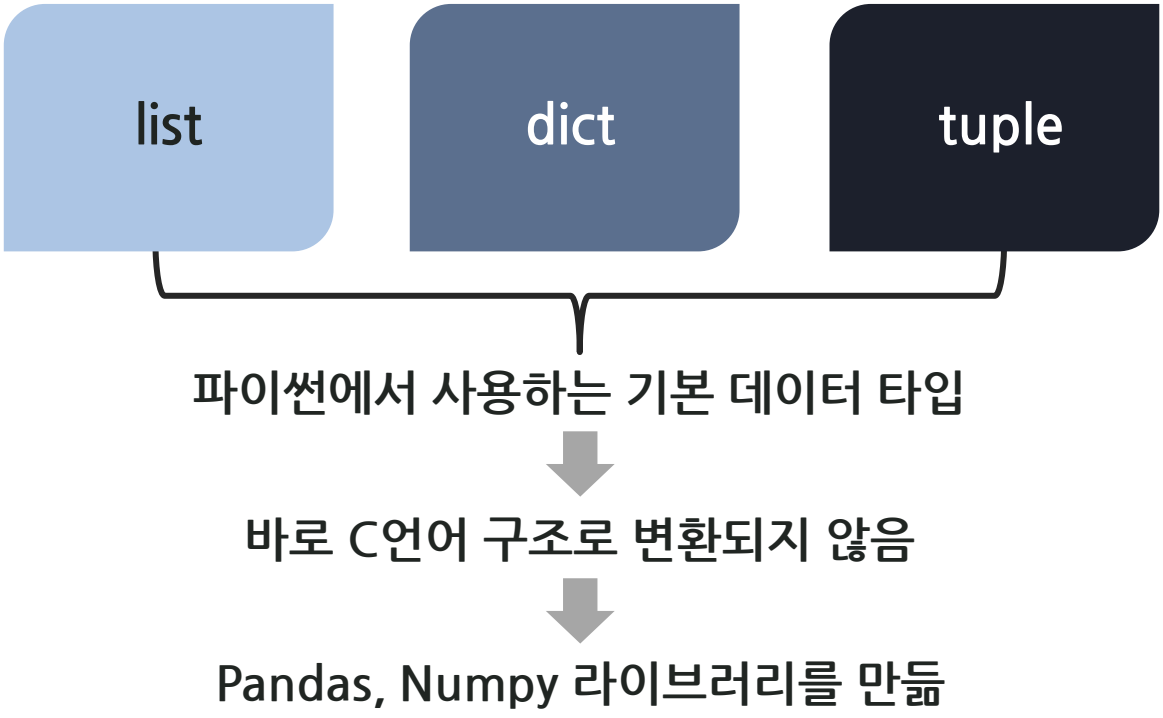
파이썬 사용 이유

- C언어와의 결합력이 우수
- C언어를 습득하는 것보다는 파이썬을 습득하는 게 더 빠름
- 코딩을 쉽게 할 수 있음

Pandas의 DataFrame API

2 조건 부여하기

2 주의할 점



Pandas의 DataFrame API

2 조건 부여하기

3 예제

```
#파일명 : exam10_7.py

import pandas as pd

data = pd.read_csv("./data/auto-mpg.csv")

print("조건식 적용하기 ")
print(data[data.cylinders==4] ) #실린더 개수 4개 짜리만

#연비 27 이상만
print(data[data.mpg>=27] )

#모델 연도 70년, 연비 27 이상만 : -(하이픈) 때문에 data.model-year는 안 됨
아래처럼 작성 혹은 컬럼명 수정
print ( data[data['model-year']==70])

#두 가지 조건을 동시에 주고 싶을 때 아래처럼 작성 시 에러 발생
#print(data[ data['model-year']==70 or data['mpg']>=25 ])
#ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(), a.item(), a.any() or a.all().

두 가지 조건을 동시에 주고 싶을 때
import numpy as np
print(data [np.logical_and(data['model-year']==70, data['mpg']>=25)])
```

Pandas의 DataFrame API

3 DataFrame이 제공하는 통계 함수

- DataFrame 객체의 각 열 : Series 타입, 1차원, 파이썬의 List구조와 유사
- 각 열의 평균이나 중간값, 최댓값, 최솟값 등은 Numpy 라이브러리를 사용하지 않더라도, Series 타입을 기본적으로 제공

max	▪ 해당 필드의 최댓값 반환
min	▪ 해당 필드의 최솟값 반환
std	▪ 표준편차 반환 ▪ 표준편차가 크면 집단의 평균이 같더라도, 최댓값과 최솟값의 차가 커서 값이 넓게 분포되어 있음을 의미

Pandas의 DataFrame API

3 DataFrame이 제공하는 통계 함수

var	▪ 분산, 집단 내 값의 흩어짐 척도 ▪ 파이썬은 표준 분산을 사용하지 않음
quantile	▪ 집단 내 1/4수, 2/4수, 3/4수 등 4분위 수를 알아보기 위해 사용 ▪ quantile(0.25), quantile(0,5), quantile(0.75)
median	▪ 집단 내에서 중간에 위치한 값을 알아내고자 할 때 사용 ▪ 집단 내 값들의 편차가 크면 평균값만으로 특정 집단의 특성을 나타내기 어려운 경우가 많음 ➡ 중간값을 사용

Pandas의 DataFrame API

3 DataFrame이 제공하는 통계 함수

```
#파일명 : exam10_8.py
import pandas as pd

data = pd.read_csv("./data/auto-mpg.csv")

print(data['model-year'].value_counts())
#value_counts 데이터별 고유 카운트

#평균, 최대, 최소
print("연비 평균 : ", data['mpg'].mean())
print("연비 최대 : ", data['mpg'].max())
print("연비 최소 : ", data['mpg'].min())
print("연비 중간 : ", data['mpg'].median())
print("연비 분산 : ", data['mpg'].var())
print("연비 표준편차 : ", data['mpg'].std())

print("1사분위수 : ", data['mpg'].quantile(0.25))
print("2사분위수 : ", data['mpg'].quantile(0.5))
print("3사분위수 : ", data['mpg'].quantile(0.75))
```

학습정리

1. Pandas의 DataFrame 구조와 데이터 타입

- Pandas에서 제공하는 데이터 구조 중에 DataFrame이 가장 많이 쓰임
- Pandas는 csv나 excel, json 등의 파일을 읽어서 바로 DataFrame 객체로 전달해 줌
- DataFrame 객체는 for문을 사용하지 않고 조건식으로 인덱스처럼 데이터에 접근할 수 있음
- DataFrame 객체의 인덱스에 논리식을 사용할 경우 파이썬의 논리식이 아닌 Numpy에서 제공하는 논리함수를 사용해야 함

학습정리

2. Pandas의 DataFrame API

- head() : DataFrame의 앞에서부터 5개의 데이터를 보여줌
- tail() : DataFrame의 뒤에서부터 5개의 데이터를 보여줌
- info() : DataFrame 내의 각 필드의 종류와 데이터 타입, 데이터 개수 등의 정보를 보여줌
- describe() : 기본 통계값인 max, min, mean,quantile(퍼센트), std 등을 적용한 통계량을 간략하게 보여줌