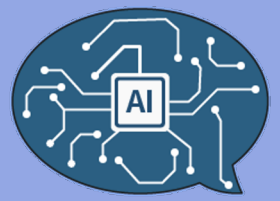
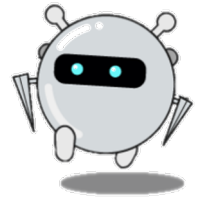
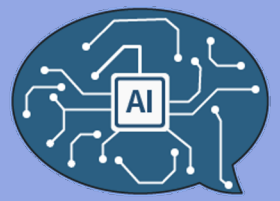
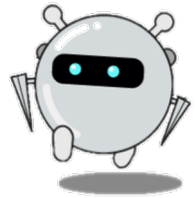


Numpy & Pandas





Pandas



Pandas 소개

- Python Data Analysis Library
- 대표적인 Python 기반 정형 데이터 분석 라이브러리
- import pandas as pd

특징

- 테이블 형태의 데이터를 분석/처리할 수 있는 다양한 함수 제공
- Excel로 할 수 있는 모든 연산/기능 수행 가능
- 데이터 통계, 크롤링, 시각화 등 가능
- Python 자료구조(List, Tuple, Dictionary, numpy array)와 호환
- 외부 데이터(CSV, txt, Excel, SQL database, XML, pdf 등) 불러올 수 있음

Pandas 자료구조

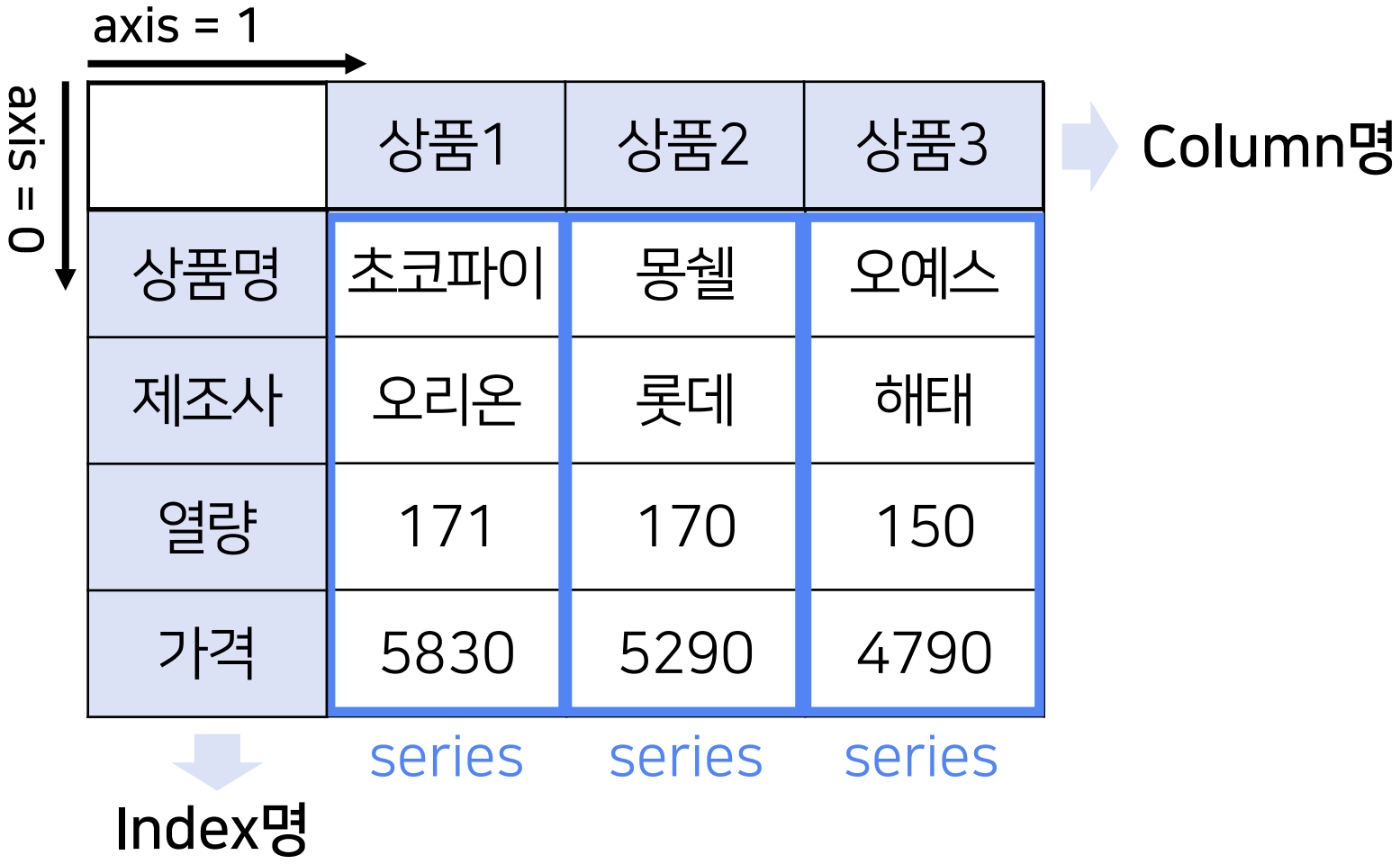
(1) Series ; 1차원 데이터

상품명	초코파이
제조사	오리온
열량	171
가격	5830

Index (데이터 위치)

Value (데이터 값)

(2) Data Frame ; 2차원 데이터



OUTTA

Series

OUTTA

Series 생성 : pd.Series(data)

상품명	초코파이
제조사	오리온
열량	171
가격	5830

```
0      초코파이
1       오리온
2       171
3      5830
dtype: object
```

방법1: List 사용

```
series = pd.Series(['초코파이', '오리온', 171, 5830])
print(series)
```

Series 생성 : pd.Series(data index=)

상품명	초코파이
제조사	오리온
열량	171
가격	5830

```
상품명      초코파이
제조사       오리온
열량         171
가격        5830
dtype: object
```

방법1: List 사용 +) index 설정

```
series = pd.Series(['초코파이', '오리온', 171, 5830],
                   index = ['상품명', '제조사', '열량', '가격'])
print(series)
```

Series 생성 : pd.Series(data, index=)

상품명	초코파이
제조사	오리온
열량	171
가격	5830

```
상품명      초코파이
제조사       오리온
열량         171
가격        5830
dtype: object
```

방법2: Tuple 사용

+) Tuple 자료형: (data1, data2, data3)

```
food_info = ('초코파이', '오리온', 171, 5830)
index_name = ('상품명', '제조사', '열량', '가격')
series = pd.Series(food_info, index = index_name)
print(series)
```

Series 생성

key	value
상품명	초코파이
제조사	오리온
열량	171
가격	5830

```
상품명      초코파이
제조사       오리온
열량         171
가격        5830
dtype: object
```

방법3: Dictionary 사용

+) Dictionary 자료형: {key : value}

(예시) {'이름' : '홍수민', '담당강의' : 'pandas'}

```
series = pd.Series({'상품명':'초코파이', '제조사':'오리온',
                    '열량':171, '가격':5830})

print(series)
```

Series 생성 : pd.Series(data, index=)

List, Dictionary, Tuple, ndarray 등

방법1: List 사용

```
series = pd.Series(['초코파이', '오리온', 171, 5830],
                   index = ['상품명', '제조사', '열량', '가격'])
print(series)
```

방법2: Tuple 사용

```
food_info = ('초코파이', '오리온', 171, 5830)
index_name = ('상품명', '제조사', '열량', '가격')
series = pd.Series(food_info, index = index_name)
print(series)
```

방법3: Dictionary 사용

```
series = pd.Series({'상품명':'초코파이', '제조사':'오리온',
                    '열량':171, '가격':5830})
print(series)
```

Series 속성 확인

객체 이름: obj

상품명	초코파이
제조사	오리온
열량	NaN
가격	5830

Index Value

[결측치(NaN) 확인]

- obj.isnull()

```
상품명     False
제조사     False
열량        True
가격      False
dtype: bool
```

- obj.notnull()

```
상품명      True
제조사      True
열량       False
가격       True
dtype: bool
```

- **obj.index**: series 객체의 index 알 수 있음
- **obj.values**: series 객체의 데이터(value) 알 수 있음

Series 데이터 선택

객체 이름: obj

상품명	초코파이
제조사	오리온
열량	171
가격	5830

상품의 제조사는 어디?

방법1: index 사용

- obj['제조사']

문자

방법2: loc 사용 #loc: location

- obj.loc['제조사']

숫자

방법3: iloc 사용 #iloc: index location

- obj.iloc[1]

Series 연산 - 단일 Series 내 연산

객체 이름: obj

a	1
b	2
c	3
d	4

덧셈

```
obj + 2
```

```
a    3
b    4
c    5
d    6
dtype: int64
```

뺄셈

```
obj - 2
```

```
a   -1
b    0
c    1
d    2
dtype: int64
```

곱셈

```
obj * 2
```

```
a    2
b    4
c    6
d    8
dtype: int64
```

나눗셈

```
obj / 2
```

```
a    0.5
b    1.0
c    1.5
d    2.0
dtype: float64
```

Series 연산 - Series간 연산

객체1: obj1

a	1
b	2
c	3
d	4

객체2: obj2

a	4
e	3
f	2
b	1

a	1
b	2
c	3
d	4
e	NaN
f	NaN

a	4
b	1
c	NaN
d	NaN
e	3
f	2

덧셈

```
obj1 + obj2
```

```
obj1.add(obj2)
```

```
obj2.add(obj1)
```

```
a    5.0
b    3.0
c    NaN
d    NaN
e    NaN
f    NaN
dtype: float64
```

Series 연산 - Series간 연산

객체1: obj1

a	1
b	2
c	3
d	4

객체2: obj2

a	4
e	3
f	2
b	1

a	1
b	2
c	3
d	4
e	0
f	0

a	4
b	1
c	0
d	0
e	3
f	2

덧셈

```
obj1 + obj2
```

```
obj1.add(obj2)
```

```
obj2.add(obj1)
```

```
a    5.0
b    3.0
c    NaN
d    NaN
e    NaN
f    NaN
dtype: float64
```

```
obj1.add(obj2, fill_value=0)
```

```
a    5.0
b    3.0
c    3.0
d    4.0
e    3.0
f    2.0
```

OUTTA

Series 연산 - Series간 연산

> 동일한 Index 가진 값끼리 연산

덧셈

- obj1 + obj2
- obj1.add(obj2, fill_value=)
- obj2.add(obj1, fill_value=)

뺄셈 순서 주의!

- obj1 - obj2
- obj1.sub(obj2, fill_value=)

곱셈

- obj1 * obj2
- obj1.mul(obj2, fill_value=)
- obj2.mul(obj1, fill_value=)

나눗셈 순서 주의!

- obj1 / obj2
- obj1.div(obj2, fill_value=)

DataFrame

DataFrame 생성 : pd.DataFrame(data)

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

	0	1	2
0	초코파이	몽쉘	오예스
1	오리온	롯데	해태
2	171	170	150
3	5830	5290	4790

방법1: List 사용

```
data = [['초코파이', '몽쉘', '오예스'], ['오리온', '롯데', '해태'],
 [171, 170, 150], [5830, 5290, 4790]]
df = pd.DataFrame(data)
```

DataFrame 생성

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

방법1: List 사용

+) index명, column명 설정

```
data = [['초코파이', '몽쉘', '오예스'], ['오리온', '롯데', '해태'],
 [171, 170, 150], [5830, 5290, 4790]]
df = pd.DataFrame(data,
                  index=['상품명', '제조사', '열량', '가격'],
                  columns=['상품1', '상품2', '상품3'])
```

DataFrame 생성

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

방법2: Dictionary 사용

```
data = {'상품1' : ['초코파이', '오리온', 171, 5830],
        '상품2' : ['몽쉘', '롯데', 170, 5290],
        '상품3' : ['오예스', '해태', 150, 4790]}
df = pd.DataFrame(data, index = ['상품명', '제조사', '열량', '가격'])
```

DataFrame 생성

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

방법2: Dictionary 사용 +) series 사용

```
product_1 = pd.Series({'상품명':'초코파이', '제조사':'오리온', '열량':171, '가격':5830})
product_2 = pd.Series({'상품명':'몽쉘', '제조사':'롯데', '열량':170, '가격':5290})
product_3 = pd.Series({'상품명':'오예스', '제조사':'해태', '열량':150, '가격':4790})
df = pd.DataFrame({'상품1' : product_1, '상품2' : product_2, '상품3': product_3})
```

DataFrame 속성 확인

객체 이름: df

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

- **df.index**: index명 확인

```
Index(['상품명', '제조사', '열량', '가격'],
```

- **df.columns**: column명 확인

```
Index(['상품1', '상품2', '상품3']
```

- **df.shape**: (행, 열) 크기 확인

```
(4, 3)
```

DataFrame 속성 확인

객체 이름: df

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

2022년 7월 기온

- df.head(n): 데이터 상단 n개 행 출력

```
df.head(2)
```

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9

- df.tail(n): 데이터 하단 n개 행 출력

```
df.tail(2)
```

	최저기온	최고기온
강원	20.4	34.7
제주	23.2	36.0

DataFrame 속성 확인

객체 이름: df

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

- df.describe(): 칼럼(열)별 기술통계량 출력

	최저기온	최고기온
count	7.000000	7.000000
mean	21.000000	34.271429
std	1.096966	1.998094
min	20.000000	31.000000
25%	20.250000	33.100000
50%	20.800000	34.700000
75%	21.250000	35.600000
max	23.200000	36.800000

DataFrame 속성 확인

객체 이름: df

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

- df.info(): 칼럼별 결측치, 데이터타입 등 출력

```
<class 'pandas.core.frame.DataFrame'>
Index: 7 entries, 서울 to 제주
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   최저기온     7 non-null      float64
 1   최고기온     7 non-null      float64
dtypes: float64(2)
memory usage: 168.0+ bytes
```

DataFrame 속성 확인

- .index: index명 확인
- .columns: column명 확인
- .shape: (행, 열) 크기 확인
- .head(n): 데이터 상단 n개 행 출력
- .tail(n): 데이터 하단 n개 행 출력
- .describe(): 칼럼별 기술통계량 출력
- .info(): 칼럼별 결측치, 데이터타입 등 출력

DataFrame 데이터 선택

객체 이름: df

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

부산의 기온은?

Row 선택

[1] loc 사용

```
df.loc['부산']
```

[2] iloc 사용

```
df.iloc[1]
```

```
최저기온     20.1
최고기온     32.9
Name: 부산, dtype: float64
```

DataFrame 데이터 선택

객체 이름: df

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

지역들의 최고기온은?

Column 선택

[1] 칼럼명 사용

```
df['최고기온']
```

[2] index 번호 사용

```
df[df.columns[1]]
```

```
서울      31.0
부산      32.9
대구      36.8
울산      33.3
경기      35.2
강원      34.7
제주      36.0
Name: 최고기온, dtype: float64
```

DataFrame 데이터 선택

객체 이름: df

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

울산의 최저기온은?

특정 데이터 선택

[1] loc 사용

```
df.loc['울산'][0]
```

```
df.loc['울산', '최저기온']
```

[2] iloc 번호 사용

```
df.iloc[3][0]
```

```
df.iloc[3, 0]
```

[3] column 접근 → row 접근

```
df['최저기온'][3]
```

DataFrame 데이터 수정 (추가/삭제)

객체 이름: df

	최저기온	최고기온	평균기온
서울	20.8	31.0	25.90
부산	20.1	32.9	26.50
대구	21.4	36.8	29.10
울산	20.0	33.3	26.65
경기	21.1	35.2	28.15
강원	20.4	34.7	27.55
제주	23.2	36.0	29.60
전주	21.2	35.6	

행 & 열 추가

- df.loc['새로운 행 이름'] = 데이터 값

```
df.loc['전주'] = [21.2, 35.6]
```

- df['새로운 열 이름'] = 데이터 값

```
df['평균기온']
        = (df['최저기온']+df['최고기온'])/2
```

행 & 열 삭제

- df.drop(행 이름, axis = 0)

```
df.drop('전주', axis = 0)
```

- df.drop(열 이름, axis = 1)

```
df.drop('평균기온', axis=1)
```

OUTTA

DataFrame 데이터 수정 (정렬)

객체 이름: df

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

Index 기준

df.sort_index(ascending=True or False)

	최저기온	최고기온
강원	20.4	34.7
경기	21.1	35.2
대구	21.4	36.8
부산	20.1	32.9
서울	20.8	31.0
울산	20.0	33.3
전주	21.2	35.6
제주	23.2	36.0

DataFrame 데이터 수정 (정렬)

객체 이름: df

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

Index 기준

df.sort_index(ascending=True or False)

	최저기온	최고기온
제주	23.2	36.0
전주	21.2	35.6
울산	20.0	33.3
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
경기	21.1	35.2
강원	20.4	34.7

DataFrame 데이터 수정 (정렬)

객체 이름: df

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0


기온이 높은 지역?

데이터 값 기준

df.sort_values('최고기온',
ascending= False)

	최저기온	최고기온
대구	21.4	36.8
제주	23.2	36.0
전주	21.2	35.6
경기	21.1	35.2
강원	20.4	34.7
울산	20.0	33.3
부산	20.1	32.9
서울	20.8	31.0

OUTTA

DataFrame 데이터 연산

함수	설명
.**sum**()	데이터의 합을 계산
.**mean**()	데이터의 평균을 계산
.**std**(), .**var**()	데이터의 표준편차, 분산을 계산
.**min**(), .**max**()	데이터의 최솟값, 최댓값 반환
.**idxmin**(), .**idxmax**()	데이터의 최솟값, 최댓값이 위치한 인덱스 반환
.**cumsum**()	첫 원소부터 끝 원소까지의 누적합 반환
.**cumprod**()	첫 원소부터 끝 원소까지의 누적곱 반환
.**median**()	데이터의 중앙값 반환
.**quantile**(n)	n분위 수 반환

DataFrame 데이터 연산

객체 이름: df

axis = 1

axis = 0

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

최저기온 / 최고기온이 가장 높은 지역과 가장 낮은 지역의 차이는?

'최저기온'열 중 최댓값 - ' 최저기온'열 중 최솟값

'최고기온'열 중 최댓값 - ' 최고기온'열 중 최솟값

→ .apply(): 괄호 안 함수를 모든 데이터에 적용

```
f = lambda x : x.max() - x.min()
```

```
df.apply(f, axis = 0)
```

```
최저기온     3.2
최고기온     5.8
dtype: float64
```

OUTTA

DataFrame 데이터 연산

객체 이름: df

axis = 1

axis = 0

	최저기온	최고기온
서울	20.8	31.0
부산	20.1	32.9
대구	21.4	36.8
울산	20.0	33.3
경기	21.1	35.2
강원	20.4	34.7
제주	23.2	36.0

지역별 최고기온과 최저기온의 편차는?

각 지역의 최고기온 - 최저기온

→ .apply(): 괄호 안 함수를 모든 데이터에 적용

```
f = lambda x : x.max() - x.min()
```

```
df.apply(f, axis = 1)
```

```
서울     10.2
부산     12.8
대구     15.4
울산     13.3
경기     14.1
강원     14.3
제주     12.8
dtype: float64
```

파일 입출력

OUTTA

Pandas 파일 입출력

파일 읽어오기

- pd.read_csv(' 파일경로/파일이름.csv',
  header = column명으로 지정하고 싶은 row번호,
  index_col = index명로 지정하고 싶은 column번호,
  skiprows = 스킵하고 싶은 row 개수
  nrows = 위에서부터 읽어오고 싶은 row 개수
  encoding = 인코딩 방식)

- pd.read_excel(' 파일경로/파일이름.csv')

Pandas 파일 입출력

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

파일 읽어오기

- pd.read_csv(' 파일경로/파일이름.csv',
  header = column명으로 지정하고 싶은 row번호,
  index_col = index명로 지정하고 싶은 column번호,
  skiprows = 스킵하고 싶은 row 개수)

Pandas 파일 입출력

	상품1	상품2	상품3
상품명	초코파이	몽쉘	오예스
제조사	오리온	롯데	해태
열량	171	170	150
가격	5830	5290	4790

파일 읽어오기

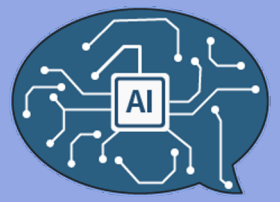
- pd.read_csv(' 파일경로/파일이름.csv',
  header = column명으로 지정하고 싶은 row번호,
  index_col = index명로 지정하고 싶은 column번호,
  skiprows = 스킵하고 싶은 row 개수)

Pandas 파일 입출력

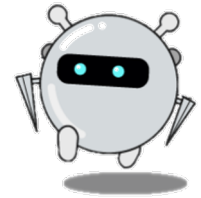
파일 저장하기

- df.to_csv(' 파일경로/파일이름.csv',
  header = True or False,
  index = True or False,
  encoding = 인코딩 방식)

- df.to_excel(' 파일경로/파일이름.csv')



감사합니다



OUTTA