

빅데이터 기반 AI 응용 솔루션 개발자 전문과정

교과목명 : Python 분석라이브러리 활용

- 평가일 : 21.7.26
- 성명 : 김광훈
- 점수 :

Q1. arange(), reshape() 이용 1차원 2차원 3차원 배열을 아래와 같이 생성하세요.

1차원

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

2차원

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]]
```

3차원

```
[[[ 1  2  3  4  5]
   [ 6  7  8  9 10]
   [11 12 13 14 15]
   [16 17 18 19 20]]]
```

```
In [1]: import numpy as np
ar = np.arange(1,21)
ar1 = ar.reshape(-1)
ar2 = ar.reshape(4,5)
ar3 = ar.reshape(2,2,5)
print('ar1', ar1, '\n\n' 'ar2', ar2, '\n\n', 'ar3', ar3)
```

```
ar1 [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```

```
ar2 [[ 1  2  3  4  5]
      [ 6  7  8  9 10]
      [11 12 13 14 15]
      [16 17 18 19 20]]
```

```
ar3 [[[ 1  2  3  4  5]
       [ 6  7  8  9 10]]]
```

```
[[11 12 13 14 15]
 [16 17 18 19 20]]]
```

Q2. 1 ~ 100 까지 배열에서 3과 7의 공배수중에서 42의 배수인 것만을 출력하세요.

```
In [2]: import numpy as np
ar = np.arange(1,101)
p_ans= ar[(ar%3 ==0) & (ar%7 == 0 )]
ans = p_ans[p_ans%42==0]
ans

array([42, 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]]]
```

```
In [3]: import numpy as np
ar = np.arange(0,24)
ar_d3 = ar.reshape(2,3,4)
ar_d1 = ar_d3.reshape(-1)
ar_d1

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])
```

Q4. array2d에서 인덱스를 이용해서 값을 선택하고 리스트로 아래와 같이 출력하세요.

```
arr2d = np.arange(11,20).reshape(3,3)

[13 16]

[[11 12]
 [14 15]]

[[11 12 13]
 [14 15 16]]
```

```
In [4]: import numpy as np
arr2d = np.arange(11,20).reshape(3,3)
arr2d

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

```
In [5]: ans1 = arr2d[:2,2:3].reshape(-1).tolist()
ans2 = arr2d[:2,:2].tolist()
ans3 = arr2d[:2].tolist()

print(ans1,ans2,ans3)

[13, 16] [[11, 12], [14, 15]] [[11, 12, 13], [14, 15, 16]]
```

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]]

[[10 11 12 13]
 [14 15 16 17]
 [18 19 20 21]
 [22 23 24 25]
 [26 27 28 29]]
```

```
In [6]: import numpy as np
a1 = np.arange(1,16).reshape(3,5)
b1 = np.arange(10,30).reshape(5,4)
answer = a1.dot(b1)
answer
```

```
array([[ 310,  325,  340,  355],
       [ 760,  800,  840,  880],
       [1210, 1275, 1340, 1405]])
```

Q6. ar의 역행렬을 출력하고 검증을 수행하세요.

지수표현식을 실수 표현으로 변경 : np.set_printoptions(precision=6, suppress=True)

```
In [7]: import numpy as np
np.random.seed(1)
ar = np.random.randint(1,6,size=(5,5))
r_ar = np.linalg.inv(ar)
check = ar.dot(r_ar)
np.set_printoptions(precision=6, suppress=True)
check
```

```
array([[ 1., -0.,  0.,  0., -0.],
       [-0.,  1., -0., -0., -0.],
       [-0.,  0.,  1.,  0., -0.],
       [ 0.,  0.,  0.,  1., -0.],
       [ 0.,  0.,  0.,  0.,  1.]])
```

Q7. 표준정규분포 난수로 10행 5열 2차원 배열 ar을 생성하고 ar_test 이름으로 저장한 후 다시 불러내서 출력하세요.

```
In [8]: ar = np.random.randn(50).reshape(5,10)
np.save('ar_test',ar)
answer = np.load('ar_test.npy')
answer
```

```
array([[ 0.729976,  0.372994,  0.533811, -0.091973,  1.91382 ,  0.330797,
         1.141943, -1.129595, -0.850052,  0.96082 ],
       [-0.217418,  0.158515,  0.873418, -0.111383, -1.038039, -1.00948 ,
        -1.058257,  0.656284, -0.062492, -1.738654],
       [ 0.103163, -0.621667,  0.275718, -1.090675, -0.609985,  0.306412,
         1.691826, -0.747954, -0.580797, -0.110754],
       [ 2.042029,  0.447521,  0.683384,  0.022886,  0.857234,  0.183931,
        -0.416112,  1.25005 ,  1.2483 , -0.757674],
       [ 0.588294,  0.346859,  1.367033,  0.673716, -1.291563, -0.848244,
        -0.1666 ,  0.917196,  0.080251,  0.228239]])
```

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

- 전체 칼럼중 'survived','who','adult_male','deck','embark_town','alive','alone' 칼럼을 제외한 df_x 데이터프레임을 생성한 후 dataset/df_x.pkl로 저장한다.
- df_x.pkl을 데이터프레임 df_x 이름으로 불러온 후 앞 5개 행을 출력한다.

```
In [9]: import seaborn as sns
import pandas as pd
import pickle
df = sns.load_dataset('titanic')
df.columns
df_x = df.drop(['survived', 'who', 'adult_male', 'deck', 'embark_town'])
pd.to_pickle(df_x, 'df_x.pkl')
df_x = pd.read_pickle('df_x.pkl')
df_x.head()
```

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

Q9. Q.8의 df_x에서 각 칼럼별 null 개수를 구하세요.

```
In [10]: import pandas as pd
df_x.isnull().sum()
```

```
pclass      0
sex          0
age        177
sibsp        0
parch        0
fare         0
embarked     2
class        0
dtype: int64
```

Q10. df_x의 전체 null 갯수를 계산하세요.

```
In [11]: import pandas as pd
df_x.isnull().sum().sum()
```

```
179
```

Q11. tdf에서 fare를 3개 카테고리로 구분하는 새로운 칼럼 'fare age'를 생성하여 출력하세요. 단, 카테고리 구분

을 수행하는 사용자 함수를 만들고 그 함수를 'fare' 칼럼에 매핑하여 결과를 tdf에 저장하고 출력하세요.

[카테고리]

fare <= 7.9: cat = 'T'

fare <= 31: cat = 'S'

fare > 31 : cat = 'F'

```
In [12]: import seaborn as sns
import pandas as pd
df = sns.load_dataset('titanic')
tdf = df[['survived', 'sex', 'age', 'class', 'fare']]

def fare_cat(x):
    cat = ''
    if x <= 7.9:
        cat = 'T'
    elif x <= 31:
        cat = 'S'
    elif x > 31:
        cat = 'F'
    return cat

tdf['fare_age'] = tdf['fare'].apply(lambda x : fare_cat(x))
tdf[['fare_age']].head()
```

<ipython-input-12-37c27f185c51>:16: 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['fare_age'] = tdf['fare'].apply(lambda x : fare_cat(x))
```

	fare_age
0	T
1	F
2	S
3	F
4	S

Q12. df의 gender, code 칼럼을 ' '으로 연결한 'gc'칼럼을 추가한 후 출력하세요. 단, 타입변환을 유의해서 수행하세요.

연결 예시 : m_10

```
In [15]: import pandas as pd
df = pd.DataFrame({'gender' : ['m', 'f', 'f'],\
                   'code' : [10, 20, 30]})
# df['code'] = df['code'].astype('string')
df['gc'] = df['gender'] + '_' + df['code'].astype('string')
df[['gc']]
# df['gc'] = df['gender'].agg('_'.join(map(str,df['code'])))
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
---  ------  -
0   gender  3 non-null        object
1   code    3 non-null        int64
2   gc      3 non-null        string
dtypes: int64(1), object(1), string(1)
memory usage: 200.0+ bytes
```

```
In [ ]: df.join()
```

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

- 데이터 셋
 1. 'dataset/stock price.xlsx'
 2. 'dataset/stock valuation.xlsx'
- df1과 df2의 합집합이 출력되도록 결합하여 df4에 저장하고 출력
- df4에서 중복된 칼럼 name 을 삭제한 후 불린 인덱싱을 이용하여 eps가 10000 보다 크고 stock_name이 이마트인 데이터를 선택하여 데이터프레임을 생성하고 df5 이름으로 저장 및 출력하세요.(단, '<' 와 '==' 를 반드시 사용해야 함)

```
In [ ]: import pandas as pd
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')
df4 = df4.drop('name', axis=1)
df5 = df4[ (df4['eps'] > 1000) & (df4['stock_name'] == '이마트') ]

df5.to_excel('df5.xlsx')
df5 = pd.read_excel('df5.xlsx')
df5
```

Q14. df의 score1 과 score2 컬럼을 합한 숫자로 구성된 새로운 컬럼 score3을 생성 후 df를 출력하세요.

```
In [ ]: import numpy as np
import pandas as pd
np.random.seed(0)
df = pd.DataFrame(np.random.randint(123456789, 123456791, size=(2, 2)))
df['score1'] = '123,234,567'
df

import re
def strip_s(x):
    clean = re.sub(',', '', x)
    clean = int(clean)
    return clean

df['score3'] = df['score1'].apply(lambda x : strip_s(x)) + df['score2']
df
```

Q15. auto_df의 horsepower 컬럼에서 숫자가 아닌 원소들을 0으로 대체한 후에 전체 평균을 구한 후 소숫점 2째 자리까지만 출력하세요.

```
In [ ]: import pandas as pd
auto_df = pd.read_csv('dataset/auto-mpg.csv')
auto_df.head()

auto_df['horsepower'].unique()
auto_df['horsepower'] = auto_df['horsepower'].replace('?', 0).astype(int)
auto_df.groupby('horsepower').mean().round(2)
```

Q16. './dataset/stock-data.csv'를 데이터프레임으로 불러와서 Date 컬럼 판다스 Timestamp로 변환한 후 인덱스

로 셋팅하고 Date 컬럼은 삭제한 후 처음 5개 행만 출력하세요.

```
In [ ]: import pandas as pd
import time
import datetime as dt
import datetime
df = pd.read_csv('dataset/stock-data.csv')
df['Timestamp'] = pd.to_datetime(df['Date'])
df = df.set_index('Timestamp')
df = df.drop('Date',axis=1)
df.head()
```

Q17. titanic 데이터셋에서 아래 5개 열을 선택하고 sex 컬럼을 기준으로 그룹화를 수행한 후 다음 사항을 출력하세요.

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

- 그룹별 평균 출력
- 그룹별 최대값 출력

```
In [ ]: import pandas as pd
import seaborn as sns

titanic = sns.load_dataset('titanic')

columns = ['age','sex', 'class', 'fare', 'survived']

tdf = titanic[columns]

grouped_col_sex=tdf.groupby('sex')
display('Mean', grouped_col_sex.mean())
display('MAX',grouped_col_sex.max())

#class encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
features = ['class']
for feature in features:
    tdf['class'] = le.fit_transform(tdf[feature])
N_grouped_col_sex = tdf.groupby('sex')
display('Mean',N_grouped_col_sex.mean())
display('MAX',N_grouped_col_sex.max())
```

Q18. titanic 데이터셋에서 class, age, fare, survived 4개 컬럼만 선택하여 df를 생성한 후 class 컬럼의 'First'그룹

만을 선택해서 group1 이름으로 저장하고 통계요약표를 출력하세요.

```
In [ ]: df = titanic[['class', 'age', 'fare', 'survived']]
grouped_by_class = df.groupby('class')
group1 = grouped_by_class.get_group('First')
group1.describe()
```

Q19. titanic 데이터셋에서 class 열, sex 열 기준으로 그룹화한 후 그룹별 최대값과 최소값을 구하세요.

```
In [ ]: titanic = sns.load_dataset('titanic')
df = titanic[['class', 'sex', 'age', 'fare', 'survived']]

grouped_by_class_sex = df.groupby(['class', 'sex'])
display('MAX', grouped_by_class_sex.max())
display('min', grouped_by_class_sex.min())
```

Q20. titanic 데이터셋에서 다음 전처리를 수행하세요.

1. df에서 'survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked', 'deck' 컬럼만을 선택하여 df1 이름으로 저장 후 출력하세요.
2. df1에서 null값이 50% 이상인 컬럼은 삭제한 후 df2를 생성하고 df2에 null이 있는 컬럼은 적절한 값으로 대체한 후 df2의 null값의 갯수가 0인지 확인하세요.
3. df2에서 문자로 되어있는 컬럼들을 원핫 인코딩 수행하여 숫자로 변환 후 처음 5개 행을 출력하세요

```
In [ ]: import seaborn as sns
df = sns.load_dataset('titanic')
df.head()

# 1
df1 = df[['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'f
df1.head()

#2
df1.isnull().sum() # age/embarked/deck
print((df1['age'].isnull().sum()/len(df1['age']) * 100).round(2))
print((df1['embarked'].isnull().sum()/len(df1['embarked']) * 100).
print((df1['deck'].isnull().sum()/len(df1['deck']) * 100).round(2)

df2 = df1.drop('deck',axis=1)
df2['age'] = df2['age'].fillna(df2['age'].median())
df2['embarked'] = df2['embarked'].fillna(method='ffill')
print('df2 np.nan counts>>', df2.isnull().sum().sum())

#3
df2.info() #sex,embarked << object
dummies = pd.get_dummies(df2[['sex', 'embarked']])
dummies.head()
```

```
In [ ]:
```