```
## 프로젝트 기반 빅데이터 서비스 솔루션 개발 전문과정

#### 교과목명 : 분석라이브러리 활용

- 평가일 : 22.1.21
- 성명 : 정현우
- 점수 : 85
```

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

In [20]:
```
import numpy as np

arr=np.random.randn(6).reshape(2,3)
print(arr.size)
print(arr.dtype)
print(arr.ndim)
```

```
6
float64
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 [25]:
```
arr=np.arange(10) #1차
arr.reshape(2,5) #2차
arr.reshape(1,2,5) #차
```

Out[25]:
```
array([[[0, 1, 2, 3, 4],
        [5, 6, 7, 8, 9]]])
```

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

In [35]:
```
arr=np.arange(1,101).tolist()
for i in arr:
    if i %21 ==0:
        print(i)


```

```
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 [40]:
```
arr= np.arange(0,30).reshape(3,2,5)
arr.reshape(30)
```

Out[40]:
```
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 [56]:
```
arr2d = np.arange(1,10).reshape(3,3)
arr2d[:2,2:3].reshape(2) #1
arr2d[:2,:2]  #2
arr2d[:2,:3]  #3
```

Out[56]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

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

In [122]:
```
import numpy as np
arr=np.arange(0,10).reshape(5,2)
np.zeros_like(arr)
np.ones_like(arr)
np.full_like(arr,5)
```

Out[122]:
```
array([[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 [114]:
```
arr=np.random.random_integers(10,21,100).reshape(2,10,-1)
arr
```

```
C:₩Users₩admin₩AppData₩Local₩Temp/ipykernel_10712/879486828.py:1: DeprecationWarning: This function is deprecated. Please call r
andint(10, 21 + 1) instead
  arr=np.random.random_integers(10,21,100).reshape(2,10,-1)
```

Out[114]:
```
array([[[21, 19, 21, 15, 11],
        [15, 17, 10, 10, 19],
        [16, 15, 17, 18, 20],
        [17, 10, 19, 14, 17],
        [21, 10, 15, 16, 16],
        [18, 18, 17, 10, 10],
        [12, 10, 18, 12, 19],
        [12, 19, 14, 11, 13],
        [16, 20, 13, 15, 10],
        [21, 10, 20, 15, 19]],

       [[12, 16, 18, 12, 17],
        [17, 10, 10, 12, 19],
        [21, 17, 13, 11, 10],
        [12, 11, 10, 10, 11],
        [16, 10, 19, 21, 18],
        [18, 21, 21, 16, 13],
        [11, 16, 18, 17, 14],
        [19, 10, 15, 19, 16],
        [18, 21, 21, 13, 18],
        [18, 15, 13, 17, 15]]])
```

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 [131]:
```
import seaborn as sns
import pandas as pd
df = sns.load_dataset('titanic')
df_x= df.drop(['survived'],axis=1)
np.save('df_x',df_x)
df=np.load('df_x.npy')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~WAppDataWLocalWTemp/ipykernel_9304/1444496141.py in <module>
      4 df_x= df.drop(['survived'],axis=1)
      5 np.save('df_x',df_x)
----> 6 df=np.load('df_x.npy')

~Wanaconda3WlibWsite-packagesWnumpyWlibWnpyio.py in load(file, mmap_mode, allow_pickle, fix_imports, encoding)
    438                 return format.open_memmap(file, mode=mmap_mode)
    439             else:
--> 440                 return format.read_array(fid, allow_pickle=allow_pickle,
    441                                          pickle_kwargs=pickle_kwargs)
    442         else:

~Wanaconda3WlibWsite-packagesWnumpyWlibWformat.py in read_array(fp, allow_pickle, pickle_kwargs)
    737         # The array contained Python objects. We need to unpickle the data.
    738         if not allow_pickle:
--> 739             raise ValueError("Object arrays cannot be loaded when "
    740                              "allow_pickle=False")
```

In [ ]:
```
-1
```

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

In [137]:
```
df = sns.load_dataset('titanic')
df[['deck']].isnull().sum()
```

Out[137]:
```
deck     688
dtype: int64
```

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

In [143]:
```
print(df.isnull().sum().sum())
print(df.isnull().sum())
```

```
869
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
```

아래 tdf 데이터프레임에서 Q11 ~ Q12 작업을 수행하세요.

In [159]:
```
import seaborn as sns
df = sns.load_dataset('titanic')
tdf = df[['survived','sex','age','class']]
tdf.head()
```

Out[159]:

	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 [174]:
```
import seaborn as sns
df = sns.load_dataset('titanic')
tdf = df[['survived','sex','age','class']]

def sol(age):
    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'
    elif age > 60 : cat = 'Elderly'
    else:
        return 0
    return cat

tdf['cat_age']=tdf.age.apply(sol).to_frame()
tdf1=tdf.copy()
tdf1
```

```
C:₩Users₩admin₩AppData₩Local₩Temp/ipykernel_9304/3615140633.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  tdf['cat_age']=tdf.age.apply(sol).to_frame()
```

Out[174]:

	survived	sex	age	class	cat_age
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
...	...	...	...	...	...
886	0	male	27.0	Second	Adult
887	1	female	19.0	First	Student
888	0	female	NaN	Third	0
889	1	male	26.0	First	Adult
890	0	male	32.0	Third	Adult

891 rows × 5 columns

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

male_third

In [220]:
```
slst=tdf1.sex.tolist()
clst=tdf1['class'].tolist()
lst=[]
for i in zip(slst,clst):
    lst.append(f'{i[0]}_{i[1]}')
tdf1['sc']=lst

tdf1
```

Out[220]:

	survived	sex	age	class	cat_age	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
...	...	...	...	...	...	...
886	0	male	27.0	Second	Adult	male_Second
887	1	female	19.0	First	Student	female_First
888	0	female	NaN	Third	0	female_Third
889	1	male	26.0	First	Adult	male_First
890	0	male	32.0	Third	Adult	male_Third

891 rows × 6 columns

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 [189]:
```
df1 = pd.read_excel('stock price.xlsx', index_col='id')
df2 = pd.read_excel('stock valuation.xlsx', index_col='id')

df3=df1.join(df2,how='inner')
df3
# |(df['stock_name']=='이마트')]
df4=df3[(df3.eps<3000)|(df3.stock_name=='이마트')]
df4
```

Out[189]:

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

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

In [190]:
```
a = np.arange(6).reshape(1,2,3)
print(a,a.shape,'₩n')
```

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

In [194]:
```
a.transpose(1,2,0)
```

Out[194]:
```
array([[[0],
        [1],
        [2]],

       [[3],
        [4],
        [5]]])
```

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

In [8]:
```
# read_csv() 함수로 df 생성
import pandas as pd
auto_df = pd.read_csv('auto-mpg.csv')

# 열 이름을 지정
auto_df.columns = ['mpg','cylinders','displacement','horsepower','weight',
                   'acceleration','model year','origin','name']
# print(auto_df.head(3))
df=auto_df.copy()

df['kpl']=round(df.mpg * 0.123,2)
df.head(5)
```

Out[8]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	name	kpl
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu	2.21
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320	1.84
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite	2.21
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst	1.97
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino	2.09

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

In [53]:
```
from datetime import datetime
df = pd.read_csv('stock_data.csv')

df['연월일']=df['연월일'].astype('datetime64').to_frame()
#-2

```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~₩AppData₩Local₩Temp/ipykernel_9304/576457272.py in <module>
      5
      6 for i in df['연월일'].tolist():
----> 7     i.split('-')

AttributeError: 'Timestamp' object has no attribute 'split'
```

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 [248]:
```
import seaborn as sns
df = sns.load_dataset('titanic')
df=df[['age','sex', 'class', 'fare', 'survived']]

grp=df.groupby('class')
grp.describe()
```

Out[248]:

	age								fare					survived				
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%
class																		
First	186.0	38.233441	14.802856	0.92	27.0	37.0	49.0	80.0	216.0	84.154687	...	93.5	512.3292	216.0	0.629630	0.484026	0.0	0.0
Second	173.0	29.877630	14.001077	0.67	23.0	29.0	36.0	70.0	184.0	20.662183	...	26.0	73.5000	184.0	0.472826	0.500623	0.0	0.0
Third	355.0	25.140620	12.495398	0.42	18.0	24.0	32.0	74.0	491.0	13.675550	...	15.5	69.5500	491.0	0.242363	0.428949	0.0	0.0

3 rows × 24 columns

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

In [253]:
```
import seaborn as sns
df = sns.load_dataset('titanic')
df.head()
group3=grp.get_group('Third')
group3.describe()
```

Out[253]:

	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 [255]:
```
df.columns
df1=df[['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked','class']]
df1.head()

# df1.info()
df2=df1


df2.age=df2.age.fillna(df2.age.mean())
df2.head()
# df2.info()
# df2.age.mean()

df2.embarked=df2.embarked.fillna(-1)
#df2.embarked=df2.embarked.fillna(method='ffill') #정답

def sex(x):
    if x=='male':
        return 0
    else:
        return 1
df2.sex=df2['sex'].apply(sex)

def eb(x):
    if x=='S':
        return 0
    elif x=='C':
        return 1
    else:
        return 2

df2.embarked=df2['embarked'].apply(eb)

def cl(x):
    if x=='Third':
        return 3
    elif x=='Second':
        return 2
    else:
        return 1
df2['class']=df2['class'].apply(cl)

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    int64
 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    int64
 8   class     891 non-null    category
dtypes: category(1), float64(2), int64(6)
memory usage: 56.8 KB

C:₩Users₩admin₩anaconda3₩lib₩site-packages₩pandas₩core₩generic.py:5516: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self[name] = value
C:₩Users₩admin₩AppData₩Local₩Temp/ipykernel_9304/3259158386.py:41: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df2['class']=df2['class'].apply(cl)
```

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

In [222]:
```
from sklearn.datasets import load_boston
# boston 데이타셋 로드 -3
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[222]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
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	4.98	24.0
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	9.14	21.6
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	4.03	34.7
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	2.94	33.4
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	5.33	36.2

In [224]:
```
df=bostonDF
df.corr()
```

Out[224]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000
PRICE	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737