6. 판다스 (Pandas) 기초1

리스트 vs. 사전 vs. 넘파이

	리스트	사전	넘파이
장점	• 각 요소가 서로 다른 자료형 가능 • 리스트 추가, 삭제, 합치기 용이	• key-value 구조 • key (레이블) 를 사용하는 편의성	• 배열 연산 성능 우수 • 배열 생성, reshape, 슬라이싱, 브로드캐스팅
단점	• 인덱스로만 데이터 접근 • 열 단위로 접근 불가 • 브로드캐스팅 기능 부재	• 순차적 접근이 불가 • 사전 자료형 끼리 연산 불가 • 브로드캐스팅 기능 부재	• 인덱스로만 데이터 접근 • 모든 요소 동일 자료형만 지원

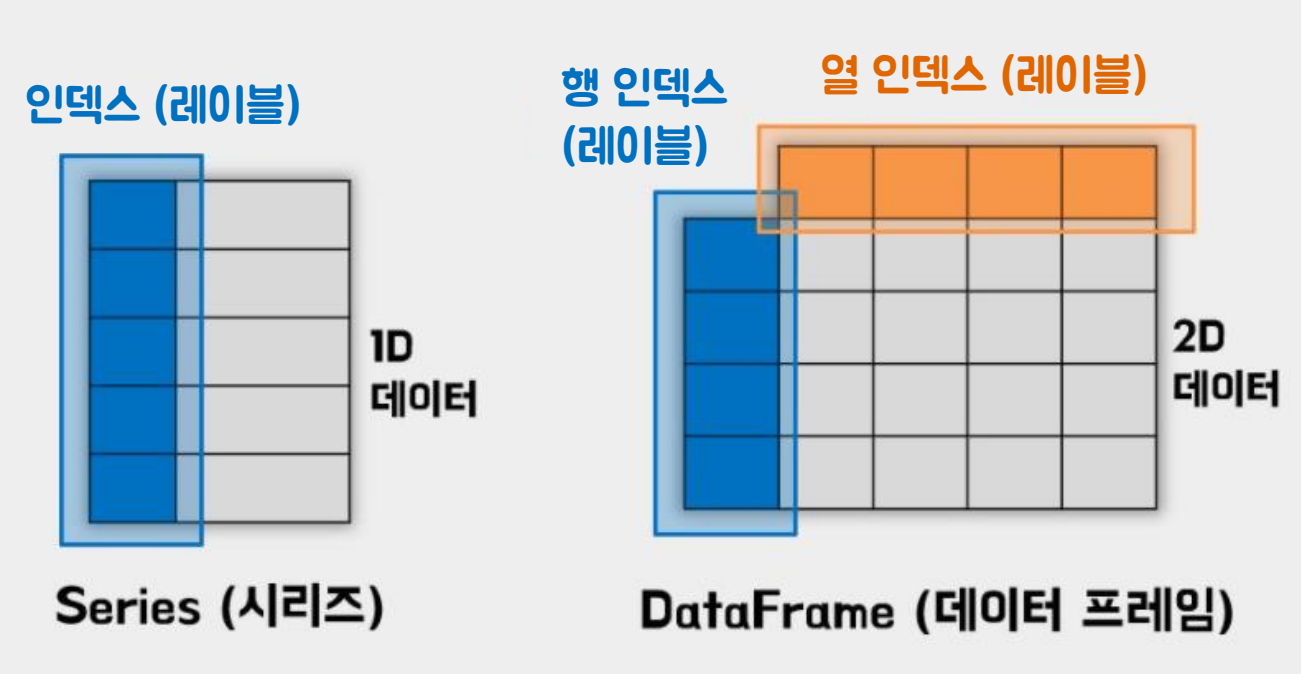
판다스 (Pandas)

- 넘파이를 기반으로 처리속도가 빠르며, 행과 열을 잘 관리할 수 있는 데이터 프레임을 제공
- 이종 자료형의 열을 가진 테이블 데이터에 적합
- 데이터 처리와 분석에 필요한 기능들을 제공
  - 행, 열 데이터 조작 및 정렬 용이
  - 결측 데이터 처리 및 필터링 용이
  - matplotlib 와 결합하여 데이터 시각화 용이

설치: pip install pandas

판다스 기본 자료구조

- 시리즈 (Series) : 1차원 동일 자료형 배열
- 데이터프레임 (DataFrame) : 복수의 (다른 자료형) 열로 구성된 2차원 배열



시리즈 생성하기

- pd.Series( ) 에 리스트를 입력해서 시리즈 를 생성 가능
- 인덱스를 별도 지정하지 않으면 0 부터 시작하는 정수 값이 자동 할당

```
import pandas as pd

ser = pd.Series([10, 20, 30, 40])
print( type(ser) )
print( ser )
print()
print( ser[0], ser[2] )   # 인덱스로 접근
print( ser.index )        # 행방향 인덱스
```

```
<class 'pandas.core.series.Series'>
0    10
1    20
2    30
3    40
dtype: int64

10 30
RangeIndex(start=0, stop=4, step=1)
```

시리즈 생성하기

- 시리즈 인덱스를 레이블로 변경할 수 있음

```
import pandas as pd


grades = ['A', 'B+', 'A+', 'A']
ser = pd.Series(grades, index=['춘향', '몽룡', '향단', '방자' ] )
print( type(ser) )
print( ser )
print()
print( ser[0], ser[2] )                          # 정수 인덱스로 접근
print( ser['춘향'], ser['향단'] )     # 레이블로 (문자열 인덱스) 접근
print( ser.index )                               # 인덱스 속성
```

```
<class 'pandas.core.series.Series'>
춘향      A
몽룡      B+
향단      A+
방자      A
dtype: object


A A+
A A+
Index(['춘향', '몽룡', '향단', '방자'], dtype='object')
```

시리즈 생성하기

- 사전을 이용하여 시리즈 를 생성할 수 있음

```
import pandas as pd


grades = { '춘향 ':'A', '몽룡':'B+', '향단':'A+', '방자':'A' }
ser = pd.Series(grades )
print( type(ser) )
print( ser )
print()
print( ser[0], ser[2] )                    # 정수 인덱스로 접근
print( ser['춘향'], ser['향단'] )          # 레이블로 (문자열 인덱스) 접근
```

```
<class 'pandas.core.series.Series'>
춘향      A
몽룡      B+
향단      A+
방자      A
dtype: object


A A+
A A+
```

데이터프레임 생성 예시

- 넘파이를 사용한 2차원 데이터프레임 생성
  - 열과 행 모두 정수 기반 인덱스가 자동 할당

```
import pandas as pd
import numpy as np

df = pd.DataFrame( np.zeros( (4, 3) ) )
print( type(df) )
print( df )
```

```
<class 'pandas.core.frame.DataFrame'>
     0    1    2
0  0.0  0.0  0.0
1  0.0  0.0  0.0
2  0.0  0.0  0.0
3  0.0  0.0  0.0
```

데이터프레임 생성 예시

넘파이를 사용한 2차원 데이터프레임 생성

열 인덱스 레이블링 지정

```
import pandas as pd
import numpy as np
classes = ['국어', '영어', '수학']
df = pd.DataFrame( np.zeros( (4, 3) ),  columns = classes)
print( type(df) )
print( df )
```

```
<class 'pandas.core.frame.DataFrame'>
   국어   영어   수학
0  0.0  0.0  0.0
1  0.0  0.0  0.0
2  0.0  0.0  0.0
3  0.0  0.0  0.0
```

데이터프레임

2차원 데이터프레임 생성

사전 & 리스트 사용 예시

```
import pandas as pd

names = ['춘향', '몽룡', '향단', '방자' ]
korean = [100, 95, 90, 85]
english = [85, 90, 95, 100]
math = [90, 85, 100, 95]

df = pd.DataFrame( {'이름':names, '국어':korean, '영어':english, '수학':math } )
print( type(df) )
print( df )
```

```
<class 'pandas.core.frame.DataFrame'>
   이름   국어   영어   수학
0  춘향   100   85    90
1  몽룡   95    90    85
2  향단   90    95    100
3  방자   85    100   95
```

데이터프레임 사용 예시

열 인덱스로 시리즈 를 접근

```
# 위 예시에 이어서
print(df.columns)          # 열 방향 인덱스 정보
print(df.index)            # 행 방향 인덱스 정보
print()
name_ser = df['이름']      #열 인덱스로 접근
print( type(name_ser) )
print( name_ser )
```

```
Index(['이름', '국어', '영어', '수학'], dtype='object')
RangeIndex(start=0, stop=4, step=1)

<class 'pandas.core.series.Series'>
0    춘향
1    몽룡
2    향단
3    방자
Name: 이름, dtype: object
```

데이터프레임 사용 예시

- 열 인덱스를 리스트 형식으로 접근하면 데이터프레임 형식으로 반환

```
# 위 예시에 이어서
classes_df = df[ ['이름', '수학'] ]  #열 인덱스 리스트로 접근
print( type(classes_df) )
print( classes_df )
```

```
<class 'pandas.core.frame.DataFrame'>
   이름   수학
0  춘향   90
1  몽룡   85
2  향단  100
3  방자   95
```

데이터프레임 사용 예시

특정 열 데이터 갱신

```
# 위 예시에 이어서
print( df['영어'] )
df['영어'] = df['영어'] + 10  # 모든 학생에게 보너스 점수
print()
print( df['영어'] )
```

```
0     85
1     90
2     95
3    100
Name: 영어, dtype: int64

0     95
1    100
2    105
3    110
Name: 영어, dtype: int64
```

데이터프레임 사용 예시

행 인덱싱 : 슬라이싱을 사용해야 함!

```
# 위 예시에 이어서
print( df[ 1:2 ] )
print( df[ 2: ] )
```

```
  이름   국어   영어   수학
1  몽룡   95   90   85

   이름   국어    영어    수학
2  향단   90    95   100
3  방자   85   100    95
```

개별 데이터 인덱싱

```
print( df ['국어'][1] )   # 열 선택 후 행 인덱싱
print()
print( df [1:2]['국어'] ) # 행 선택 후 열 인덱싱
```

```
95

1    95
Name: 국어, dtype: int64
```

데이터프레임 행 인덱스 레이블링

데이터프레임 행 인덱스 레이블링

```
import pandas as pd

names = ['춘향', '몽룡', '향단', '방자' ]
korean = [100, 95, 90, 85]
english = [85, 90, 95, 100]
math = [90, 85, 100, 95]

df = pd.DataFrame( {'국어':korean, '영어':english, '수학':math }, index=names)
print( type(df) )
print( df )
```

```
<class 'pandas.core.frame.DataFrame'>
      국어   영어   수학
춘향   100    85    90
몽룡    95    90    85
향단    90    95   100
방자    85   100    95
```

불리언 배열을 사용한 인덱싱

```
# 위 예시에 이어서
bool_index = df[ '수학' ] > 90
print( bool_index )
print()
print( df[ bool_index ] )
```

```
춘향     False
몽룡     False
향단      True
방자      True
Name: 수학, dtype: bool

     국어   영어   수학
향단   90    95    100
방자   85   100     95
```

```
# 위 예시에 이어서
bool_index = df > 90
print(bool_index)
print()
print( df[ bool_index ] )
```

```
       국어    영어    수학
춘향    True   False   False
몽룡    True   False   False
향단   False    True    True
방자   False    True    True

       국어    영어    수학
춘향   100.0    NaN     NaN
몽룡    95.0    NaN     NaN
향단     NaN   95.0   100.0
방자     NaN  100.0    95.0
```

데이터프레임 인덱싱 요약

행 인덱싱

	가능 여부	출력 자료형
레이블	X	
레이블 리스트	X	
(정수) 인덱스	X	
(정수) 슬라이스	O	데이터프레임
불리언 배열	O	데이터프레임

열 인덱싱

	가능 여부	출력 자료형
레이블	O	시리즈
레이블 리스트	O	데이터프레임
(정수) 인덱스	X	
(정수) 슬라이스	X	
불리언 배열	X	

데이터프레임 고급 인덱싱

- (행 인덱스, 열 인덱스) 형식의 2차원 인덱싱을 지원하는 속성
  - loc : 레이블 기반의 2차원 인덱싱
    - df.loc[행 인덱싱 값]
    - df.loc[행 인덱싱 값, 열 인덱싱 값]
    - df.loc[행 슬라이싱]
    - df.loc[불리언 배열]
  - iloc : 순서를 나타내는 정수 기반의 2차원 인덱싱
    - 레이블이 아닌 정수 인덱스만 사용. 다른 사항은 loc 인덱서와 동일

loc 속성 사용 예시

```
import pandas as pd
names = ['춘향', '몽룡', '향단', '방자' ]
korean = [100, 95, 90, 85]
english = [85, 90, 95, 100]
math = [90, 85, 100, 95]
df = pd.DataFrame( {'국어':korean, '영어':english, '수학':math }, index=names)
print( df )
print()
print( df.loc[ '춘향' ] ) # 행 접근
print()
print( df.loc[ ['춘향', '향단'] ] ) # 행 리스트 접근
print()
print( df.loc[ '춘향', '국어' ] ) # 개별 값 접근
```

```
     국어   영어   수학
춘향  100   85   90
몽룡   95   90   85
향단   90   95  100
방자   85  100   95

국어   100
영어    85
수학    90
Name: 춘향, dtype: int64

     국어   영어   수학
춘향  100   85   90
향단   90   95  100

100
```

loc 속성 사용 예시

```
# 위 예시 이어

print( df.loc[ '몽룡' : '방자' ] )
# 주의: 슬라이스의 마지막 값이 포함됨
print()
# 불리언 시리즈로 접근
print( df.loc[ df['국어'] > 90] )
print()
# 행 & 열 모두 리스트로 접근
print( df.loc[  ['춘향', '몽룡'],  ['수학', '영어' ] ] )
print()
# 수학 90점 넘는 학생의 국어 & 영어 점수는?
print( df.loc[  df['수학']>90,   ['국어', '영어' ] ] )
```

```
      국어   영어   수학
몽룡    95    90    85
향단    90    95   100
방자    85   100    95

      국어   영어   수학
춘향   100    85    90
몽룡    95    90    85

      수학   영어
춘향    90    85
몽룡    85    90

      국어   영어
향단    90    95
방자    85   100
```

iloc 속성 사용 예시

```
import pandas as pd
names = ['춘향', '몽룡', '향단', '방자' ]
korean = [100, 95, 90, 85]
english = [85, 90, 95, 100]
math = [90, 85, 100, 95]
df = pd.DataFrame( {'국어':korean, '영어':english, '수학':math }, index=names)
print( df )
print()
# print( df.loc['춘향'] )
print( df.iloc[0] ) # 한 행 접근
print( df.iloc[ [0, 2] ] ) # 한 행 접근
print( df.iloc[0, 0] ) # 단일 값 접근
```

```
    국어  영어  수학
춘향  100   85   90
몽룡   95   90   85
향단   90   95  100
방자   85  100   95

국어   100
영어    85
수학    90
Name: 춘향, dtype: int64

    국어  영어  수학
춘향  100   85   90
향단   90   95  100

100
```

iloc 속성 사용 예시

```
# 위 예시 이어


print()
print( df.iloc[ 0:2 ] ) # 행 슬라이싱
print()
print( df.iloc[ 0:2, 1:3 ] ) # 행&열 슬라이싱
print()
print( df.iloc[  [0, 3],  [1, 2 ] ] ) # 행&열 리스트 접근
print()
print( df.iloc[0] > 90 ) # 불리언 시리즈
print()
print( df.iloc[ 0 ][ df.iloc[0] > 90 ] )
# 행렬 인덱스 0  학생 점수 중 90 점 넘는 것은?
```

```
     국어   영어   수학
춘향   100   85   90
몽룡    95   90   85

     영어   수학
춘향   85   90
몽룡   90   85

     영어   수학
춘향    85   90
방자   100   95

국어     True
영어     False
수학     False
Name: 춘향, dtype: bool

국어     100
Name: 춘향, dtype: int64
```

판다스 CSV 파일 사용 예시 : read_csv

- 많은 데이터셋이 CSV 파일로 제공되며, 판다스 read_csv 함수로 읽고 데이터프레임 생성

```
import pandas as pd


df=pd.read_csv( "iris.csv" )
print( df )
print(df.head())
print(df.head(10))
print(df.tail())
print(df.tail(10))
```

```
     caseno  SepalLength  SepalWidth  PetalLength  PetalWidth     Species
0         1          5.1         3.5          1.4         0.2      setosa
1         2          4.9         3.0          1.4         0.2      setosa
2         3          4.7         3.2          1.3         0.2      setosa
3         4          4.6         3.1          1.5         0.2      setosa
4         5          5.0         3.6          1.4         0.2      setosa
..      ...          ...         ...          ...         ...         ...
145     146          6.7         3.0          5.2         2.3   virginica
146     147          6.3         2.5          5.0         1.9   virginica
147     148          6.5         3.0          5.2         2.0   virginica
148     149          6.2         3.4          5.4         2.3   virginica
149     150          5.9         3.0          5.1         1.8   virginica

[150 rows x 6 columns]
```

판다스 CSV 파일 사용 예시 : head( )

- 판다스 head() 함수로 데이터프레임 앞 일부분만 확인

```
import pandas as pd


df=pd.read_csv("iris.csv")
print(df)
print( df.head() )
print( df.head(10) )
print(df.tail())
print(df.tail(10))
```

```
   caseno  SepalLength  SepalWidth  PetalLength  PetalWidth Species
0       1          5.1         3.5          1.4         0.2  setosa
1       2          4.9         3.0          1.4         0.2  setosa
2       3          4.7         3.2          1.3         0.2  setosa
3       4          4.6         3.1          1.5         0.2  setosa
4       5          5.0         3.6          1.4         0.2  setosa

   caseno  SepalLength  SepalWidth  PetalLength  PetalWidth Species
0       1          5.1         3.5          1.4         0.2  setosa
1       2          4.9         3.0          1.4         0.2  setosa
2       3          4.7         3.2          1.3         0.2  setosa
3       4          4.6         3.1          1.5         0.2  setosa
4       5          5.0         3.6          1.4         0.2  setosa
5       6          5.4         3.9          1.7         0.4  setosa
6       7          4.6         3.4          1.4         0.3  setosa
7       8          5.0         3.4          1.5         0.2  setosa
8       9          4.4         2.9          1.4         0.2  setosa
9      10          4.9         3.1          1.5         0.1  setosa
```

판다스 CSV 파일 사용 예시 : head( )

- 판다스 tail() 함수로 데이터프레임 뒤 일부분만 확인

```
import pandas as pd


df=pd.read_csv("iris.csv")
print(df)
print(df.head())
print(df.head(10))
print( df.tail() )
print( df.tail(10) )
```

```
     caseno  SepalLength  SepalWidth  PetalLength  PetalWidth    Species
145     146          6.7         3.0          5.2         2.3  virginica
146     147          6.3         2.5          5.0         1.9  virginica
147     148          6.5         3.0          5.2         2.0  virginica
148     149          6.2         3.4          5.4         2.3  virginica
149     150          5.9         3.0          5.1         1.8  virginica

     caseno  SepalLength  SepalWidth  PetalLength  PetalWidth    Species
140     141          6.7         3.1          5.6         2.4  virginica
141     142          6.9         3.1          5.1         2.3  virginica
142     143          5.8         2.7          5.1         1.9  virginica
143     144          6.8         3.2          5.9         2.3  virginica
144     145          6.7         3.3          5.7         2.5  virginica
145     146          6.7         3.0          5.2         2.3  virginica
146     147          6.3         2.5          5.0         1.9  virginica
147     148          6.5         3.0          5.2         2.0  virginica
148     149          6.2         3.4          5.4         2.3  virginica
149     150          5.9         3.0          5.1         1.8  virginica
```

판다스 CSV 파일 사용 예시 : index, columns, describe()

```
# 이전 코드에 이어서
print()
print( df.index )
print()
print( df.columns )
print()
print( df.describe() )
```

```
RangeIndex(start=0, stop=150, step=1)

Index(['caseno', 'SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Species'], dtype='object')

            caseno  SepalLength  SepalWidth  PetalLength  PetalWidth
count  150.000000   150.000000  150.000000   150.000000  150.000000
mean    75.500000     5.843333    3.057333     3.758000    1.199333
std     43.445368     0.828066    0.435866     1.765298    0.762238
min      1.000000     4.300000    2.000000     1.000000    0.100000
25%     38.250000     5.100000    2.800000     1.600000    0.300000
50%     75.500000     5.800000    3.000000     4.350000    1.300000
75%    112.750000     6.400000    3.300000     5.100000    1.800000
max    150.000000     7.900000    4.400000     6.900000    2.500000
```

describe() 출력은 데이터프레임!

```
# 이전 코드에 이어서
res = df.describe()
print( type(res) )
print()
print( res.loc['mean'] )
print()
print( res.loc['mean'] ['PetalWidth'])
```

	caseno	SepalLength	SepalWidth	PetalLength	PetalWidth
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.057333	3.758000	1.199333
std	43.445368	0.828066	0.435866	1.765298	0.762238
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
<class 'pandas.core.frame.DataFrame'>

caseno          75.500000
SepalLength      5.843333
SepalWidth       3.057333
PetalLength      3.758000
PetalWidth       1.199333
Name: mean, dtype: float64

1.1993333333333336
```