CH-3

1. Numpy를 사용하여 배열 A를 만들었다. 다음 중 생성한 배열의 데이터 유형에 관한 정보를 조회하는 명령어는?

```
import numpy as np

X = np.array([[1,2],[3,4]])
```

- 1. np.shape(X)
  2. np.ndim(X)
  3. np.size(X)
  4. X.dtype ❤
  5. np.min(X)

2. 다음 NumPy 배열 `arr`이 주어졌을 때, `arr[arr > 5]`의 결과는 무엇인가요?

```
import numpy as np
arr = np.array([1, 6, 3, 8, 5, 10])
print(arr[arr > 5])
```

- 1. `[6, 8, 10]` ❤
  2. `[1, 3, 5]`
  3. `[1, 6, 3, 8, 5, 10]`
  4. `[True, True, False, True, False, True]`

3. NumPy를 사용하여 다음과 같은 생성하였다. 보기 중 다른 모양의 배열을 생성하는 명령은?

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

```
1) Y = np.identity(3)
2) np.ones_like(A) ❤
3) np.eye(3)
4) np.array([[1.,0.,0.],[0.,1.,0.],[0.,0.,1.]])
5) np.diag((1.,1.,1.))
```

다음과 같이 numpy 이를 이용하여 배열 A를 생성하였다.

```
import numpy as np

A = np.array([[1., 2., 3.],[4., 5., 6.],[7., 8., 9.]])
A
'''
array([[1., 2., 3.],
       [4., 5., 6.],
       [7., 8., 9.]])
'''
```

3. 위의 배열 A에 대하여 그 요소 "2.0"에 접근하는 명령어로 올바른 것은?
    1. A[1]
    2. A.at[1]
    3. A.item(1) ❤
    4. A.iloc[1]
    5. A.slice[1]
4. 위의 배열 A가 있을 때 2번째 열의 모든 요소를 가져오는 명령은 ?
    1. A[1,:]
    2. A[:,2]
    3. A[:,1] ❤
    4. A[2,:]
    5. A[2,2]
5. 위의 배열 A에 대하여 각 **행**의 평균을 계산하는 방법과 결과의 형태(shape)를 올바르게 나타낸 것을 고르시오.
    1. np.mean(A, axis=1), shape: (3,) ❤
    2. np.mean(A, axis=0), shape: (3,1)
    3. np.mean(A, axis=1), shape: (1,3)
    4. np.mean(A, axis=0), shape: (3,3)
    5. np.mean(A, axis=1), shape: (,3)
6. 위 5번 문제의 결과를 아래와 같은 열 벡터로 변환하기 위해 추가해야 하는 코드와 결과의 형태(shape)를 올바르게 나타낸 것을 고르시오.

```
array([[2.],
       [5.],
      [8.]])
```

```
1) .reshape(1,-1), shape: (3,)
2) .reshape(-1,1), shape: (3,1) ❤
3) .reshape(-1,1), shape: (1,3)
4) .reshape(1,-1), shape: (3,3)
5) .reshape(1,-1), shape: (,3)
```

7. NumPy를 사용하여 다음과 같이 배열을 생성하였다. 보기 중 A 배열의 요소 값을 변경 시키는 것은 ?

```
f = lambda m, n: 10*m + n
A = np.fromfunction(f, (6,6), dtype=int)
```

```
1) B = A[1:5, 1:5] ; B[:,:] = 0  ❤
2) B = A[1:5,1:5].copy() ; B[:,:] = 0
3) B = A[[0,2,4]] ; B[:,:] = 0
4) B = A[np.array([0,2,4])] ; B[:,:] = 0
5) B = A[A<24].reshape(4,4) ; B[:,:] = 0
```

8. NumPy의 다음 명령 중 설명이 잘못된 것을 고르시오.
    1. np.concatenate # 지정한 축을 따라 배열을 이어 붙여서 새로운 배열을 만든다.
    2. np.transpose # 배열을 전치한다.
    3. np.where # 지정한 위치에 원소를 추가한다.#np.insert ❤
    4. np.delete # 지정한 위치의 원소를 삭제한다.
    5. np.sort # 지정한 축을 따라 배열 정렬
9. 다음과 같이 Pandas 를 이용하여 filename.csv 파일을 읽어 들였다. 다음 중 명령의 설명이 잘못된 것은

```
import pandas as pd
df = pd.read_csv("filename.csv")
```

- 1. df.dtypes # 컬럼별 데이터 유형을 확인한다.
  2. df.shape # 읽어들인 df DataFrame 의 열과 행의 정보를 출력한다.

3. df.isnull().sum() # df DataFrame에 존재하는 Null 값의 개수 구한다.
4. df.info() # df의 기초 통계량 정보를 제공한다. # df.describe() ❤
5. df.head(10) # 행의 앞에서 부터 10번째 행까지를 출력한다.

10. 다음과 같이 seaborn 라이브러리의 기본 데이터 중 'titanic' 데이터를 읽어들이고 보기의 명령을 실행하였다. 명령에 대한 설명이 잘못된 것은

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

- 1. titanic["age"].shape # titanic DataFrame의 "age"컬럼의 데이터 모양을 확인한다.
  2. titanic["age"] > 35 # titanic DataFrame의 "age"컬럼의 35 보다 큰 요소에 대해서 True, 이하의 요소에 False 를 출력한다.
  3. above_35 = titanic[titanic["age"] > 35] # titanic DataFrame의 "age"컬럼의 35 보다 큰 요소에 대해서 above_34 라는 새로운 DataFrame 변수에 입력한다.
  4. age_no_na = titanic[titanic["age"].notna()] # titanic DataFrame의 "age"컬럼의 NA (값이 빈 요소)를 포함한 행을 제하고 새로운 DataFrame 변수 age_no_na에 넣는다.
  5. class_23 = titanic[titanic["pclass"].isin([2, 3])] # titanic DataFrame의 "pclass"컬럼에 요소들 중 2, 3 을 제외한 행만 class_23 DataFrame 에 담는다. ❤

11. 다음 Pandas 명령 코드에 대해서 옳게 설명한 것은 무엇인가

```
titanic.groupby(["sex", "pclass"])["fare"].mean()
```

- 1. titanic DataFrame의 "sex", "pclass" 컬럼의 평균을 구한다.
  2. titanic DataFrame의 "fare" 컬럼을 사용하여 "sex","pclass" 컬럼의 평균을 구한다.
  3. titanic DataFrame의 "sex","pclass" 컬럼이 가진 구릅별로 구분하여 "fare" 컬럼의 평균을 구한다. ❤
  4. titanic DataFrame의 "fare" 컬럼의 평균을 구하여 "sex","pclass" 컬럼에 입력한다.
  5. titanic DataFrame의 "sex","pclass" 컬럼의 평균을 구하여 "fare" 컬럼에 입력한다.

12. 다음 Pandas 명령을 포함한 코드에 대해서 옳게 설명한 것은 무엇인가?

```
# titanic 'age' 컬럼에 null 값 총수 확인
print(titanic['age'].isnull().sum())
# null 값을 해당 컬럼의 성별 평균 값으로 결측치 채우기
titanic['age'].fillna(titanic_df['age'].mean(), inplace=True)
print(titanic['age'].isnull().sum())

def range_age(age):
    age = int(age)
    if age >= 70:e
        return 'Old'
    elif age >= 10:
        return str(age//10) + 'age range'
    else:
        return 'baby'

titanic['age_range'] = titanic['age'].apply(range_age)
```

- 1. range_age 함수를 titanic DataFrame 전체에 적용하고 있다.
  2. range_age 함수를 적용하기 전에 titanic DataFrame의 Null 값에 대한 조회만을 징행하였다.
  3. titanic DataFrame의 "age" 컬럼의 평균값을 모든 "age" 컬럼에 적용하였다.
  4. titanic DataFrame의 "age" 컬럼의 평균값을 "age" 컬럼에 빈값에 적용하고 apply()를 사용하여 range_age 함수를 "age" 컬럼에 적용한 결과를 'age_range' 컬럼를 생성하여 입력하였다. ❤
  5. titanic DataFrame에 Null 값에 대한 사전 전처리를 하지 않아도 apply()를 사용하여 range_age 함수하였을 때 error가 발생하지 않는다.

다음과 같이 matplotlib 라이브러리를 import하고 생성한 x와 y변수들의 그래프를 그리는 코드를 만들고 있다.

```
import matplotlib.pyplot as plt
import numpy as np

# 데이터 준비
x= np.linspace(-5,2, 100)

y1 = x**3 + 5*x**2 + 10
y2 = 3*x**2 + 10*x
y3 = 6*x + 10
y4 = x

fig, ax = plt.subplots(figsize=(8,2.5), facecolor="skyblue")
```

```
ax.plot(x, y1, color="blue", linestyle='-', label="y1(x)")
ax.plot(x, y2, color="red", linestyle='-.', label="y2(x)")
ax.plot(x, y3, color="green", linestyle=':', label="y3(x)")
ax.plot(x, y4, color="black", linestyle='--', label="y3(x)")
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.grid(color="blue", which="both",ls=':')
ax.legend(loc=0) # loc 래전트 위치
ax.set_title("Example")
```

13. 위의 코드에 대한 설명이다 잘못된 것을 고르시오.

1) facecolor="skyblue" , 생성되는 figure의 색을 지정한다.
2) plt.subplots 은 figure와 axis 객체를 동시에 생성한다.
3) plot 메서드를 통해서 생성된 Axes에 데이터를 그려 넣을 수 있다.
4) plot 메서드 안의 linestyle 또는 ls 옵션은 선의 두께를 지정 할 수 있다. ❤
5) legend 메서드를 aixs에 적용하여 범례를 지정하고 loc 옵션으로 위치를 변경할 수 있다.

14. 위의 코드에 대한 설명이다 잘못된 것을 고르시오.

1) fig, ax = plt.subplots(2,2,figsize=(8,2.5), facecolor="skyblue")와 같이 2,2 로 추가 인자를 줄 경우 ax 는 총 subplot 4개 Axes를 가진 객체가 된다.
2) subplot을 생성할 경우 figure 인 fig 객체의 크기는 figsize=(8,2.5) 상관 없이 자동으로 커진다.❤
3) 기존 하나의 Axes 인 ax에 4개의 그래프 선을 그린 것과는 다르게 보기 1)과 같이 Axes를 4개 생성하였다면, ax[0,1].plot(x, y2, color="red", linestyle='-.', label="y2(x)")와 같이 각 ax의 위치 인덱스로 Axes에 데이터를 지정하여야 한다.
4) 각 subplot의 Axes에 각기 다른 4개의 y 변수들을 각각 지정해 그래프를 그릴 수 있다.
5) 이때 각 subplot 에 대하여 title, label, grid, legend 를 각각 Axes location 인덱스에 따라 지정해 주어야 한다.

15. 다음은 matplotlib의 pyplot 을 사용하여 그래프를 그리기 위한 figure 객체를 만들고 ax를 추가하는 코드이다. 다음 설명 중 옳지 않은 것은 무엇인가?

```
#figure 객체생성
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(4,2), facecolor="skyblue") # hex code 가능 "#f1f1f1"
#실제 그래프나 그림이 들어 가는 축을 제공하는 명령은 다음과 같다.
```

```
ax_01 = fig.add_axes((0.1,0.1,0.5,0.5), facecolor="blue")
ax_02 = fig.add_axes((0.3,0.3,0.5,0.5), facecolor="red")
plt.show()
```

1. plt.figure()는 그래프나 그림을 넣기 위한 fig 객체를 생성한다.
2. plt.figure()의 figsize 는 figure 크기를 가로와 세로로 지정한다.
3. plt.figure()의 facecolor 는 figure의 색을 지정한다.
4. .add_axes()는 fig 객체가 생성한 공간에 ax 축을 생성한다.
5. 위 코드에서 ax_01로 지정한 좌표 축의 그래프는 ax_02로 지정한 좌표 축 그래프 위에 위치하게 된다. ❤ (추가하는 axes 순서대로 위에 중첩된다.)