과정명		모듈명		회차명	

◈ 학습열기

이전에는 데이터라는 것이 한정된 공간과 한정된 사람들의 것이었습니다. 오늘날은 더 많은 사람이 더 많은 양의 데이터를 서로 나누거나 공유합니다.
데이터를 공유하는 이유는 하나의 사업체나 개인 혼자서 모든 정보를 수집하기 어렵기때문에 서로 데이터를 주고 받으면서 더 많은 정보나 얻어서 사업에 이용하거나 새로운 사업을 창출하기 위해서 입니다. 이전에는 데이터 공유는 액티브 액스와 같은 작성하기 어려운 어플리케이션을 활용했다면 요즘에는 웹이라는 자원을 활용하여 서로 데이터만 주고 받는 형태로 데이터 공유가 이루어집니다.
판다스 라이브러리는 외부 파일을 불러와 데이터 프레임을 만들기 쉽게 되어 있습니다. 물론 데이터를 추가하거나 수정 삭제등의 변화를 가하여 특정 형식의 외부파일로 저장도 손쉽게 되어 있습니다.
빅데이터분석에서의 통계 자료들을 다룰때 보통 csv나 엑셀형식의 파일들을 많이 다루는데, 판다스는 특히 이 분야에 최적화 되어 있습니다 .
이 장에서는 외부파일들을 읽어서 데이터프레임 객체를 만들고, 데이터프레임 객체 자체에서 제공하는 많은 함수들의 사용법을 살펴보도록 하겠습니다.

내레이션	학습자가 학습 화면에 구성한 내용을 보고 있다고 가정하고, 학습 설명으로 함께 들어야 할 음성 내용을 자세하게 기입해 주세요. **➔ 이건 나래이션해요? 저도 이런건 나래이션하면 좋겠어요..ㅜㅜ**

◈ 학습내용

- Pandas 의 데이터프레임 구조와 데이터 타입
- Pandas 를 이용한 외부파일 읽고 쓰기

◈ 학습목표

- 데이터프레임이 구조를 이해할 수 있다
- 판다스를 이용하여 외부파일을 읽고 쓸 수 있다

내레이션

(기입하지 않습니다.)

간지 부분

◈ 데이터프레임

1. 외부파일 읽고 쓰기

내레이션

(기입하지 않습니다.)

◈ 데이터프레임

- - 데이터프레임은 판다스가 제공하는 데이터 타입입니다.

- 형태는 2차원 배열형태이나, 하나의 타입으로만 구성되는 것이 아니라 각 열이 각기 다른 타입으로 구성되어있어서 , 별도의 구조체나 클래스를 만들지 않고도 사용자 데이터를 취급하기 쉬운 객체입니다 .

- 또한 많은 통계관련 함수들도 제공하고 있고 각 열과 행에 대해서 자유롭게 접근할 수 있다는 것도 데이터프레임의 장점입니다.

내레이션

◈판다스는 외부파일을 읽어서 데이터 프레임으로 손쉽게 전환합니다.

- 판다스가 제공하는 파일 형식들

파일형식	설명	read	write
csv	text형태로 데이터를 저장한다. 데이터와 데이터 사이에 ,나 구분자를 이용해서 저장한다. 메모장에서도 작성 가능하다	read_csv	to_csv
excel	엑셀 프로그램이 있어야 한다	read_excel	to_excel
json	javascript 객체 저장형식으로 데이터를 저장한다. 웹 등을 이용해 데이터를 주고 받기 위해 사용하는 형식이다	read_json	to_json
html	웹페이지 파일 형식이다	read_html	to_html
hd5	딥러닝에서의 모델을 저장할때 사용하는 형식이다	read_hd5	to_hd5

내레이션

◈ csv파일 읽고 쓰기

- csv 파일은 데이터를 쉼표(,) 로 구분하고, 텍스트파일의 형식이기때문에 특정 프로그램이 필요없이 보통의 에디터
- 로 작성하기 쉽다는 장점이 있습니다.
- 엑셀에서도 열어서 수정작업 진행후 다시 저장할 수 도 있어서 빅데이에서 가장 많이 사용하는 데이터 저장형태입니다

- 판다스를 이용한 csv 파일 읽기

```
import pandas a pd
data = pd.read_csv(“./data/score.csv”)
```

자주 사용하는 옵션
header - 제목줄이 없으면 None, 제목줄이 첫줄에 없고 다른줄에 있을 경우 그 줄을 명시한다

내레이션

◈ 경로표현

파일의 경로는 절대경로와 상대경로 모두 가능합니다. 경로를 기술할 경우 주위할 점은 ₩ 절대경로를 처리 경우에는 c:₩pandas_workspace₩data₩score.csv 처럼 모든 경로를 다 써줘야 하는데 문제는 파이썬에 ₩는 다른 문자와 결합하여 제어문자로 사용된다는데 있습니다. 이럴경우에는 경로를 제대로 인식하지 못해 문제가 될 수 도 있습니다. 파이썬에 이를 해결하는 방법은 두가지가 있습니다.

1) ₩ 를 하나 더 붙여서 ₩₩ 로 해야 합니다.
c:₩₩pandas_workspace₩₩data₩₩score.csv

2) 경로앞에 r을 쓰면 ₩문자가 가지고 있는 기능이 무력화 됩니다.
r"c:₩pandas_workspace₩data₩score.csv"

3)마지막은 ₩(역슬래쉬) 대신에 /(슬래쉬)를 사용하면 됩니다.
윈도우os 는 경로를 나타낼때 ₩(역슬래쉬)를 사용하고 리눅스 os는 /(슬래쉬)를 사용했는데 요즘은 윈도우 os가 /(슬래쉬)도 지원합니다.

내레이션

◈ 상대경로와 절대경로

1. 상대경로
- 상대경로는 현재 어플리케이션이 가동중인 폴더를 기준으로 경로를 배정합니다.
. (도트) : 현재 어플리케이션이 가동중인 폴더
..(도트 두개) : 자신보다 하나 위의 폴더를 나타냅니다.

현재 파이썬 파일이 있는 폴더가 uni_10 인데 이 폴더 안에 data 폴더가 있을 경우에는
./data/score.csv 형태로 기술하는것을 상대경로라 합니다.

2. 절대경로는 /(루트)부터 모든 경로를 기술하는것을 말합니다 .
예) c:/pandas_workspace/uni_10/data/score.csv c:는 생략해도 됩니다.

절대경로 보다는 상대경로를 사용하는 것이 더 좋습니다. 절대 경로를 사용할 경우 폴더 이동시 모든 경로를 다시 작성해야 하는데 반해서 상대경로는 폴더를 이동해도 굳이 경로를 수정할 필요가 없습니다 .

내레이션

◈데이터 파일

/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

내레이션

◈ csv읽기 예 1

```
#파일명 : exam10_1.py

import pandas as pd

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

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

#총점, 평균 구하기 – 새로운 필드 추가하기
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
```

내레이션

◈데이터 파일2-제목줄이 없을 경우

/data/score_noheader.csv

	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

제목줄이 없을경우에는 파일을 읽을때 header=None을 주고 별도로 컬럼명을 부여하면 됩니다.

내레이션

◈ csv읽기 예 2

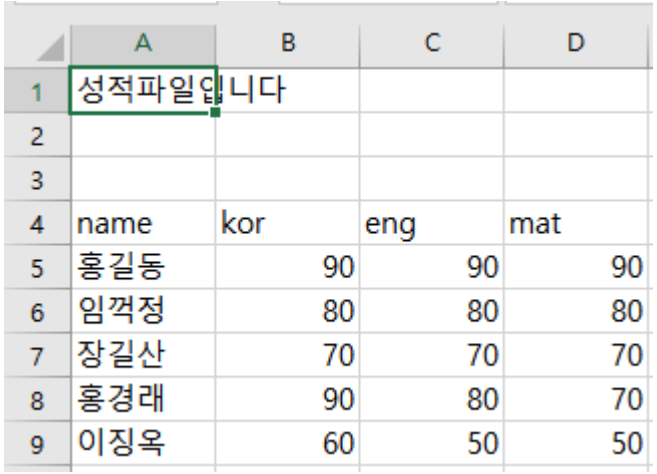
```
#파일명 : exam10_2.py

import pandas as pd

#제목줄이 없을 경우에 heade옵션을 None 으로 주면 된다.
data = pd.read_csv( " ./data/score_noheader.csv " , header=None)

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

#직접 컬럼을 부여한다
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 )
```

```
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
```

내레이션

◈데이터 파일3- 첫번째 줄에 제목줄이 있지 않은 경우

/data/score_header.csv



	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

제목줄이 첫번째 줄에 없을 경우에는 실제로 제목줄이 있는 라인수를 지정하면 됩니다.
header= 3

작업내용을 별도의 파일로 저장해보자
저장은 dataframe에서 제공하는 to_csv함수를 사용한다. 저장시 만일 csv파일을 엑셀에서 보고싶다면 엑셀은 문자셋을 cp949를 사용하므로 별도의 encoding작업을 해줘야 한다

내레이션

◈ csv읽기 예3

```
#파일명 : exam10_3.py

import pandas as pd

#header가 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 )

#excel이 인코딩을 cp949를 사용한다. 한글 안깨지고 엑셀서 열어보려면 cp949로 인코딩을 해야 한다
#index = False 옵션이 없으면 index까지 저장되면서 필드가 늘어난다.
data.to_csv("score_result.csv", mode='w', encoding="cp949", index=False)

#파일을 직접  score_result.csv파일을 열어보세요 현재 작업폴더에 있습니다
```

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

내레이션

◈ 엑셀파일 읽기

#엑셀파일도 손쉽게 처리할 수 있다 별도의 com 라이브러리나 다른 라이브러리를 설치할 필요가 없다

#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)
```

내레이션

◈ score_result1.xlsx

	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

index=False옵션을 주지 않을경우
인덱스 필드도 출력이된다.

index=False옵션을 주었을 경우
index필드는 출력이 되지 않는다

score_result2.xlsx

	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
7	일지매	90	90	0	180	60

내레이션

◈ DataFrame 파일 옵션

옵션	설명
path	파일의 위치(파일명 포함) url
sep(delimiter)	csv파일의 구분자가 ,(콤마)가 아니라 다른 문자일 경우에 이 옵션을 이용해 다른 문자를 지정할 수 있다
header	열 이름으로 사용할 행의 번호
index_col	행 인덱스로 사용할 열의 번호 또는 열의 이름
names	열 이름으로 사용할 문자열 리스트
skiprows	처음 몇줄을 스킵할지 지정한다 , 제목줄이 첫번째 줄에 없을때, header 도 사용 가능하다

간지 부분

◈ 데이터프레임 API

1. 데이터프레임이 제공하는 API의 활용
2. 데이터프레임이 제공하는 통계함수

내레이션

(기입하지 않습니다.)

◈ DataFrame API

- DataFrame 은 데이터를 분석하기 위한 많은 API를 제공합니다.
- 실습으로 사용할 데이터는 UCLA 대학의 머신러닝 데이터를 사용합니다.
- 이 데이터는 자동차의 연비, 실런더수, 배기량, 출력, 차중, 가속능력, 출시년도, 제조국, 모델명에 대한 정보를 가지고 있습니다
- 예제파일은 data 폴더 안에 있습니다.
- 데이터 프레임은 파이썬 클래스로 만들어집니다. 이 클래스에는 데이터프레임의 크기, 데이터구성항목, 자료형, 통계 수치등의 정보를 담고 있습니다.

내레이션

◈ DataFrame API

- head()
  - 데이터 프레임의 앞에서부터 5개의 데이터를 보여줍니다.
  - 인수를 직접 지정하여 n개의 데이터를 확인할 수 도 있습니다.
  - 데이터 분석용 파일들은 데이터 개수가 커서, 모든 내용을 한번에 확인하기엔 어렵습니다.
  - head 함수등을 이용해 데이터의 대략적 성격을 파악할 수 있습니다
  - data.head(3)

- tail()
  - 데이터 프레임의 뒤에서부터 5개의 데이터를 보여줍니다
  - data.tail(3)

- shape
  - 데이터프레임의 행과 열이 크기를 알려줍니다. tuple타입을 전달합니다.
  - tuple 타입을 전달하기 때문에 다음처럼 사용이 가능합니다.
  - row , col = data.shape

내레이션

◈ DataFrame API

```
#파일명 : exam10_5.py

import pandas as pd

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

print("앞에서 부터 5개만 미리 보기")
print( data.head() )

print("뒤에서 부터 5개만 미리 보기")
print( data.tail() )

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

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

```
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
```

내레이션

◈ DataFrame API

info ()
dataFrame 객체 안에 어떤 필드들을 갖고 있는지 구조를 파악할 수 있습니다. 데이터 갯수는 몇개이고, 인덱스는 무엇이고,
각 컬럼별로 어떤 데이터 타입인지 확인이 가능합니다. auto-mpg.csv 파일의 예입니다 .

```
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
```

내레이션

◈ DataFrame API

describe : 각 필드별로 카운트, 평균, 표준편차, 최소값, 사분위수(1/4, 2/4, ¾), 최대값을 확인할 수 있습니다
이 함수는 파이썬을 이용하여 데이터를 분석할때, 대략적인 데이터에 대한 파악을 도와주는 함수입니다.
각 필드별로 통계에서 자주 사용하는 통계량들을 보여줍니다.

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

내레이션

◈ DataFrame의 요약 정보 보기 예

```
#파일명 : exam10_6.py

import pandas as pd

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

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

print("데이터의 요약정보 확인")
print( data.describe())
```

내레이션

◈ 조건부여하기

- DataFrame 객체는 데이터를 검색할때 특정 조건을 부여하기가 쉽습니다. 별도의 for문을 필요로 하지 않습니다.

- 예) data[조건식] 조건식의 결과가 참인 경우의 데이터만 반환합니다.

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

  조건식은 == 뿐만아니라 관계연산자(==, !=, >, <. >=. <=) 모두 사용이 가능합니다 .

- 조건이 두개이상 결합할때는 파이썬이 제공하는 논리연산자인 and(&) 나 or(|) 연산자를 사용하면 안됩니다.

- 파이썬 논리연산자를 사용할 경우에는 다음과 같은 에러가 발생합니다.

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

내레이션

◈ 조건부여하기

- 조건식을 결합해야 할 경우에는 numpy 라이브러리를 사용합니다.

- numpy 라이브러리에서 logical_and 메서드와 logical_or 메서드를 대신 사용합니다.

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

- 주의할점: 결국 파이썬의 사용이 가장 많이 이루어지는 영역은 머신러닝이나 딥러닝입니다. 머신러닝이나 딥러닝 라이브러리는 모두 C언어로 되어 있습니다. C언어로 되어 있는데 파이썬은 사용하는 이유는 파이썬이 C언어와의 결합력이 우수하고, C언어를 습득하는것 보다는 파이썬을 습득하는게 더 빠르고, 코딩을 쉽게 할 수있기때문에 대부분의 빅데이터분야 개발자들은 파이썬을 사용합니다. 그러나 파이썬에서 사용하는 기본 데이터 타입들(list, dict, tuple)등은 바로 C언어 구조로 변환되지 않습니다. 그래서 pandas나 numpy 라이브러리를 만든겁니다. 빅데이터 분석에서는 numpy에서 제공하는 데이터타입이나 pandas 에서 제공하는 데이터 타입으로 파이썬 데이터 타입들을 변환해야만 합니다.

내레이션

◈ 조건부여하기

```
#파일명 : exam10_7.py

import pandas as pd

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

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

#연비가 27 이상만
print( data[data.mpg>=27] )

#모델 연도가 70년이고 연비가 27이상만, -(하이픈) 때문데 data.model-year 는 안된다.
#아래처럼 하던지 아니면 컬럼명을 수정해야 한다
print ( data[data['model-year']==70])

#두가지 조건을 동시에 주고 싶을때 - 에러발생 이렇게 못쓴다
#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().

#두가지 조건을 동시에 주고 싶을때
import numpy as np
print(data [np.logical_and(data['model-year']==70, data['mpg']>=25)])
```

내레이션

◈ DataFrame이 제공하는 통계 함수

- DataFrame 객체의 각 열은 Series티입입니다.

- 각 열의 평균이나 중간값, 최대값, 최소값등을 구하려면 numpy 라이브러리를 사용하지 않더라도, Series 타입도 기본적을 제공합니다.

- max : 해당 필드의 최대값을 반환한다.

- min : 해당 필드의 최소값 반환한다.

- std : 표준편차를 반환한다. 표준편차랑 집단의 성격을 알아내기 위해 사용하는 값이다. 표준편차가 크면 집단의 평균이 같더라도, 최대값과 최소값의 차가 커서 값이 넓게 분포되어 있음을 의미한다.
- var : 분산, 집단내 값이 흩어짐의 척도, 파이썬은 표준 분산을 사용하지 않고 있다

- quantile : 사분위수, 집단내의 값들의 ¼ 수 2/4 수 ¾ 수 등을 알아보기 위해 사용한다
  quantile(0.25), quantile(0,5), quantile(0.75)

- median : 집단내에서 중간번째 위치한 값을 알아내고자 할때 사용한다. 집단내의 값들의 편차가 클 경우 평균값만으로는 특정 집단의 특성을 나타내기 어려운 경우가 많다 이럴때 중간값을 사용한다

내레이션

◈ DataFrame이 제공하는 통계 함수 예제

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

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

print( data['model-year'].value_counts()) #value_counts 각 데이터별 고유카운트

#평균, 최대, 최소
print("연비평균 : ", data['mpg'].mean())
print("연비최대 : ", data['mpg'].max())
print("연비최소 : ", data['mpg'].min())
print("연비중간 : ", data['mpg'].median())
print("연비분산 : ", data['mpg'].var())
print("연비표준편차 : ", data['mpg'].std())

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

내레이션

◈ 적용하기

data 폴더 아래에 있는 iris.csv 파일을 읽어서 다음 동작들을 수행해 보세요

1) iris 데이터셋에 몇개의 필드가 있고 각 필드의 타입이 무엇인지 확인해 보세요

2) 맨 앞의 데이터를 7개만 출력해보세요

3) iris 데이터셋의 통계량을 확인해보세요(평균, 표준편차, 중간값, 사분위수...)

4) variety 이 Setosa 인 데이터의 통계량을 출력하세요

5) 각각 variety가 Setosa, Virginica Versicolor 의 sepal.length 값의 평균값을 출력하시오

6) 꽃의 종류가 Setosa 이면서 sepal.length 길이가 5cm이상인 것의 개수를 출력하시오

내레이션

(기입하지 않습니다.)

◈ 적용하기 -> 풀이

파일명 : homework10.py

```
#파일명 : exam10_8.py

import pandas as pd
import numpy as np
#header가 3번째 줄에 있음
data = pd.read_csv("./data/iris.csv")

#1) iris 데이터셋에 몇개의 필드가 있고 각 필드의 타입이 무엇인지 확인해 보세요
print(data.info())

#2) 앞에 7개 데이터
print(data.head(7))

#3)통계량 요약정보
print(data.describe())
```

내레이션

(기입하지 않습니다.)

◈ 적용하기 -> 풀이

#4)variety 이 Setosa 인 데이터의 통계량을 출력하세요

```
temp = data[data['variety']=='Setosa']
print(temp)
```

#5) 각각 variety가 Setosa, Virginica Versicolor 의 sepal.length 값의 평균값을 출력하시오

```
temp = data[data['variety']=='Setosa']['sepal.length']
print("Setosa 평균 : ", temp.mean())
temp = data[data['variety']=='Versicolor']['sepal.length']
print("Versicolor 평균 : ", temp.mean())
temp = data[data['variety']=='Virginica']['sepal.length']
print("Virginica 평균 : ", temp.mean())
```

#6) 꽃의 종류가 Setosa 이면서 sepal.length 길이가 5cm이상인 것의 개수를 출력하시오

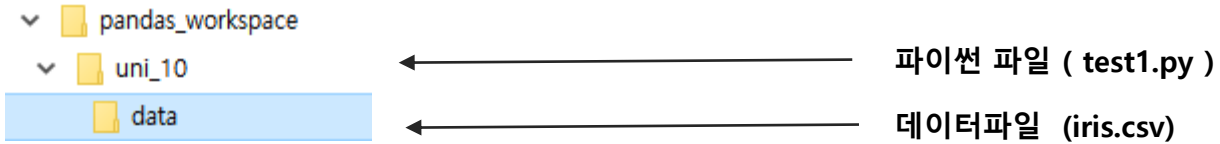
```
temp = data[np.logical_and(data['variety']=='Setosa', data['sepal.length']>=5)]
print(temp)
```

내레이션

(기입하지 않습니다.)

◈ 문제풀기

문제 1) csv 파일을 읽으려고 합니다. 폴더 구조가 다음과 같을 때 경로를 틀리게 지정한 것은?



① data = pd.read_csv(“/pandas_workspace/uni_10/data/iris.csv”)

② data = pd.read_csv(r“c:\pandas_workspace\uni_10\data\iris.csv”)

③ data = pd.read_csv(“c:\pandas_workspace\uni_10\data\iris.csv”)

④ data = pd.read_csv(“.\\data\\iris.csv”)

정답) 3

해설) 경로 지정시 \ 처리 문제입니다. \를 사용하고자 할때는 \\ 두개를 겹쳐 쓰거나 문자열앞에 r을 붙여줘야 합니다

난이도) 1(1:아주어려움, 2:어려움 3: 보통 4: 쉬움 5: 아주 쉬움)

관련페이지번호) 9

내레이션

(기입하지 않습니다.)

번호	문제	정답	난이도	해설	관련학습보기
2	**제목이 없는 csv파일을 읽었을때 별도의 컬럼 제목을 붙이고자 합니다. 파이르 제대로 읽은 명령문은? (단 파일명은 iris.csv이고 파이썬 파일과 동일한 디렉토리에 있습니다.)** ① data = pd.read_csv(“iris.csv”) ② data = pd.read_csv(“iris.csv”, header=None) ③ data = pd.read_csv(“./data/iris.csv”, header=None) ④ data = pd.read_csv(“iris.csv”, index_col=1)	2	4	제목줄이 없을때는 heade필드에 None 속성을 부여하면 됩니다. 파이썬 파일과 데이터파일이 동일한 경로일때는 파일명만 기술하면 됩니다.	13
3	**DataFrame 객체에서 특정 조건을 만족하는 필드만 추출하고자 합니다. auto-mpg.csv 파일에서 cylinders 필드값이 8인 데이터의 horsepower 필드값만 보고자 합니다. 바르게 작성하세요**	해설참조	2	data[data['cylinders']==8]['horsepower’] 또는 data[data.cylinders]==8]['horsepower']	27
4	**auto-mpg.csv 파일을 읽어 각 필드들에 대한 정보(데이터타입, 필드명등)를 확인하고자 할때 어떠한 함수를 사용하는가** ① info() ② describe() ③ tail() ④ head()	1	5	필드에 대한 정보는 info함수를 사용합니다 .	24

번호	문제	정답	난이도	해설	관련학습보기
5	**auto-mpg.csv 파일을 읽어 각 필드들에 대한 통계량(평균, 표준편차, 분산, 중앙값등) 을 확인하기 위한 함수는 ?** ① info() ② describe() ③ tail() ④ head()	2	5	통계량을 확인하려면 describe함수를 사용합니다	24
6	**데이터프레임의 컬럼명을 변경하고자 할때 사용하는 속성은 ?** ① info ② shape ③ cols ④ columns	4	5	data.columns= ['필드명1', '필드명2', '필드명3'] 등으로 부여하면 됩니다.	29
7	**데이터프레임 객체의 데이터의 행과 열의 개수를 확인하는 속성은?**	shape	4	shape속성을 활용하면 행과 열등 크기를 확인 할 수 있습니다.	

번호	문제	정답	난이도	해설	관련학습보기
8	**auto-mpg.csv 파일을 읽어서 모델의 연도고 70년과 71년 두 종류의 데이터만 보고싶을때 올바른 수식은?** ① data[np.logical_and(data['model-year']==70, data['model-year']==71)] ② data[np.logical_or(data['model-year']==70, data['model-year']==71)] ③ data[data['model-year']==70 and data['model-year']==71] ④ data[data['model-year']==70 or data['model-year']==71]	2	1	논리식의 경우에는 and나 or 연산이 아니라 numpy의 logical_or 나 logical_and 연산을 사용해야 한다	28
9	**auto-mpg.csv 파일에서 cylinders 가 8개인 데이터의 개수를 알아내고자 한다. 올바른 방법은?** ① data[data['cylinders']==8].count() ② data[data['cylinders']==8].shape ③ data[data['cylinders']==8].length() ④ data[data['cylinders']=8].length()	2	4	data[data['cylinders']==8].shape data[조건식].shape 속성을 사용하면 열의 개수와 행의 개수 모드를 알 수 있습니다.	29
10	**auto-mpg.csv 파일에서 cylinders 가 8개인 horsepower의 평균값을 알아내는 수식을 올바르게 작성하시오**	해설참조	5	data[data['cylinders']==8]['horsepower'].mean()	29

과정명		모듈명		회차명	

◈ 핵심요약

▶ DataFrame

- 판다스에서 제공하는 데이터 구조중에 DataFrame이 가장 많이 쓰입니다.
- 판다스는 csv나 excel, json 등의 파일을 읽어서 바로 DataFrame 객체를 전달해줍니다.
- DataFrame 객체는 for문을 사용하지 않고도 조건식을 마치 인덱스처럼 데이터에 접근할 수 있습니다.
- DataFrame 객체의 인덱스에 논리식을 사용할 경우에는 파이썬의 논리식이 아니라 numpy 에서 제공하는 논리함수를 사용해야 합니다

▶ DataFrame API

head() - dataframe 객체의 데이터를 기본적으로 앞에서부터 5개를 보여준다

tail() - dataframe객체의 데이터를 기본적으로 뒤애서부터 5개를 보여준다

info() - dataframe 내의 각 필드의 종류와 데이터 타입등, 데이터 개수등의 정보를 보여준다

describe() - 기본 통계값인 max, min, mean, quantile(퍼센트), std 등을 적용한 통계량을 간략하게 보여준다

내레이션

(기입하지 않습니다.)