# ## 프로젝트 기반 빅데이터 서비스 솔루션 개발 전문과정

### #### 교과목명 : 분석라이브러리 활용

- 평가일 : 22.1.21
- 성명 : 오주완
- 점수 : 70

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

In [154]:

```
import numpy as np
ar = np.random.randn(2,3)
print(ar.dtype,ar.shape,ar.ndim)
```

```
float64 (2, 3) 2
```

## 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 [156]:

```
ar = np.arange(10)
ar1 = ar.reshape(-1,)
ar2 = ar.reshape(2,5)
ar3 = ar.reshape(-1,2,5)
print(ar1,'₩n')
print(ar2,'₩n')
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]]]
```

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

In [12]:

```
ar = np.arange(1,101)
ar1 = ar[ar%21==0]
print(ar1)
```

```
[21 42 63 84]
```

## 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 [159]:

```
ar = np.arange(30).reshape(3,2,-1)
ar1 = ar.reshape(-1,)
ar1
```

Out[159]:

```
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])
```

## 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 [161]:

```
arr2d = np.arange(1,10).reshape(3,3)
print(arr2d,'₩n')
print(arr2d[[0,1],2],'₩n')
print(arr2d[0:2,0:2],'₩n')
print(arr2d[0:2,:])
```

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

[3 6] 

[[1 2]
 [4 5]] 

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

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

In [162]:

```
a = np.arange(10).reshape(2,5)
z = np.zeros_like(a)
o = np.ones_like(a)
f = np.full_like(a,3)
print(z,'₩n')
print(o,'₩n')
print(f)
```

```
[[0 0 0 0 0]
 [0 0 0 0 0]] 

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

[[3 3 3 3 3]
 [3 3 3 3 3]]
```

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

In [164]:

```
import pandas as pd
np.random.seed(0)
ar = np.random.randint(10,20,size=(10,5))
ar
np.save('ar_test',ar)
np.load('ar_test.npy')
```

Out[164]:

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

## 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 [167]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
df_x = df.drop('survived',axis=1)
df_x.to_pickle('dataset/df_x.pkl')
df_x = pd.read_pickle('dataset/df_x.pkl')
df_x.head()
```

Out[167]:

| | 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 |

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

In [168]:

```
df = sns.load_dataset('titanic')
print(df['deck'].isnull().sum(axis=0))
```

688

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

In [169]:

```
print(df.isnull().sum())
print()
print(df.isnull().sum(axis=0).sum())
```

```
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

869
```

## 아래 tdf 데이터프레임에서 Q11 ~ Q12 작업을 수행하세요.

In [171]:

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

Out[171]:

| | 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 결

## 과를 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 [176]:

```
import warnings
warnings.filterwarnings('ignore')
def cat_age(age):
    cat=''
    if age <= 5: cat = 'Baby'
    elif age <= 12: cat = 'Child'
    elif age <= 18: cat = 'Teenager'
    elif age <= 25: cat = 'student'
    elif age <= 60: cat = 'Adult'
    else:
        cat = 'Elderly'
    return cat
tdf['age_cat'] = tdf.age.apply(lambda x: cat_age(x))
tdf[['age','age_cat']].head()
```

Out[176]:

| | age | age_cat |
|---|---|---|
| **0** | 22.0 | student |
| **1** | 38.0 | Adult |
| **2** | 26.0 | Adult |
| **3** | 35.0 | Adult |
| **4** | 35.0 | Adult |

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

In [177]:

```
tdf['sc'] = tdf[['sex','class']].agg('_'.join,axis=1)
tdf.head()
```

Out[177]:

| | 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 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 [180]:

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

In [182]:

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

df3.drop('name', axis=1, inplace=True)

df4 = df3[(df3['eps'] < 3000) | (df3['stock_name'] == '이마트')]
df4
```

```
       stock_name           value    price     name           eps      bps  ₩
id
130960      CJ E&M    58540.666667    98900   CJ E&M    6301.333333    54068
139480         이마트   239230.833333   254500      이마트   18268.166667   295780
145990         삼양사    82750.000000    82000      삼양사    5741.000000   108090
185750         종근당    40293.666667   100500      종근당    3990.333333    40684
204210      모두투어리츠     3093.333333     3475   모두투어리츠      85.166667     5335

              per       pbr
id
130960  15.695091  1.829178
139480  13.931338  0.860437
145990  14.283226  0.758627
185750  25.185866  2.470259
204210  40.802348  0.651359
```

Out[182]:

| id | stock_name | value | price | eps | bps | per | pbr |
|---|---|---|---|---|---|---|---|
| **139480** | 이마트 | 239230.833333 | 254500 | 18268.166667 | 295780 | 13.931338 | 0.860437 |
| **204210** | 모두투어리츠 | 3093.333333 | 3475 | 85.166667 | 5335 | 40.802348 | 0.651359 |

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

In [73]:

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

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

In [133]:

```
a1 = np.transpose(a,(1,2,0))
print(a1,a1.shape)
```

```
[[[0]
  [1]
  [2]]

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

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

In [187]:

```
# read_csv() 함수로 df 생성
import pandas as pd
auto_df = pd.read_excel('./dataset/auto-mpg.xlsx')
# 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  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
```

In [188]:

```
mpg_to_kpl = 1.60934 / 3.78541

auto_df['kpl'] = auto_df['mpg'] * mpg_to_kpl
print(auto_df.head(3))
print('₩n')

auto_df['kpl'] = auto_df['kpl'].round(2)
print(auto_df.head(3))
```

```
    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       kpl
0       1  chevrolet chevelle malibu  7.652571
1       1          buick skylark 320  6.377143
2       1         plymouth satellite  7.652571


    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   kpl
0       1  chevrolet chevelle malibu  7.65
1       1          buick skylark 320  6.38
2       1         plymouth satellite  7.65
```

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

In [190]:

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

df['new_Date'] = pd.to_datetime(df['Date'])
df.drop('Date',axis=1,inplace=True)

df['Year'] = df['new_Date'].dt.year
df['Month'] = df['new_Date'].dt.month
df['Day'] = df['new_Date'].dt.day
df.set_index('Year',inplace=True)
df.head()
```

Out[190]:

| Year | Close | Start | High | Low | Volume | new_Date | Month | Day |
|---|---|---|---|---|---|---|---|---|
| **2018** | 10100 | 10850 | 10900 | 10000 | 137977 | 2018-07-02 | 7 | 2 |
| **2018** | 10700 | 10550 | 10900 | 9990 | 170253 | 2018-06-29 | 6 | 29 |
| **2018** | 10400 | 10900 | 10950 | 10150 | 155769 | 2018-06-28 | 6 | 28 |
| **2018** | 10900 | 10800 | 11050 | 10500 | 133548 | 2018-06-27 | 6 | 27 |
| **2018** | 10800 | 10900 | 11000 | 10700 | 63039 | 2018-06-26 | 6 | 26 |

## 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 [191]:

```
titanic = sns.load_dataset('titanic')
df = titanic.loc[:,['age','sex','class','fare','survived']]
grouped = df.groupby(['class'])
a = grouped.mean()
b = grouped.max()
print(a)
print(b)
```

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

## 요약표를 출력하세요.

In [75]:

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

Out[75]:

| | 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 [152]:

```
group3 = grouped.get_group('Third')

group3.describe()
```

Out[152]:

| | 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 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 [25]:

```
df.head()
```

Out[25]:

| | 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 |

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

In [32]:

```
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[32]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | 4 |
| **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 | 9 |
| **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 | 4 |
| **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 | 2 |
| **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 | 5 |

In [ ]:

```

```