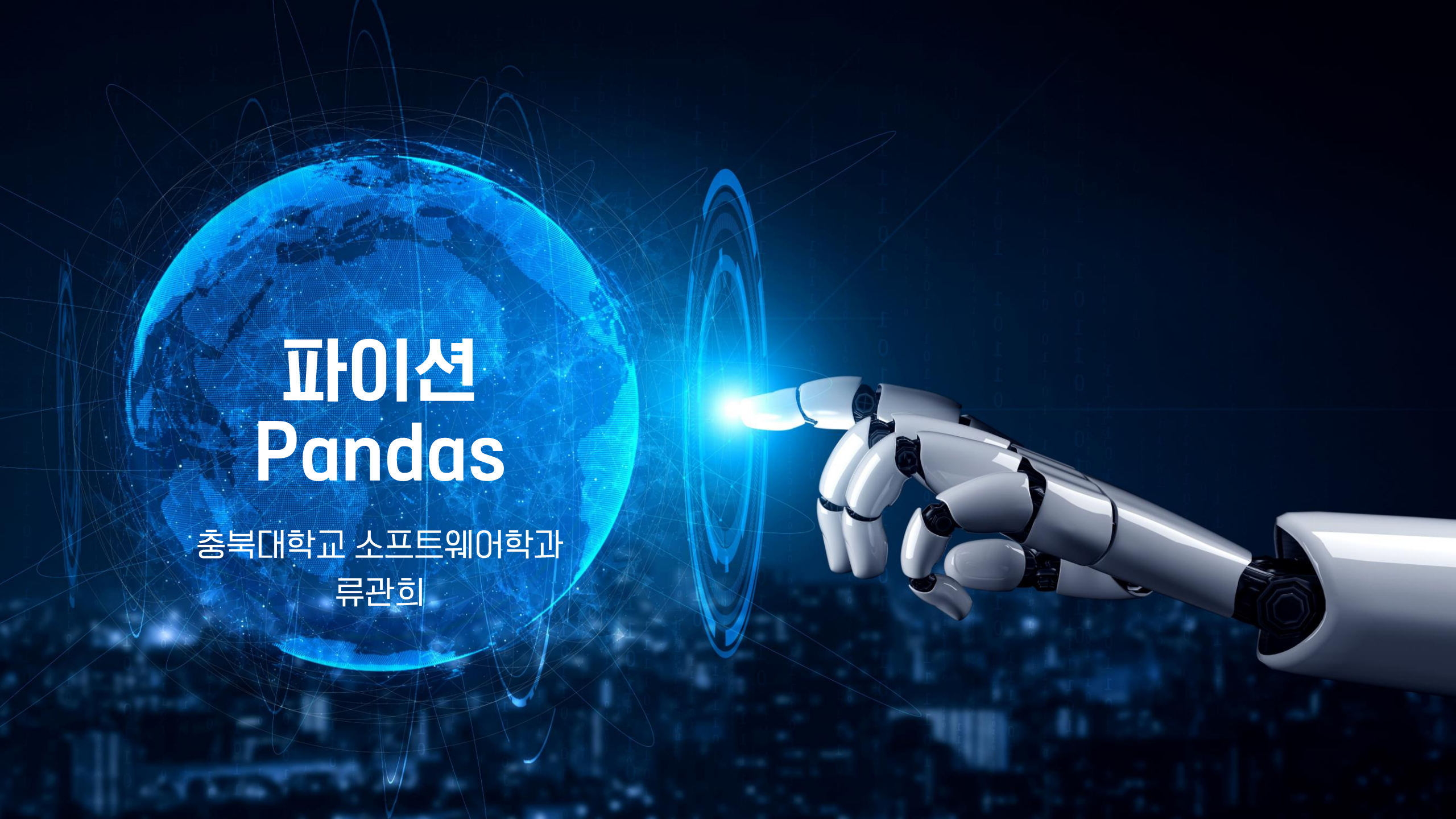
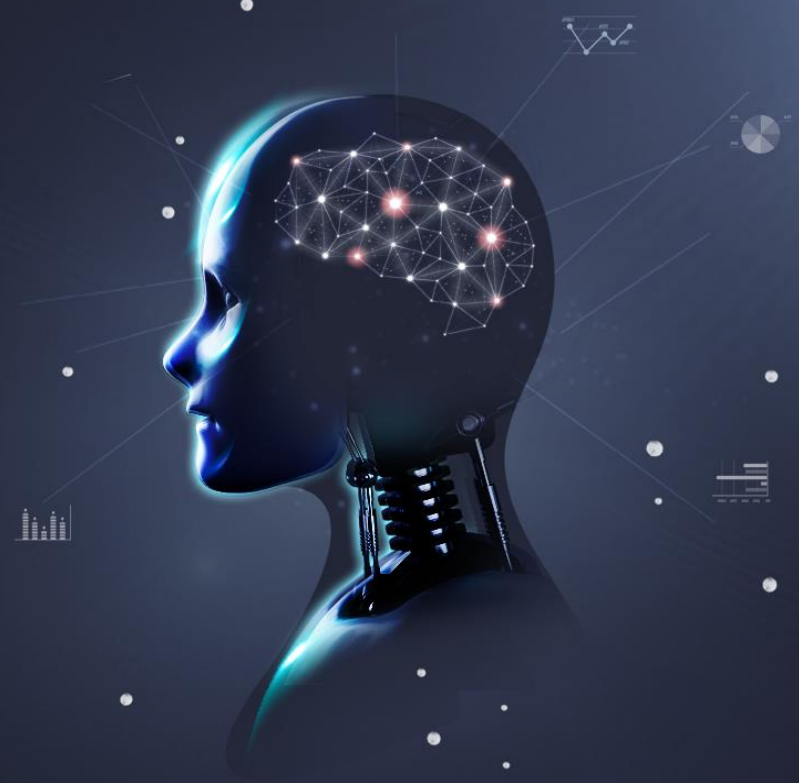


파이션 Pandas

충북대학교 소프트웨어학과
류관희

목 차



01

Pandas 소개

- 개요 및 설치
- 1차원 저장 구조

02

Pandas 2차원 데이터 처리

- 2차원 데이터 구조
- 2차원 데이터 조작

03

Pandas 통계 및 함수

- 통계기능
- Pandas 함수

학습목표

- Python 프로그램에서 데이터 저장을 위해 널리 사용되는 Pandas에 대해 공부한다.
  - 설치 방법
  - 1차원 데이터 저장 구조

Pandas 개요

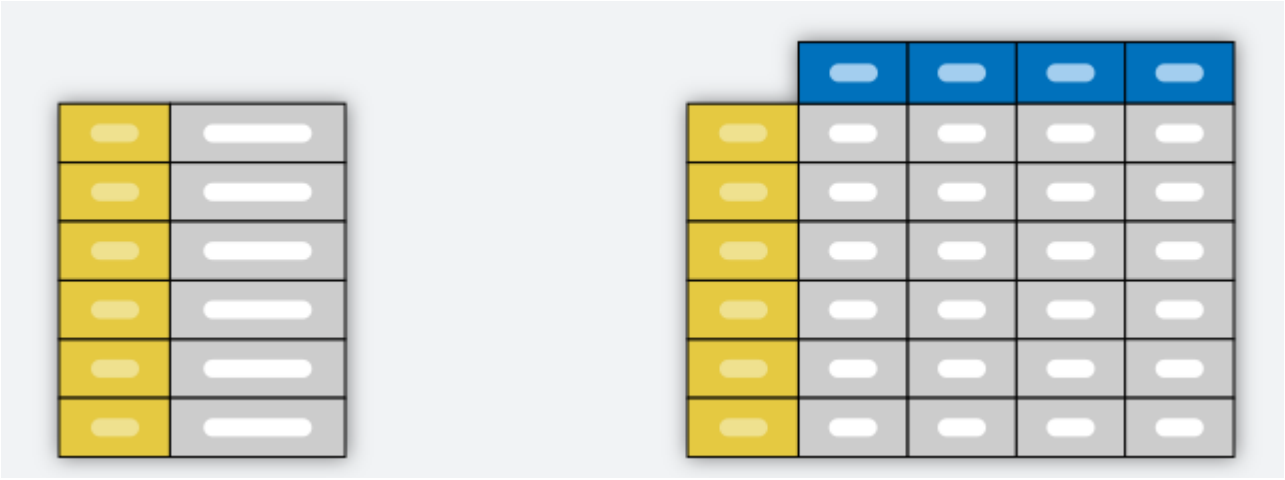
- Python 프로그래밍 언어를 기반으로 구축 된 빠르고 강력하고 유연하며 사용하기 쉬운 오픈 소스 데이터 분석 및 조작 도구
- 기능 http://pandas.pydata.org
  - ✓ Input/output
  - ✓ General functions
  - ✓ Series
  - ✓ DataFrame
  - ✓ pandas arrays
  - ✓ Panel
  - ✓ Index objects
  - ✓ Date offsets
  - ✓ Window
  - ✓ GroupBy
  - ✓ Resampling
  - ✓ Style
  - ✓ Plotting
  - ✓ General utility functions
  - ✓ Extensions

Pandas 설치

- pip install pandas
- import pandas as pd

Pandas 저장 구조

- Series
  - 1차원 데이터 저장 구조
- DataFrames
  - 2차원 데이터 저장 구조



1차원 저장 구조(Series)

- Constructor
- Attributes
- Conversion
- Indexing, iteration
- Binary operator functions
- Function application, GroupBy & window
- Computations / descriptive stats
- Reindexing / selection / label manipulation
- Missing data handling
- Reshaping, sorting
- Combining / comparing / joining / merging
- Time Series-related
- Accessors
- Plotting
- Serialization / IO / conversion

1차원 데이터 생성

Scalar 데이터 사용

```
>>> import pandas as pd
>>> s = pd.Series(5, index=[0, 1, 2, 3])
>>> print(s)
0    5
1    5
2    5
3    5
dtype: int64
```

```
>>> s = pd.Series(5, index=['a', 'b', 'c', 'd'])
>>> s
a    5
b    5
c    5
d    5
dtype: int64

>>> s.astype('float64')
a    5.0
b    5.0
c    5.0
d    5.0
dtype: float64
```

1차원 데이터 생성

np.array 데이터 사용

```
>>> import pandas as pd
>>> import numpy as np
>>> data = np.array(['a','b','c','d'])
>>> s = pd.Series(data)
>>> print(s)
0  a
1  b
2  c
3  d
dtype: object
```

```
>>> s =
pd.Series(data,index=[100,101,102,103])
>>> print(s)
100  a
101  b
102  c
103  d
dtype: object
```

1차원 데이터 생성

dict 데이터 사용

```
>>> data = {'a' : 0., 'b' : 1., 'c' : 2.}
>>> s = pd.Series(data)
>>> print(s)
a    0.0
b    1.0
c    2.0
dtype: float64
>>> s = pd.Series(data,index=['b','c','d','a'])
>>> print(s)
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

1차원 데이터 생성

random 데이터 사용

```
>>> s=pd.Series(np.random.randn(4))
>>> s
0   -0.997358
1    1.123125
2   -0.729292
3   -0.232918
dtype: float64
>>> s.axes
[RangeIndex(start=0, stop=4, step=1)]
>>> s.ndim
1
>>> s.size
4
```

```
>>> s.values
array([-0.99735799,  1.12312476, -
0.7292916 , -0.23291822])
>>> type(s.values)
<class 'numpy.ndarray'>

>>> s.head(2)
0   -0.997358
1    1.123125
dtype: float64

>>> s.tail(2)
2   -0.729292
3   -0.232918
dtype: float64
```

1차원 데이터 접근


```
>>> s = pd.Series([1,2,3,4,5],index = list('abcde'))
>>> s[0]
1
>>> s[:3]
a    1
b    2
c    3
dtype: int64
>>> s['a']
1
>>> s[['a', 'c', 'd']]
a    1
c    3
d    4
dtype: int64
```

1차원 데이터 연산

```
>>> s1 = pd.Series([1,2,3],index = ['a','b','c'])
>>> s2 = pd.Series([4,5,6],index = ['c','a','b'])
>>> s3 = s1 + s2
>>> s1
a    1
b    2
c    3
dtype: int64
>>> s2
c    4
a    5
b    6
dtype: int64
>>> s3
a    6
b    8
c    7
dtype: int64
```

문제풀이

- Pandas에서 제공하는 1차원 저장구조와 2차원 저장구조를 무엇이라고 하나요?

- Pandas에서 1차원 저장구조를 생성하는 방법을 말해보세요.

요약

- Pandas에서 제공하는 1차원 및 2차원 저장구조인 Series, DataFrame을 공부하였음
- 1차원 저장 구조인 Series를 생성하기 위한 방법, 데이터접근 방법, 연산을 소개하였음

01

Pandas 소개

- 개요 및 설치
- 1차원 저장 구조

02

Pandas 2차원 데이터 처리

- 2차원 데이터 구조
- 2차원 데이터 조작

03

Pandas 통계 및 함수

- 통계 기능
- Pandas 함수

학습목표

- Python Pandas 에서 제공하는 2차원 데이터를 처리하는 방법을 공부한다.
  - 2차원 데이터 구조
  - 2차원 데이터 조작 (행 추가, 병합 등)

2차원 저장구조(DataFrame)

- Constructor
- Attributes and underlying data
- Conversion
- Indexing, iteration
- Binary operator functions
- Function application, GroupBy & window
- Computations / descriptive stats
- Reindexing / selection / label manipulation
- Missing data handling
- Reshaping, sorting, transposing
- Combining / comparing / joining / merging
- Time Series-related
- Metadata
- Plotting
- Sparse accessor
- Serialization / IO / conversion

DataFrame 생성

list 데이터 사용

```
1 import pandas as pd
2 data = ['A','B','C']
3 df = pd.DataFrame(data)
4 print(df)
```

```
   0
0  A
1  B
2  C
```

```
1 df.T
```

	0	1	2
0	A	B	C

```
1 data = [['A','B','C'], ['D','E','F']]
2 df = pd.DataFrame(data)
3 print(df)
```

```
   0  1  2
0  A  B  C
1  D  E  F
```

```
1 df.T
```

	0	1
0	A	D
1	B	E
2	C	F

DataFrame 생성


```
1  data = [['Alex',10],['Bob',12],['Clarke',13]]
2  df = pd.DataFrame(data,columns=['Name','Age'])
3  print(df)
```

```
     Name  Age
0    Alex   10
1     Bob   12
2  Clarke   13
```

```
1  df = pd.DataFrame(data,columns=['Name','Age'], dtype=float)
2  print(df)
```

```
     Name   Age
0    Alex  10.0
1     Bob  12.0
2  Clarke  13.0
```

DataFrame 생성

dict데이터 사용

```
data = {'Name':['A', 'B', 'C', 'D'],'Age':[28,34,29,42]}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
print(df)
```

```
      Name  Age
rank1    A   28
rank2    B   34
rank3    C   29
rank4    D   42
```

```
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)
print(df)
```

```
   a   b     c
0  1   2   NaN
1  5  10  20.0
```

```
df = pd.DataFrame(data, index=['first', 'second'])
print(df)
```

```
        a   b     c
first   1   2   NaN
second  5  10  20.0
```

DataFrame 생성

```
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df1=pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b'])
print(df1)
```

```
        a   b
first   1   2
second  5  10
```

```
df2=pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b1'])
print(df2)
```

```
        a  b1
first   1 NaN
second  5 NaN
```

```
df3=pd.DataFrame(data, index=['first', 'second'], columns=['a', 'c'])
print(df3)
```

```
        a     c
first   1   NaN
second  5  20.0
```

DataFrame 생성

Series 데이터 사용

```
>>> d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
         'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
>>> df = pd.DataFrame(d)
>>> print(df)
   one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
>>> print(df['one']) # column selection
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64
```

DataFrame 생성

Numpy 데이터 사용

```
1  dates = pd.date_range('20200901', periods=6)
```

```
1  df = pd.DataFrame(np.random.randn(6,2), index=dates, columns=list('AB'))
```

```
1  df
```

	A	B
2020-09-01	-1.086466	0.412586
2020-09-02	-0.337997	1.341769
2020-09-03	-0.788851	0.207951
2020-09-04	0.624435	0.473969
2020-09-05	0.376454	0.000762
2020-09-06	1.137598	0.361033

```
1  df.index
```

```
DatetimeIndex(['2020-09-01', '2020-09-02', '2020-09-03', '2020-09-04',
               '2020-09-05', '2020-09-06'],
              dtype='datetime64[ns]', freq='D')
```

```
1  df.columns
```

```
Index(['A', 'B'], dtype='object')
```

```
1  df.values
```

```
array([[-1.08646569e+00,  4.12585999e-01],
       [-3.37996785e-01,  1.34176908e+00],
       [-7.88850849e-01,  2.07950971e-01],
       [ 6.24435380e-01,  4.73968511e-01],
       [ 3.76454003e-01,  7.61785615e-04],
       [ 1.13759792e+00,  3.61032574e-01]])
```

DataFrame 데이터 생성

```
df = pd.DataFrame({ 'B': pd.Timestamp('20200927'),
                    'A': '삼성',
                    'C': pd.Series([1000.5, 900.0, 950.0, 1100.0], index=list(range(4)), dtype='float32'),
                    'D': np.array([10,30, 40, 20], dtype='int32'),
                    'E': pd.Categorical(['kindA', 'kindB', 'kindC', 'kindC']),
                    'F': 'smart phone'})
df
```

	B	A	C	D	E	F
0	2020-09-27	삼성	1000.5	10	kindA	smart phone
1	2020-09-27	삼성	900.0	30	kindB	smart phone
2	2020-09-27	삼성	950.0	40	kindC	smart phone
3	2020-09-27	삼성	1100.0	20	kindC	smart phone

DataFrame에 열 추가 방법

```
>>> df['three']=df['one']+df['two'] # column addition
>>> df
   one  two  three
a  1.0    1    2.0
b  2.0    2    4.0
c  3.0    3    6.0
d  NaN    4    NaN
>>> del df['three'] # column deletion
```

DataFrame에 행 첨가 및 행 삭제


```
>>> df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
>>> df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])
>>> df = df.append(df2) # append rows
>>> df
   a  b
0  1  2
1  3  4
0  5  6
1  7  8
>>> df = df.drop(0) # drop rows with label 0
>>> df
   a  b
1  3  4
1  7  8
```

DataFrame 전치

```
>>> df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
>>> df
   a  b
0  1  2
1  3  4
0  5  6
1  7  8
>>> df.T # Transpose
   0  1
a  1  3
b  2  4
>>> df.axes
[RangeIndex(start=0, stop=2, step=1),
 Index(['a', 'b'], dtype='object')]
>>> df.dtypes
a    int64
b    int64
dtype: object
```

```
>>> df.ndim
2
>>> df.shape
(2, 2)
>>> df.size
4
>>> df.values
array([[1, 2],
       [3, 4]], dtype=int64)
>>> df.head(2)
   a  b
0  1  2
1  3  4
>>> df.tail(2)
   a  b
0  1  2
1  3  4
```

문제풀이

- Pandas에서 2차원 저장구조를 생성하는 방법을 말해보세요.

- Pandas에서 생성한 2차원 저장구조를 조작하는 다음과 같은 방법을 말해보세요.
  - 열, 행 추가
  - 열, 행 삭제
  - 전치

요약

- Pandas에서 2차원 저장구조를 생성하는 방법을 공부하였음.

- Pandas에서 생성한 2차원 저장구조를 조작하는 다음과 같은 방법을 공부하였음
  - 열, 행 추가
  - 열, 행 삭제
  - 전치

01

Pandas 소개

- 개요 및 설치
- 1차원 저장 구조

02

Pandas 2차원 데이터 처리

- 2차원 데이터 구조
- 2차원 데이터 조작

03

Pandas 통계 및 함수

- 통계 기능
- Pandas 함수

학습목표

- Python Pandas 에서 제공하는 부가적인 기능을 공부한다.
  - 통계 기능
  - 데이터 특성을 파악하고 가공을 위한 기능

Pandas 통계 기능


```
>>> s = pd.Series(np.random.randn(50))
>>> s.sample(n=3) # 3 random elements
29   -0.869480
17    0.029184
45    1.603979
dtype: float64
>>> df = pd.DataFrame(np.random.randn(50, 4),
columns=list('ABCD'))
>>> df.sample(frac=0.1, replace=True) #a random 10%, with
replacement
          A         B         C         D
3   0.639683 -0.511654  2.634091  0.588999
7  -0.880275 -1.023833  1.586016 -0.648274
31 -1.008890 -0.930047  0.048127 -0.209305
39  0.799976 -0.558303  1.214371 -0.086115
3   0.639683 -0.511654  2.634091  0.588999
```

Pandas 통계 기능

```
>>> df = pd.DataFrame([[1, 2], [3, 4], [5, 6]], columns = ['a','b'])
>>> df.sum() # axis=0
a    9
b   12
dtype: int64
>>> df.sum(1) # axis=1
0    3
1    7
2   11
dtype: int64
>>> df.mean() # axis=0
a   3.0
b   4.0
dtype: float64
>>> df.mean(1) # axis=1
0   1.5
1   3.5
2   5.5
dtype: float64
```

```
>>> df.std() # axis=0
a   2.0
b   2.0
dtype: float64
>>> df.std(1) # axis=1
0   0.707107
1   0.707107
2   0.707107
dtype: float64
>>> df.describe()
              a    b
count  3.0  3.0
mean   3.0  4.0
std    2.0  2.0
min    1.0  2.0
25%    2.0  3.0
50%    3.0  4.0
75%    4.0  5.0
max    5.0  6.0
```

Pandas 함수

메서드	설명
count	NA 값을 제외한 값의 수를 반환
describe	시리즈 혹은 데이터프레임의 각 열에 대한 요약 통계
min, max	최소, 최대값
argmin, argmax	최소, 최대값을 갖고 있는 색인 위치 반환
idxmin, idxmanx	최소 최대값 갖고 있는 색인의 값 반환
quantile	0부터 1까지의 분위수 계산
sum	합
mean	평균
median	중위값

Pandas 함수

mad	평균값에서 절대 평균편차
var	표본 분산
std	표본 정규분산
skew	표본 비대칭도
kurt	표본 첨도
cumsum	누적 합
cummin, cummax	누적 최소값, 누적 최대값
cumprod	누적 곱
diff	1차 산술차 (시계열 데이터 사용시 유용)
pct_change	퍼센트 변화율 계산

Pandas 함수

메서드	설명
dropna	누락된 데이터가 있는 축(행, 열)을 제외 어느 정도의 누락 데이터까지 용인할 것인지 지정 가능
fillna	누락된 데이터를 대신할 값을 채우거나 'ffill', 'bfill' 같은 메서드 적용 가능
isnull	누락되거나 NA인 값을 알려주는 불리언 값 객체 반환
notnull	isnull의 반대

Pandas 함수

- Pandas의 read_csv 함수를 이용하여 데이터 불러오기
  - iris.csv 파일을 로컬 저장소에 저장

```
import pandas as pd

iris_df = pd.read_csv("iris.csv")
print(iris_df)
```

Pandas 함수

- head() 함수
  - data frame의 처음 5개 행을 출력
    - iris_df.head()
  - head()에 숫자를 입력해 사용할 수 있음
    - iris_df.head(10)
- tail() 함수
  - 마지막 5개 행을 출력
    - iris_df.tail()
  - 마지막 2개 레코드를 보고싶다면, 다음과 같이 숫자를 입력
    - iris_df.tail(2)

Pandas 함수

- info()
  - 데이터셋에 대한 필수 세부정보를 제공

  - 특징
    - 행의 수
    - 열의 수
    - Null이 아닌 값의 수
    - 각 열의 데이터 타입
    - Data frame이 사용 중인 메모리 양

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

Pandas 함수

- shape
  - 행과 열의 크기를 출력
    - iris_df.shape
    - (150,5)
  - 모양에는 괄호가 없으며 출력은 (행, 열) 형식의 간단한 tuple이다
  - 데이터를 정리하고 변환할 때 자주 사용
    - 예를 들어, 몇 가지 기준에 따라 일부 행을 필터링한 다음 제거된 행의 수를 빠르게 알고 싶을 때

Pandas 함수

- describe()
  - 연속 변수의 분포 요약을 출력
    - iris_df.describe()

  - 특징
    - 행의 수
    - 카테고리의 고유 개수
    - 상위 카테고리
    - 상위 카테고리의 빈도

	sepal.length	sepal.width	petal.length	petal.width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Pandas 함수

- columns
  - 데이터셋의 열 이름을 출력
    - iris_df.columns

```
Index(['sepal.length', 'sepal.width', 'petal.length', 'petal.width',
       'variety'],
      dtype='object')
```

  - 열을 사용해 열 이름을 바꿀 수 있음

Pandas 함수

- rename()
  - dict를 통해 특정 또는 모든 열의 이름 바꾸기

```
iris_df.rename(columns={'variety': 'species',}, inplace=True)
iris_df.columns
```

```
Index(['sepal.length', 'sepal.width', 'petal.length', 'petal.width',
       'species'],
      dtype='object')
```

Pandas 함수

- Data frame 조작
  - 다음 테이블 출력

  - 소스코드

```
subset  = irisdf[['sepal.width', 'petal.length']]

subset.head()
```

Pandas 함수

- Data frame 조작
  - Species = Setosa 인 데이터 출력
    - iris_df[iris_df['species'] == 'Setosa'].head(5)

	sepal.length	sepal.width	petal.length	petal.width	species
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

Pandas 함수

- Data frame 조작
  - Petal.length이 1.4 이상인 값을 출력
    - iris_df[iris_df['Petal.length'] >= 1.4].head(5)

  - Speices가 Setosa 혹은 Virginica인 데이터 출력
    - VerOR (|) operator 사용

```
iris_df[(iris_df['species'] == 'Setosa') |
(iris_df['species'] == 'Virginica')].head(5)
```

Pandas 함수

- Data frame 조작
  - 값 별로 정렬하기 위한 sort_value() 함수
    - 정렬하려는 data frame의 열 이름을 나타내는 “by” 인수
  - sort_value() 예시
    - iris_df.sort_values(by='sepal.length', ascending=False).head(5)

Pandas 함수

- 누락된 값 다루기
  - 누락되거나 null 값
    - 데이터셋의 품질을 가리킴

  - isnull() 함수를 사용해 누락된 값이 있는 위치를 확인
    - iris_df.isnull().tail(5)

- 누락된 값 다루기
  - 누락된 값의 요약을 확인
    - iris_df.isnull().sum()

Pandas 함수

- 누락된 값 다루기
  - 누락된 값의 요약을 확인
    - iris_df.isnull().sum()
- Null 값 제거
  - dropna() 함수를 사용하여 null 값이 있는 행 제거
    - iris_df.dropna(inplace=True)
    - iris_df.isnull().sum()

Pandas 함수

- Null 값 제거
  - dropna()를 사용해 열을 제거 할 수도 있음
    - iris_df.dropna(axis=1)
      - 0, or 'index' : 누락된 값이 포함된 행 제거
      - 1, or 'columns' : 누락된 값이 포함된 열 제거
  - 힌트
    - 누락된 데이터가 적은 경우에만 null 데이터를 제거하는 것이 좋음
- 데이터 대체
  - Null을 채우기 위한 fillna() 함수
    - iris_df[sepal.lenght'].fillna(iris_df['sepal.length'].mean(), inplace=True)
    - iris_df.isnull().sum()

문제풀이

- Pandas에서 제공하는 통계 기능 세가지를 설명하시오.

- Pandas에서 제공하는 다음과 같은 함수를 설명하시오.
  - Info
  - describe
  - dropna
  - fillna

요약

- Pandas에서 제공하는 통계 기능을 공부하였음.
- Pandas에서 제공하는 함수를 공부하였음.