빅데이터 기반 AI 응용 솔루션 개발자 전문과정

교과목명 : Python 분석라이브러리 활용

- 평가일 : 21.7. 23
- 성명 : 김광훈
- 점수 :

Q1. 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 [1]:

```
#1
import numpy as np
ar1 = np.arange(10)
ar2 = np.arange(10).reshape(2,5)
ar3 = np.arange(10).reshape(1,2,5)
print(ar1)
print(ar2)
print(ar3)
```

```
[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]]]
```

Q2. 1 ~ 100 까지 배열에서 3과 7의 공배수인 것만을 출력하세요.

In [2]:

```
#2
import numpy as np
ar = np.arange(1,101)
ar[(ar%3==0) & (ar%7==0)]
```

```
array([21, 42, 63, 84])
```

Q3. 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 [3]:

```
#3
import numpy as np
ar = np.arange(10).reshape(1,2,5)
ar2 = np.arange(10,20).reshape(1,2,5)
ar3 = np.arange(20,30).reshape(1,2,5)

n_ar = ar.reshape(-1)
n_ar2 = ar2.reshape(-1)
n_ar3 = ar3.reshape(-1)

print(n_ar)
print(n_ar2)
print(n_ar3)
```

```
[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 [4]:

```
# ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ
# 눈 나쁜게 죄다 이거시야!
import numpy as np
ar = np.arange(30).reshape(3,2,5)
ar1 = ar.reshape(-1,)
ar1
```

```
array([ 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])
```

Q4. 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 [5]:

```
#4
import numpy as np
arr2d = np.arange(1,10).reshape(3,3)
arr2d
a1 = arr2d[:2,2:].reshape(-1).tolist()
a2 = arr2d[:2,:2].reshape(2,2).tolist()
a3 = arr2d[:2,].reshape(2,3).tolist()

print(a1)
print(a2)
print(a3)
```

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

Q5. 아래 두행렬을 np.arange, reshape를 이용해서 생성 각각 a1, b1으로 저장하고 행렬 내적을 계산한 결과를 출력하세요.

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

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

In [6]:

```
#5
import numpy as np
a1 = np.arange(1,10).reshape(3,3)
b1 = np.arange(10,19).reshape(3,3)
a1.dot(b1)
```

```
array([[ 84,  90,  96],
       [201, 216, 231],
       [318, 342, 366]])
```

Q6. ar의 역행렬을 출력하고 검증을 수행하세요.

In [7]:

```
import numpy as np
np.random.seed(1)
ar = np.random.randint(1,4,size=(3,3))
ar
rar = np.linalg.inv(ar)
rar
check = ar.dot(rar)
check
```

```
array([[ 1.0000000e+00,  0.0000000e+00,  0.0000000e+00],
       [-4.4408921e-16,  1.0000000e+00,  0.0000000e+00],
       [ 0.0000000e+00,  0.0000000e+00,  1.0000000e+00]])
```

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

In [8]:

```
import numpy as np
ar = np.random.randint(10,20,size=(10,5))
np.save('dataset/ar',ar)
L_ar = np.load('dataset/ar.npy')
L_ar
```

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

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 [9]:

```
#8
import seaborn as sns
import pandas as pd
df =sns.load_dataset('titanic')
df.columns
new_columns = ['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone']
df_x = df[new_columns]

df_x.to_pickle('df_x.pkl')
df_x = pd.read_pickle('df_x.pkl')

df_x.head()
```

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

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

In [10]:

```
#9
import pandas as pd
import seaborn as sns
df = sns.load_dataset('titanic')
df['deck'].isnull().sum()
```

```
688
```

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

In [11]:

```
#10
import pandas as pd
import seaborn as sns
print('각 컬럼별 null 개수', df.isnull().sum(),'\n')
print('df 전체 null 개수', df.isnull().sum().sum())
```

```
각 컬럼별 null 개수 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

df 전체 null 개수 869
```

아래 tdf 데이터프레임에서 Q11 ~ Q12 작업을 수행하세요.

In [12]:

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

	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 [13]:

```
#11
import pandas as pd
import seaborn as sns
df = sns.load_dataset('titanic')
tdf = df[['survived','sex','age','class']]
tdf.head()

def age_cat(x):
    cat = ''
    if x <= 5:
        cat = 'Baby'
    elif x <= 12:
        cat = 'Child'
    elif x <= 18:
        cat = 'Teenager'
    elif x <= 25:
        cat = 'Student'
    elif x <= 60:
        cat = 'Adult'
    elif x > 60 :
        cat = 'Elderly'
    return cat
tdf1 = tdf.copy()

tdf1['age_cat'] = tdf['age'].apply(lambda x : age_cat(x))
tdf1[['age_cat']]
tdf1.head()
```

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

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

In [14]:

```
#12
tdf1.info()
tdf1['class'] = tdf1['class'].astype('object')
tdf1['sc'] = tdf1['sex']+'_'+tdf1['class']
tdf1.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   survived  891 non-null    int64
 1   sex       891 non-null    object
 2   age       714 non-null    float64
 3   class     891 non-null    category
 4   age_cat   891 non-null    object
dtypes: category(1), float64(1), int64(1), object(2)
memory usage: 29.0+ KB
```

	survived	sex	age	class	age_cat	sc
0	0	male	22.0	Third	Student	male_Third
1	1	female	38.0	First	Adult	female_First
2	1	female	26.0	Third	Adult	female_Third
3	1	female	35.0	First	Adult	female_First
4	0	male	35.0	Third	Adult	male_Third

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 df4에 저장하고 출력
- df4에서 중복된 칼럼을 삭제한 후 블린 인덱싱을 이용하여 eps가 3000 보다 적거나 stock_name이 이마트인 데이터를 선택하여 데이터프레임을 생성하고 df5 이름으로 저장 및 출력하세요.(단, '<' 와 '==' 를 반드시 사용해야 함)

In [15]:

```
#13
df1 = pd.read_excel('dataset/stock price.xlsx')
df2 = pd.read_excel('dataset/stock valuation.xlsx')
df4 = df1.merge(df2)
display(df4)
df4 = df4.drop('name',axis=1)
df5 = df4[ (df4['eps']< 3000) | (df4['stock_name']=='이마트')]
df5
```

	id	stock_name	value	price	name	eps	bps	per
0	130960	CJ E&M	58540.666667	98900	CJ E&M	6301.333333	54068	15.695091
1	139480	이마트	239230.833333	254500	이마트	18268.166667	295780	13.931338
2	145990	삼양사	82750.000000	82000	삼양사	5741.000000	108090	14.283226
3	185750	종근당	40293.666667	100500	종근당	3990.333333	40684	25.185866
4	204210	모두투어리츠	3093.333333	3475	모두투어리츠	85.166667	5335	40.802348

	id	stock_name	value	price	eps	bps	per	pb
1	139480	이마트	239230.833333	254500	18268.166667	295780	13.931338	0.86043
4	204210	모두투어리츠	3093.333333	3475	85.166667	5335	40.802348	0.65135

```
#13
df1 = pd.read_excel('dataset/stock price.xlsx',index_col='id')
df2 = pd.read_excel('dataset/stock valuation.xlsx',index_col='id')
df4 = df1.join(df2,how='inner')
display(df4)
df4 = df4.drop('name',axis=1)
df5 = df4[ (df4['eps']< 3000) | (df4['stock_name']=='이마트')]
df5
```

Q14. 아래 df에서 drop_duplicates() 메소드를 사용하여 c4, c5열을 기준으로 중복 행을 제거한 후 df3에 저장하고 출력하세요.

In [16]:

```
import pandas as pd
df = pd.DataFrame({'c1':['a', 'a', 'b', 'a', 'b'],
                   'c2':[1, 1, 1, 2, 2],
                   'c3':[1, 1, 2, 2, 2],
                   'c4':[1, 1, 1, 2, 3],
                   'c5':[1, 1, 2, 2, 5]})
print(df)

df3 = df.drop_duplicates(subset=('c4','c5'),keep=False)
df3
```

```
  c1  c2  c3  c4  c5
0  a   1   1   1   1
1  a   1   1   1   1
2  b   1   2   1   2
3  a   2   2   2   2
4  b   2   2   3   5
```

	c1	c2	c3	c4	c5
2	b	1	2	1	2
3	a	2	2	2	2
4	b	2	2	3	5

Q15. 'mpg'를 'kpl' 로 환산하여 새로운 열을 생성하고 처음 3개 행을 소수점 아래 둘째 자리에서 반올림하여 출력하세요.

In [29]:

```
import pandas as pd
auto_df = pd.read_csv('dataset/auto-mpg.csv')

auto_df.columns = ['mpg','cylinders','displacement','horsepower','weight',
                   'acceleration','model year','origin','name']
print(auto_df.head(3))

mpg_to_kpl = 1.60934 / 3.78541

auto_df['kpl'] = auto_df['mpg'] * mpg_to_kpl
auto_df['kpl'] = auto_df['kpl'].round(2)

print(auto_df['kpl'])
```

```
    mpg  cylinders  displacement horsepower  weight  acceleration  model 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
0      7.65
1      6.38
2      7.65
3      6.80
4      7.23
      ...
393    11.48
394    18.71
395    13.60
396    11.90
397    13.18
Name: kpl, Length: 398, dtype: float64
```

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

In [18]:

```
#16
import pandas as pd
import datetime as dt
import datetime
data = pd.read_csv('dataset/stock-data.csv')
data['Date'] = pd.to_datetime(data['Date'])
data['Year'] = data['Date'].dt.year
data['Month'] = data['Date'].dt.month
data['Day'] = data['Date'].dt.day
data = data.set_index('Year')
data.head(1)
```

Year	Date	Close	Start	High	Low	Volume	Month	Day
2018	2018-07-02	10100	10850	10900	10000	137977	7	2

Q17. titanic 데이터셋에서 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 [19]:

```
#17
import pandas as pd
import seaborn as sns
titanic = sns.load_dataset('titanic')
n_titanic = titanic[['age','sex', 'class', 'fare', 'survived']]
mean_grouped = n_titanic.groupby('class').mean()
max_grouped = n_titanic.groupby('class').max()

display('평균',mean_grouped)
display('최대값',max_grouped)
```

'평균'

class	age	fare	survived
First	38.233441	84.154687	0.629630
Second	29.877630	20.662183	0.472826
Third	25.140620	13.675550	0.242363

'최대값'

class	age	sex	fare	survived
First	80.0	male	512.3292	1
Second	70.0	male	73.5000	1
Third	74.0	male	69.5500	1

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

In [20]:

```
#18
grouped = n_titanic.groupby('class')
group3 = grouped.get_group('Third')
display('Third',group3.describe())
```

'Third'

	age	fare	survived
count	355.000000	491.000000	491.000000
mean	25.140620	13.675550	0.242363
std	12.495398	11.778142	0.428949
min	0.420000	0.000000	0.000000
25%	18.000000	7.750000	0.000000
50%	24.000000	8.050000	0.000000
75%	32.000000	15.500000	0.000000
max	74.000000	69.550000	1.000000

Q19. titanic 데이터셋에서 class 열, sex열 기준으로 그룹화한 후 그룹별 평균과 표준편차를 구하세요.

In [21]:

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

#19
mean_grouped = n_titanic.groupby(['class','sex']).mean()
std_grouped = n_titanic.groupby(['class','sex']).std()

display('평균',mean_grouped)
display('표준편차',std_grouped)
```

'평균'

		age	fare	survived
class	sex			
First	female	34.611765	106.125798	0.968085
	male	41.281386	67.226127	0.368852
Second	female	28.722973	21.970121	0.921053
	male	30.740707	19.741782	0.157407
Third	female	21.750000	16.118810	0.500000
	male	26.507589	12.661633	0.135447

'표준편차'

		age	fare	survived
class	sex			
First	female	13.612052	74.259988	0.176716
	male	15.139570	77.548021	0.484484
Second	female	12.872702	10.891796	0.271448
	male	14.793894	14.922235	0.365882
Third	female	12.729964	11.690314	0.501745
	male	12.159514	11.681696	0.342694

Q20. 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 [22]:

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

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

1. df에서 중복 칼럼으로 고려할 수 있는 컬럼들(6개 내외)을 삭제한 후 나머지 컬럼들로 구성되는 데이터프레임을 df1 이름으로 저장 후 출력하세요.

In [32]:

```
# pclass , class / survived, alive / embarked , embark_town
df1 = df.drop(['class','alive','embark_town','who','alone'],axis=1)
df1.head(1)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	adult_male	deck
0	0	3	male	22.0	1	0	7.25	S	True	NaN

2. df1에서 null값이 50% 이상인 칼럼을 삭제 후 df2 이름으로 저장하고 출력하세요.

In [33]:

```
num_null = df1.isnull().sum() /len(df1) #deck column - 0.772166
df2 = df1.drop('deck',axis=1)
df2.head(1)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	adult_male
0	0	3	male	22.0	1	0	7.25	S	True

3. df2에서 결측값이 있는 age 칼럼에 대해서 평균값으로 대체 처리

를 수행하세요.

In [34]:

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

```
177
```

```
0
```

4. df2에서 결측값이 있는 embarked 칼럼에 대해서 앞행의 값으로 대체 처리를 수행하세요.

In [35]:

```
print(df2['embarked'].isnull().sum())
df2['embarked'] = df2['embarked'].fillna(method='ffill')
df2['embarked'].isnull().sum()
```

```
2
```

```
0
```

5. df2 문자로 되어있는 칼럼들을 레이블 인코딩 수행하여 숫자로 변환 후 df2.info()를 출력하세요

In [36]:

```
df2.head(1) # sex / who / adult_male/alone
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
features = ['sex','adult_male']
for feature in features:
    df2[feature] = le.fit_transform(df2[feature])
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   survived    891 non-null    int64
 1   pclass      891 non-null    int64
 2   sex         891 non-null    int32
 3   age         891 non-null    float64
 4   sibsp       891 non-null    int64
 5   parch       891 non-null    int64
 6   fare        891 non-null    float64
 7   embarked    891 non-null    object
 8   adult_male  891 non-null    int64
dtypes: float64(2), int32(1), int64(5), object(1)
memory usage: 59.3+ KB
```

In [ ]: