빅데이터 기반 AI 응용 솔루션 개발자 전문과정

교과목명 : 분석라이브러리 활용

- 평가일 : 22.7.8
- 성명 : 박혜린
- 점수 :

Q1. 표준정규분포 기반의 2행 3열 배열을 랜덤하게 생성하여 크기, 자료형, 차원을 출력하세요.

In [1]:

```
import pandas as pd
import numpy as np
```

In [450]:

```
a = np.random.randn(1,2,3)
print(a)

# 크기
```

```
[[[ 3.42600956 -0.38859226 -1.40178503]
  [ 0.01494693  0.90486641 -0.33844567]]]
```

In [448]:

```
# 자료형
a.dtype
```

Out[448]:

```
dtype('float64')
```

In [451]:

```
# 차원
a.ndim
```

Out[451]:

```
3
```

Q2. arange(), reshape() 이용 1차원 2차원 3차원 배열을 아래와 같이 생성하세요.

[0 1 2 3 4 5 6 7 8 9]

[[0 1 2 3 4]
[5 6 7 8 9]]

[[[0 1 2 3 4]
[5 6 7 8 9]]]

In [164]:

```
a = np.arange(10).reshape(10)
print(a)
b = a.reshape(2,5)
print(b)
c = np.arange(10).reshape(1,2,5)
print(c)
```

```
[0 1 2 3 4 5 6 7 8 9]
[[0 1 2 3 4]
 [5 6 7 8 9]]
[[[0 1 2 3 4]
  [5 6 7 8 9]]]
```

Q3. 1 ~ 100 까지 배열에서 3과 7의 공배수인 것만을 출력하세요.

In [453]:

```
np.arange(1,101).map(lambda x : x%3==0 & x%7==0)
```

```
---------------------------------------------------------------------
--------
AttributeError                            Traceback (most recent call
last)
Input In [453], in <cell line: 1>()
----> 1 np.arange(1,101).map(lambda x : x%3==0 & x%7==0)

AttributeError: 'numpy.ndarray' object has no attribute 'map'
```

In [ ]:

```

```

Q4. 아래 3차원 배열을 생성하여 출력한 후 1차원으로 변환하여 출력하세요.(reshape() 사용)

[[[0 1 2 3 4]
[5 6 7 8 9]]

[[10 11 12 13 14]
[15 16 17 18 19]]

[[20 21 22 23 24]
[25 26 27 28 29]]]

In [63]:

```
a = np.arange(0,30).reshape(3,2,5)
print(a)
print(a.ndim)

b = a.reshape(30)
print(b)
b.ndim
```

```
[[[ 0  1  2  3  4]
  [ 5  6  7  8  9]]

 [[10 11 12 13 14]
  [15 16 17 18 19]]

 [[20 21 22 23 24]
  [25 26 27 28 29]]]
3
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
23
 24 25 26 27 28 29]
```

Out[63]:

```
1
```

Q5. array2d에서 인덱스를 이용해서 값을 선택하고 리스트로 아래와 같이 출력하세요.

arr2d = np.arange(1,10).reshape(3,3)

[3, 6]

[[1, 2],
[4, 5]]

[[1, 2, 3]
[4, 5, 6]]

In [76]:

```
arr2d = np.arange(1,10).reshape(3,3)
print(arr2d)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [85]:

```
print(arr2d[:2,2],'\n')
print(arr2d[:2,:2],'\n')
print(arr2d[:2,:])
```

```
[3 6]

[[1 2]
 [4 5]]

[[1 2 3]
 [4 5 6]]
```

Q6. zeros_like, ones_like, full_like 함수 사용 예를 작성하세요.

In [110]:

```
a = np.random.randint(1,11,(5,5))
print(a,'\n')
b = np.zeros_like((a))
print(b,'\n')
c = np.ones_like((a))
print(c,'\n')
d = np.full_like((a),5)
print(d)
```

```
[[ 3  5  8  2  6]
 [ 7 10  7  8  2]
 [ 8  5  4  6  1]
 [ 2  7 10  4  2]
 [ 9  5  9  5  8]]

[[0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]]

[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]

[[5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]
 [5 5 5 5 5]]
```

Q7. 10 ~ 20 사이의 정수 난수로 10행 5열 2차원 배열을 생성하고 저장한 후 다시 불러내서 출력하세요.

In [293]:

```
random_array = np.random.randint(10,20,(2,10,5))
random_array
```

Out[293]:

```
array([[[19, 17, 10, 18, 16],
        [19, 12, 16, 15, 16],
        [15, 18, 16, 17, 13],
        [13, 11, 16, 11, 13],
        [15, 10, 19, 18, 11],
        [10, 19, 14, 19, 19],
        [14, 14, 15, 14, 13],
        [13, 16, 19, 18, 19],
        [18, 12, 16, 15, 11],
        [15, 16, 15, 11, 18]],

       [[12, 11, 15, 16, 13],
        [14, 15, 13, 18, 14],
        [15, 18, 11, 10, 16],
        [16, 10, 14, 15, 13],
        [13, 18, 13, 12, 11],
        [11, 17, 16, 18, 17],
        [16, 16, 17, 19, 15],
        [14, 17, 13, 16, 16],
        [15, 19, 10, 17, 12],
        [15, 16, 19, 18, 13]]])
```

In [455]:

```
random_array = pd.save_csv('dataset/random_array.csv')
```

```
---------------------------------------------------------------------
--------
AttributeError                            Traceback (most recent call
last)
Input In [455], in <cell line: 1>()
----> 1 random_array = pd.save_csv('dataset/random_array.csv')

File ~\anaconda3\envs\cakd7\lib\site-packages\pandas\__init__.py:261
, in __getattr__(name)
    257     from pandas.core.arrays.sparse import SparseArray as _Sp
arseArray
    259     return _SparseArray
--> 261 raise AttributeError(f"module 'pandas' has no attribute '{nam
e}'")

AttributeError: module 'pandas' has no attribute 'save_csv'
```

In [ ]:

```

```

Q8. df = sns.load_dataset('titanic')로 불러와서 다음 작업을 수행한 후 출력하세요.

- 전체 칼럼중 'survived'외에 모든 칼럼을 포함한 df_x를 산출한 후 dataset/df_x.pkl로 저장한다.
- df_x.pkl을 데이터프레임 df_x 이름으로 불러온 후 앞 5개 행을 출력한다.

In [458]:

```
import seaborn as sns

df = sns.load_dataset('titanic')
df.head()
```

Out[458]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True

In [457]:

```
df_x = df.drop(columns=['survived'])
df_x.head()
```

Out[457]:

	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	em
0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [ ]:

```
df_x = pd.
```

In [ ]:

```

```

Q9. df = sns.load_dataset('titanic')로 불러와서 deck 열에서 NaN 갯수를 계산하세요.

In [116]:

```
df['deck'].isnull().sum()
```

Out[116]:

688

Q10. Q9의 df에서 각 칼럼별 null 개수와 df 전체의 null 개수를 구하세요.

In [120]:

```
df.isnull().sum()
```

Out[120]:

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [121]:

```
df.isnull().sum().sum()
```

Out[121]:

```
869
```

아래 tdf 데이터프레임에서 Q11 ~ Q12 작업을 수행하세요.

In [379]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
tdf = df[['survived','sex','age','class']]
tdf.head()
```

Out[379]:

	survived	sex	age	class
0	0	male	22.0	Third
1	1	female	38.0	First
2	1	female	26.0	Third
3	1	female	35.0	First
4	0	male	35.0	Third

Q11. age를 7개 카테고리로 구분하는 새로운 칼럼 'cat_age'를 생성하여 출력하세요. 단, 카테고리 구분을 수행하는 사용자 함수를 만들고 그 함수를 age 칼럼에 매핑하여 결과를 tdf1에 저장하고 출력하세요.

[카테고리]
age <= 5: cat = 'Baby'
age <= 12: cat = 'Child'
age <= 18: cat = 'Teenager'
age <= 25: cat = 'Student'
age <= 60: cat = 'Adult'
age > 60 : cat = 'Elderly'

In [336]:

```
tdf['age'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 891 entries, 0 to 890
Series name: age
Non-Null Count  Dtype
--------------  -----
714 non-null    float64
dtypes: float64(1)
memory usage: 7.1 KB
```

In [380]:

```
def catage(x):
    cat=''
    if x <= 5:
        return 'Baby'
    elif x <=12:
        return 'Child'
    elif x <=18:
        return 'Teenager'
    elif x <=25:
        return 'Student'
    elif x <=60:
        return 'Adult'
    elif x > 60:
        return 'Elderly'
    else :
        pass
```

In [386]:

```
tdf1 = tdf.copy()
tdf1['cat_age'] = tdf1['age']
tdf1.head()
```

Out[386]:

	survived	sex	age	class	cat_age
0	0	male	22.0	Third	22.0
1	1	female	38.0	First	38.0
2	1	female	26.0	Third	26.0
3	1	female	35.0	First	35.0
4	0	male	35.0	Third	35.0

In [387]:

```
tdf1 = tdf1.cat_age.apply(catage)
tdf1
```

Out[387]:

```
0      Student
1        Adult
2        Adult
3        Adult
4        Adult
        ...
886      Adult
887    Student
888       None
889      Adult
890      Adult
Name: cat_age, Length: 891, dtype: object
```

Q12. tdf1의 sex, class 칼럼을 '_'으로 연결한 'sc'칼럼을 추가한 후 아래와 같이 출력하세요.

In [ ]:

```

```

In [ ]:

```

```

Q13. join() 메소드는 두 데이터프레임의 행 인덱스를 기준으로 결합한다. 2개의 주식데이터를 가져와서 join() 메소드로 아래와 같이 결합한 후 다음 사항을 수행하세요.

- df1과 df2의 교집합만 출력되도록 결합하여 df3에 저장하고 출력
- df3에서 중복된 칼럼을 삭제한 후 블린 인덱싱을 이용하여 eps가 3000 보다 적거나 stock_name이 이마트인 데이터를 선택하여 데이터프레임을 생성하고 df4 이름으로 저장 및 출력하세요.(단, '<' 와 '==' 를 반드시 사용해야 함)

In [390]:

```
df1 = pd.read_excel('./dataset/stock price.xlsx', index_col='id')
df2 = pd.read_excel('./dataset/stock valuation.xlsx', index_col='id')
```

In [393]:

```
df3 = df1.join(df2, how='inner')
df3
```

Out[393]:

id	stock_name	value	price	name	eps	bps	per	pbr
130960	CJ E&M	58540.666667	98900	CJ E&M	6301.333333	54068	15.695091	1.829178
139480	이마트	239230.833333	254500	이마트	18268.166667	295780	13.931338	0.860437
145990	삼양사	82750.000000	82000	삼양사	5741.000000	108090	14.283226	0.758627
185750	종근당	40293.666667	100500	종근당	3990.333333	40684	25.185866	2.470259
204210	모두투어리츠	3093.333333	3475	모두투어리츠	85.166667	5335	40.802348	0.651359

In [417]:

```
df4 = df3.drop(columns =['name'])
df4.head()
```

Out[417]:

id	stock_name	value	price	eps	bps	per	pbr
130960	CJ E&M	58540.666667	98900	6301.333333	54068	15.695091	1.829178
139480	이마트	239230.833333	254500	18268.166667	295780	13.931338	0.860437
145990	삼양사	82750.000000	82000	5741.000000	108090	14.283226	0.758627
185750	종근당	40293.666667	100500	3990.333333	40684	25.185866	2.470259
204210	모두투어리츠	3093.333333	3475	85.166667	5335	40.802348	0.651359

In [ ]:

```
df3에서 중복된 칼럼을 삭제한 후 블린 인덱싱을 이용하여 eps가 3000 보다 적거나 stock_name이 이
데이터프레임을 생성하고 df4 이름으로 저장 및 출력하세요.(단, '<' 와 '==' 를 반드시 사용해야 함)
```

In [425]:

```
df4 = df4.applymap(lambda columns : ('eps' < 3000) & ('stock_name'== '이마트'))
```

```
---------------------------------------------------------------------
--------
TypeError                                 Traceback (most recent call
last)
Input In [425], in <cell line: 1>()
----> 1 df4 = df4.applymap(lambda columns : ('eps' < 3000) & ('stock_
name'== '이마트'))

File ~\anaconda3\envs\cakd7\lib\site-packages\pandas\core\frame.py:8
924, in DataFrame.applymap(self, func, na_action, **kwargs)
   8921         return lib.map_infer(x, func, ignore_na=ignore_na)
   8922     return lib.map_infer(x.astype(object)._values, func, igno
re_na=ignore_na)
-> 8924 return self.apply(infer).__finalize__(self, "applymap")

File ~\anaconda3\envs\cakd7\lib\site-packages\pandas\core\frame.py:8
839, in DataFrame.apply(self, func, axis, raw, result_type, args, **
kwargs)
   8828 from pandas.core.apply import frame_apply
   8830 op = frame_apply(
   8831     self,
   8832     func=func,
   (...)
   8837     kwargs=kwargs,
   8838 )
-> 8839 return op.apply().__finalize__(self, method="apply")

File ~\anaconda3\envs\cakd7\lib\site-packages\pandas\core\apply.py:7
27, in FrameApply.apply(self)
    724 elif self.raw:
    725     return self.apply_raw()
--> 727 return self.apply_standard()

File ~\anaconda3\envs\cakd7\lib\site-packages\pandas\core\apply.py:8
51, in FrameApply.apply_standard(self)
    850 def apply_standard(self):
--> 851     results, res_index = self.apply_series_generator()
    853     # wrap results
    854     return self.wrap_results(results, res_index)

File ~\anaconda3\envs\cakd7\lib\site-packages\pandas\core\apply.py:8
67, in FrameApply.apply_series_generator(self)
    864 with option_context("mode.chained_assignment", None):
    865     for i, v in enumerate(series_gen):
    866         # ignore SettingWithCopy here in case the user mutates
--> 867         results[i] = self.f(v)
    868         if isinstance(results[i], ABCSeries):
    869             # If we have a view on v, we need to make a copy b
ecause
    870             #  series_generator will swap out the underlying d
ata
    871             results[i] = results[i].copy(deep=False)

File ~\anaconda3\envs\cakd7\lib\site-packages\pandas\core\frame.py:8
922, in DataFrame.applymap.<locals>.infer(x)
   8920 if x.empty:
   8921     return lib.map_infer(x, func, ignore_na=ignore_na)
```

```
-> 8922 return lib.map_infer(x.astype(object)._values, func, ignore_n
a=ignore_na)

File ~\anaconda3\envs\cakd7\lib\site-packages\pandas\_libs\lib.pyx:2
870, in pandas._libs.lib.map_infer()

Input In [425], in <lambda>(columns)
----> 1 df4 = df4.applymap(lambda columns : ('eps' < 3000) & ('stock_
name'== '이마트'))

TypeError: '<' not supported between instances of 'str' and 'int'
```

Q14. 배열 a에 대하여 3차원 자리에 2차원을 2차원 자리에 1차원을 1차원 자리에 3차원을 넣어서 변환하여 출력하세요

In [161]:

```
a = np.arange(6).reshape(1,2,3)
print(a,a.shape,'\n')

print(a.ndim)
b = a.reshape(2,3)
print(b,b.shape,'\n')
print(b.ndim)
c = b.reshape(6)
print(c,c.shape,'\n')
print(c.ndim)
d = c.reshape(1,2,3)
print(d,d.shape)
print(d.ndim)
```

```
[[[0 1 2]
  [3 4 5]]] (1, 2, 3)

3
[[0 1 2]
 [3 4 5]] (2, 3)

2
[0 1 2 3 4 5] (6,)

1
[[[0 1 2]
  [3 4 5]]] (1, 2, 3)
3
```

Q15. 'mpg'를 'kpl' 로 환산하여 새로운 열을 생성하고 반올림하여 소수점 아래 둘째 자리까지 처음 5개행을 출력하세요.

In [167]:

```
# read_csv() 함수로 df 생성
import pandas as pd
auto_df = pd.read_csv('./dataset/auto-mpg.csv')
# 열 이름을 지정
auto_df.columns = ['mpg','cylinders','displacement','horsepower','weight',
                   'acceleration','model year','origin','name']
print(auto_df.head(3))
```

```
    mpg  cylinders  displacement horsepower  weight  acceleration  mod
el year  \
0  18.0          8         307.0        130    3504          12.0
70
1  15.0          8         350.0        165    3693          11.5
70
2  18.0          8         318.0        150    3436          11.0
70

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
```

In [432]:

```
auto_df['kpl'] = auto_df['mpg']/2.352
auto_df.head()
```

Out[432]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	name	
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu	7.
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320	6.
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite	7.
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst	6.
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino	7.

In [ ]:

```

```

Q16. './dataset/stock-data.csv'를 데이터프레임으로 불러와서 datetime64 자료형으로 변환한 후에 년, 월, 일로 분리하고 year를 인덱스로 셋팅하여 출력하세요.

In [180]:

```
df_time = pd.read_csv('./dataset/stock-data.csv')
df_time.head()
```

Out[180]:

	Date	Close	Start	High	Low	Volume
0	2018-07-02	10100	10850	10900	10000	137977
1	2018-06-29	10700	10550	10900	9990	170253
2	2018-06-28	10400	10900	10950	10150	155769
3	2018-06-27	10900	10800	11050	10500	133548
4	2018-06-26	10800	10900	11000	10700	63039

In [440]:

```
from datetime import datetime
df_time['Date'].dtype
```

Out[440]:

```
dtype('O')
```

In [ ]:

```

```

Q17. titanic 데이터셋(titanic = sns.load_dataset('titanic'))에서 5개 열을 선택해서 class열을 기준으로 그룹화를 수행한 후 아래와 같이 출력하였다. 다음 사항을 출력하세요.

5개 열 : ['age','sex', 'class', 'fare', 'survived']

- 그룹별 평균 출력
- 그룹별 최대값 출력

In [218]:

```
titanic = sns.load_dataset('titanic')
df3 = titanic.groupby(['class'])
df3
```

Out[218]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001E0E508
65B0>
```

In [ ]:

```

```

In [ ]:

```
```

Q18. titanic 데이터셋에서 'Third'그룹만을 선택해서 group3 이름으로 저장하고 통계 요약표를 출력하세요.

In [24]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
df.head()
```

Out[24]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True

Q19. titanic 데이터셋에서 다음 전처리를 수행하세요.

1. df에서 중복 칼럼으로 고려할 수 있는 컬럼들(6개 내외)을 삭제한 후 나머지 컬럼들로 구성되는 데이터프레임을 df1 이름으로 저장 후 출력하세요.
2. df1에서 null값이 50% 이상인 칼럼을 삭제 후 df2 이름으로 저장하고 출력하세요.
3. df2에서 결측값이 있는 age 칼럼에 대해서 평균값으로 대체 처리를 수행하세요.
4. df2에서 결측값이 있는 embarked 칼럼에 대해서 앞행의 값으로 대체 처리를 수행하세요.
5. df2 문자로 되어있는 칼럼들을 레이블 인코딩 수행하여 숫자로 변환 후 df2.info()를 출력하세요

In [238]:

```
df.head()
```

Out[238]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True

In [232]:

```
df1 = df.drop(columns = ['class','who','adult_male','embark_town','alive'])
df1.head()
```

Out[232]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	deck	alone
0	0	3	male	22.0	1	0	7.2500	S	NaN	False
1	1	1	female	38.0	1	0	71.2833	C	C	False
2	1	3	female	26.0	0	0	7.9250	S	NaN	True
3	1	1	female	35.0	1	0	53.1000	S	C	False
4	0	3	male	35.0	0	0	8.0500	S	NaN	True

In [234]:

```
df1.isnull().sum()
```

Out[234]:

```
survived      0
pclass        0
sex           0
age         177
sibsp         0
parch         0
fare          0
embarked      2
deck        688
alone         0
dtype: int64
```

In [236]:

```
df2 = df1.drop(columns = ['deck'])
df2.head()
```

Out[236]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	alone
0	0	3	male	22.0	1	0	7.2500	S	False
1	1	1	female	38.0	1	0	71.2833	C	False
2	1	3	female	26.0	0	0	7.9250	S	True
3	1	1	female	35.0	1	0	53.1000	S	False
4	0	3	male	35.0	0	0	8.0500	S	True

In [ ]:

```
df2에서 결측값이 있는 age 칼럼에 대해서 평균값으로 대체 처리를 수행하세요.
df2에서 결측값이 있는 embarked 칼럼에 대해서 앞행의 값으로 대체 처리를 수행하세요.
df2 문자로 되어있는 칼럼들을 레이블 인코딩 수행하여 숫자로 변환 후 df2.info()를 출력하세요
```

In [244]:

```
df2['age'].mean()
```

Out[244]:

```
29.69911764705882
```

In [251]:

```
df2.age = fillna(df2['age'].mean())
```

```
---------------------------------------------------------------------
--------
NameError                                 Traceback (most recent call
last)
Input In [251], in <cell line: 1>()
----> 1 df2.age = fillna(df2['age'].mean())

NameError: name 'fillna' is not defined
```

In [ ]:

In [ ]:

In [ ]:

Q20. 보스톤 주택가격 데이터를 탐색한 후 가장 중요한 독립변수 2개를 선정하고 그 이유를 시각화하여 설명하세요.

In [254]:

```
import warnings
warnings.filterwarnings('ignore')

from sklearn.datasets import load_boston
# boston 데이타셋 로드
boston = load_boston()

# boston 데이타셋 DataFrame 변환
bostonDF = pd.DataFrame(boston.data , columns = boston.feature_names)

# boston dataset의 target array는 주택 가격임. 이를 PRICE 컬럼으로 DataFrame에 추가함.
bostonDF['PRICE'] = boston.target
print('Boston 데이타셋 크기 :',bostonDF.shape)
bostonDF.head()
```

```
Boston 데이타셋 크기 : (506, 14)
```

Out[254]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90

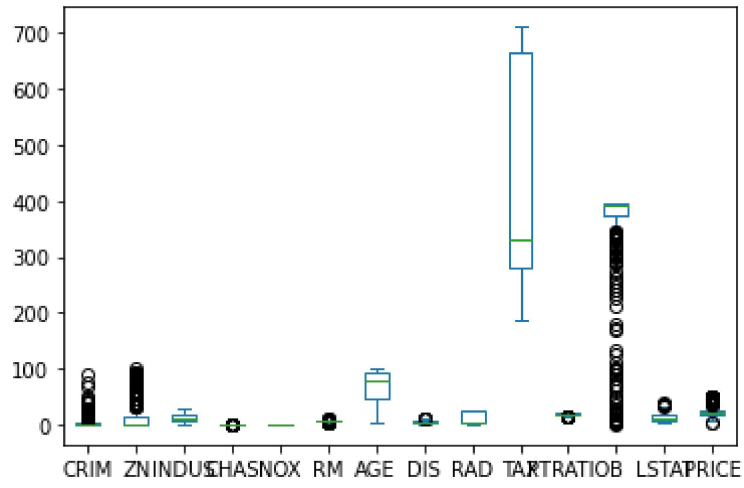
In [465]:

```
import matplotlib.pyplot as plt

bostonDF.plot(kind='box')
```

Out[465]:

```
<AxesSubplot:>
```



In [ ]: