apply 메소드 활용

- 간단한 함수 만들기
- apply 메소드 사용하기 – 기초
- apply 메소드 사용하기 – 고급

■ 간단한 함수 만들기

● 제곱 함수와 n 제곱 함수 만들기

- 제곱 사용자 함수

```
def my_sq(x):
    return x ** 2
```

- n 제곱 사용자 함수

```
def my_exp(x, n):
    return x ** n
```

- 사용자 함수 사용

```
print(my_sq(4))
print(my_exp(3, 4))
```

```
16
81
```

■ apply 메소드 사용하기 – 기초

● 데이터프레임 준비

```
import pandas as pd

df = pd.DataFrame({'a': [10, 20, 30], 'b': [20, 30, 40]})

print(df)
```

```
    a   b
0  10  20
1  20  30
2  30  40
```

■ apply 메소드 사용하기 – 기초

● 시리즈에 apply 메소드 사용

- 제곱 함수 (my_sq) 적용 : 인자 1개

```
sq = df['a'].apply(my_sq)
print(sq)
```

```
0     100
1     400
2     900
Name: a, dtype: int64
```

```
def my_sq(x):
    return x ** 2
```

	a	b
0	10	20
1	20	30
2	30	40

■ apply 메소드 사용하기 – 기초

● 시리즈에 apply 메소드 사용

- n 제곱 함수 (my_exp) 적용 : 인자 2개

```
ex = df['a'].apply(my_exp, n=2)
print(ex)
```

```
0     100
1     400
2     900
Name: a, dtype: int64
```

```
def my_exp(x, n):
    return x ** n
```

	a	b
0	10	20
1	20	30
2	30	40

■ apply 메소드 사용하기 – 기초

● 데이터프레임에 apply 메소드 사용

- 사용자 함수 준비

```
def print_me(x):
    print(x)
```

- 출력 함수 (print_me) 적용

	a	b
0	10	20
1	20	30
2	30	40

열 방향 (↓)

```
df.apply(print_me, axis=0)
```

```
0    10
1    20
2    30
Name: a, dtype: int64
0    20
1    30
2    40
Name: b, dtype: int64
a    None
b    None
dtype: object
```

행 방향 (→)

```
df.apply(print_me, axis=1)
```

```
a    10
b    20
Name: 0, dtype: int64
a    20
b    30
Name: 1, dtype: int64
a    30
b    40
Name: 2, dtype: int64
0    None
1    None
2    None
dtype: object
```

■ apply 메소드 사용하기 – 기초

● 데이터프레임에 apply 메소드 사용

- 데이터 3개를 입력받아 평균을 계산하는 사용자 함수

```
def avg_3(x, y, z):
    return (x + y + z) / 3
```

```
print(df.apply(avg_3, y=1, z=2))
```

```
              a           b
0   4.333333   7.666667
1   7.666667  11.000000
2  11.000000  14.333333
```

```
def avg_3_apply(col):
    x = col[0]
    y = col[1]
    z = col[2]
    return (x + y + z) / 3
```

```
print(df.apply(avg_3_apply))
```

```
a    20.0
b    30.0
dtype: float64
```

■ apply 메소드 사용하기 – 고급

- 데이터프레임의 누락값을 처리한 다음 apply 메소드 사용하기
  - 타이타닉호 침몰 시 생존자 데이터 가져오기

```
import seaborn as sns
titanic = sns.load_dataset("titanic")
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.6+ KB
```

■ apply 메소드 사용하기 – 고급

● 데이터프레임의 누락값을 처리한 다음 apply 메소드 사용하기

- 누락값의 개수를 확인하는 함수 작성 및 적용

```
import numpy as np

def count_missing(vec):
    null_vec = pd.isnull(vec)
    null_count = np.sum(null_vec)
    return null_count

cmis_col = titanic.apply(count_missing)
print(cmis_col)
```

```
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
```

■ apply 메소드 사용하기 – 고급

● 데이터프레임의 누락값을 처리한 다음 apply 메소드 사용하기

- 누락값의 비율을 계산하는 함수 작성 및 적용

```
def prop_missing(vec):
    num = count_missing(vec)
    dem = vec.size
    return num / dem

pmis_col = titanic.apply(prop_missing)
print(pmis_col)
```

```
survived       0.000000
pclass         0.000000
sex            0.000000
age            0.198653
sibsp          0.000000
parch          0.000000
fare           0.000000
embarked       0.002245
class          0.000000
who            0.000000
adult_male     0.000000
deck           0.772166
embark_town    0.002245
alive          0.000000
alone          0.000000
dtype: float64
```

■ apply 메소드 사용하기 – 고급

● 각 행의 누락값 개수를 확인하여 데이터 추가

- apply 메소드 행 방향으로 적용

```
titanic['num_missing'] = titanic.apply(count_missing, axis=1)

titanic.iloc[:5, list(range(11)) + [-1]]
```

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

■ apply 메소드 사용하기 – 고급

● 각 행의 누락값 개수를 확인하여 데이터 추가

- 조건식 적용 및 sample 추출

```
print(titanic.loc[titanic.num_missing > 1, ].sample(10))
```

```
     survived  pclass     sex  age  sibsp  parch      fare embarked   class  \
547         1       2    male  NaN      0      0   13.8625        C  Second
589         0       3    male  NaN      0      0    8.0500        S   Third
301         1       3    male  NaN      2      0   23.2500        Q   Third
324         0       3    male  NaN      8      2   69.5500        S   Third
552         0       3    male  NaN      0      0    7.8292        Q   Third
87          0       3    male  NaN      0      0    8.0500        S   Third
415         0       3  female  NaN      0      0    8.0500        S   Third
560         0       3    male  NaN      0      0    7.7500        Q   Third
334         1       1  female  NaN      1      0  133.6500        S   First
410         0       3    male  NaN      0      0    7.8958        S   Third

       who  adult_male deck  embark_town alive  alone  num_missing
547    man        True  NaN    Cherbourg   yes   True            2
589    man        True  NaN  Southampton    no   True            2
301    man        True  NaN   Queenstown   yes  False            2
324    man        True  NaN  Southampton    no  False            2
552    man        True  NaN   Queenstown    no   True            2
87     man        True  NaN  Southampton    no   True            2
415  woman       False  NaN  Southampton    no   True            2
560    man        True  NaN   Queenstown    no   True            2
334  woman       False  NaN  Southampton   yes  False            2
410    man        True  NaN  Southampton    no   True            2
```