05 판다스를 활용한 데이터 이해

학습 내용

- 판다스를 이해하고 실습을 통해 알아본다.

01 파이썬 기본 다지기

리스트

In [1]:

```
myfood = ['banana', 'apple', 'candy']
print(myfood[0])
print(myfood[1])
print(myfood[2])
print(myfood[1:3])  # 두번째 세번째 가져오기
```

```
banana
apple
candy
['apple', 'candy']
```

In [2]:

```
for item in myfood:
  print(item)
```

```
banana
apple
candy
```

딕셔너리(Dictionary)

In [3]:

```
dict1 = {'one':'하나', 'two':"둘", 'three':'셋'}
dict2 = {1:"하나", 2:"둘", 3:"셋"}
dict3 = {'col1':[1,2,3], 'col2':['a','b','c']}
```

In [4]:

```
print(dict1)
print(dict2)
print(dict3)
```

```
{'one': '하나', 'two': '둘', 'three': '셋'}
{1: '하나', 2: '둘', 3: '셋'}
{'col1': [1, 2, 3], 'col2': ['a', 'b', 'c']}
```

In [5]:

```
print(dict1['one'])
print(dict2[2])
print(dict3['col2'])
```

```
하나
둘
['a', 'b', 'c']
```

판다스 모듈 불러오기

In [6]:

```
import pandas as pd   # pandas 를 불러오고 밑에서 이를 pd 약자로서 쓰겠다.
```

In [7]:

```
# pandas안의 Series와 DataFrame를 불러옴.
from pandas import Series, DataFrame
```

In [8]:

```
print("pandas 버전 : ", pd.__version__)
```

```
pandas 버전 :  1.1.3
```

홍길동 팀별 대항 게임 5일간의 점수

[1000, 14000, 3000, 3000, 1000]

In [9]:

```
score = Series( [1000, 14000, 3000, 3000, 1000] )
print(score)
print("자료형 확인 : ", type(score))
```

```
0     1000
1    14000
2     3000
3     3000
4     1000
dtype: int64
자료형 확인 :  <class 'pandas.core.series.Series'>
```

In [10]:

```
## Series 인덱스 확인
print(score.index)

# 인덱스를 리스트 자료형으로 변경 후, 확인하기
print(list(score.index))

## Series 값 확인
print(score.values)

## Series 값 자료형 확인
print(score.dtype)
```

```
RangeIndex(start=0, stop=5, step=1)
[0, 1, 2, 3, 4]
[ 1000 14000  3000  3000  1000]
int64
```

판다스 시리즈 인덱스 지정

In [11]:

```
### 인덱스(index) 속성 이용
score = Series( [1000, 14000, 3000],
               index = ['2019-05-01', '2019-05-02', '2019-05-03'] )

print(score)
```

```
2019-05-01     1000
2019-05-02    14000
2019-05-03     3000
dtype: int64
```

In [12]:

```
print(score['2019-05-01'])   # 인덱스 이용 - 5월 1일 날짜 점수 확인
print("-------------")
print(score['2019-05-02':'2019-05-03']) # 5월 2일, 3일 날짜 팀 점수 확인
```

```
1000
-------------
2019-05-02    14000
2019-05-03     3000
dtype: int64
```

In [13]:

```
for idx in score.index:
  print(idx)
```

```
2019-05-01
2019-05-02
2019-05-03
```

In [14]:

```
for value in score.values:
  print(value)
```

```
1000
14000
3000
```

날짜값 인덱스 지정

In [17]:

```
dict_dat = { 'one':[10,20,30,40],
             'two':[100,200,300,400],
             'three':[1000,2000,3000,4000]}

date = ['2022-05-01', '2022-05-02', '2022-05-03', '2022-05-05']
df = pd.DataFrame(dict_dat, index=date)
df
```

Out[17]:

	one	two	three
2022-05-01	10	100	1000
2022-05-02	20	200	2000
2022-05-03	30	300	3000
2022-05-05	40	400	4000

In [18]:

```
one_idx = list( range(10,50,10) )
two_idx = list( range(100,500,100) )
three_idx = list( range(1000,5000,1000) )

dict_dat = { 'one':[10,20,30, 40],
             'two':[100,200,300, 400],
             'three':[1000,2000,3000, 4000]}

date = pd.date_range('2022-05-01', '2022-05-04')
df = pd.DataFrame(dict_dat, index=date)
df
```

Out[18]:

	one	two	three
2022-05-01	10	100	1000
2022-05-02	20	200	2000
2022-05-03	30	300	3000
2022-05-04	40	400	4000

두 팀의 팀점수 합산해보기

- 길동팀의 3일간의 점수와 toto 팀의 3일간의 점수

In [19]:

```
from pandas import Series
```

In [21]:

```
gildong = Series([1500, 3000, 2500],
                 index = ['2022-05-01', '2022-05-02', '2022-05-03'] )
toto = Series([3000, 3000, 2000],
              index = ['2022-05-01', '2022-05-03', '2022-05-02'] )
```

In [22]:

```
gildong + toto
```

Out[22]:

```
2022-05-01    4500
2022-05-02    5000
2022-05-03    5500
dtype: int64
```

02 데이터 프레임의 이해

- 데이터 프레임의 객체를 생성하는 가장 간단한 방법은 딕셔너리를 이용하는 방법
- 데이터 프레임은 Series의 결합으로 이루어진 것으로 생각할 수 있음.
- Pandas(판다스)의 대표적인 기본 자료형이다.
- DataFrame 함수를 이용하여 객체 생성이 가능하다.

In [23]:

```
from pandas import DataFrame
```

In [24]:

```
dat = { 'col1' : [1,2,3,4],
        'col2' : [10,20,30,40],
        'col3' : ['A', 'B', 'C', 'D'] }
df = DataFrame(dat)
df
```

Out[24]:

	col1	col2	col3
0	1	10	A
1	2	20	B
2	3	30	C
3	4	40	D

네 팀의 5일간의 팀별 점수

- 팀은 toto, gildong, apple, catanddog 팀이다.

In [25]:

```
from pandas import DataFrame

team_score = { "toto":[1500,3000,5000,7000,5500],
               "apple":[4000,5000,6000,5500,4500],
               "gildong":[2000,2500,3000,4000,3000],
               "catanddog":[7000,5000,3000,5000,4000]}

team_df = DataFrame(team_score)
team_df
```

Out[25]:

	toto	apple	gildong	catanddog
0	1500	4000	2000	7000
1	3000	5000	2500	5000
2	5000	6000	3000	3000
3	7000	5500	4000	5000
4	5500	4500	3000	4000

In [26]:

```
date = ['22-05-01','22-05-02', '22-05-03', '22-05-04', '22-05-05']
team_df = DataFrame(team_score,
                    columns=['catanddog', 'toto', 'apple', 'gildong'],
                    index=date)
team_df
```

Out[26]:

	catanddog	toto	apple	gildong
22-05-01	7000	1500	4000	2000
22-05-02	5000	3000	5000	2500
22-05-03	3000	5000	6000	3000
22-05-04	5000	7000	5500	4000
22-05-05	4000	5500	4500	3000

toto팀의 날짜별 점수를 확인해 보자.

- 팀별 컬럼명을 이용하여 접근이 가능하다.

In [27]:

```
team_df['toto']
```

Out[27]:

```
22-05-01    1500
22-05-02    3000
22-05-03    5000
22-05-04    7000
22-05-05    5500
Name: toto, dtype: int64
```

- toto와 gildong 팀 확인

In [28]:

```
team_df[ ['toto', 'gildong'] ]
```

Out[28]:

	toto	gildong
22-05-01	1500	2000
22-05-02	3000	2500
22-05-03	5000	3000
22-05-04	7000	4000
22-05-05	5500	3000

loc와 iloc를 이용한 접근

- loc는 데이터 프레임의 컬럼명(인덱스)를 사용하여 데이터 추출한다.
- iloc는 데이터 프레임의 데이터 순서(번호)를 사용하여 데이터 추출(시작번호 : 0)
- loc[행, 열] 접근이라고 쉽게 생각한다.

In [29]:

```
print(team_df.loc[ '22-05-02' ] ) # 22-05-02 일
print("-----------")
print(team_df.loc[ ['22-05-02', '22-05-03'] ]) # 5월 2일, 3일
print("-----------")
print(team_df.loc[ '22-05-02': ])  # 5월 2일 이후 전체 데이터 가져오기
```

```
catanddog    5000
toto         3000
apple        5000
gildong      2500
Name: 22-05-02, dtype: int64
-----------
          catanddog  toto  apple  gildong
22-05-02       5000  3000   5000     2500
22-05-03       3000  5000   6000     3000
-----------
          catanddog  toto  apple  gildong
22-05-02       5000  3000   5000     2500
22-05-03       3000  5000   6000     3000
22-05-04       5000  7000   5500     4000
22-05-05       4000  5500   4500     3000
```

loc를 이용한 열에 접근

In [30]:

```
## 컬럼명 확인
print(team_df.columns)
print("-----")
print(team_df.loc[:, 'toto'])   # 전체행, toto팀
print("-----")
print(team_df.loc[:, ['toto', 'gildong'] ])   # 전체행, toto, gildong팀
print("-----")
print(team_df.loc[:, 'toto': ])   # 전체행, toto 부터 끝까지
```

```
Index(['catanddog', 'toto', 'apple', 'gildong'], dtype='object')
-----
22-05-01    1500
22-05-02    3000
22-05-03    5000
22-05-04    7000
22-05-05    5500
Name: toto, dtype: int64
-----
            toto  gildong
22-05-01  1500      2000
22-05-02  3000      2500
22-05-03  5000      3000
22-05-04  7000      4000
22-05-05  5500      3000
-----
            toto  apple  gildong
22-05-01  1500   4000      2000
22-05-02  3000   5000      2500
22-05-03  5000   6000      3000
22-05-04  7000   5500      4000
22-05-05  5500   4500      3000
```

iloc 속성을 이용한 행, 열 데이터 접근하기

In [31]:

```
print(team_df.iloc[0])      # 첫번째 행 접근
print("------")
print(team_df.iloc[ [0,1] ])  # 첫번째 두번째 행 접근
print("------")
print(team_df.iloc[ 0:3:1] )  # 첫번째부터 세번째 행 접근
print("------")
range_num = list(range(0,3,1))
print(team_df.iloc[ range_num ] )  # 첫번째부터 세번째 행 접근
```

```
catanddog    7000
toto         1500
apple        4000
gildong      2000
Name: 22-05-01, dtype: int64
------
          catanddog  toto  apple  gildong
22-05-01       7000  1500   4000     2000
22-05-02       5000  3000   5000     2500
------
          catanddog  toto  apple  gildong
22-05-01       7000  1500   4000     2000
22-05-02       5000  3000   5000     2500
22-05-03       3000  5000   6000     3000
------
          catanddog  toto  apple  gildong
22-05-01       7000  1500   4000     2000
22-05-02       5000  3000   5000     2500
22-05-03       3000  5000   6000     3000
```

In [32]:

```
print(team_df.iloc[:, 0])      # 첫번째 열 접근
print("------")
print(team_df.iloc[:, [0,1] ])  # 첫번째 두번째 열 접근
print("------")
print(team_df.iloc[:, 0:3:1] )  # 첫번째부터 세번째 열 접근
print("------")
range_num = list(range(0,3,1))
print(team_df.iloc[:, range_num ] )  # 첫번째부터 세번째 열 접근
```

```
22-05-01    7000
22-05-02    5000
22-05-03    3000
22-05-04    5000
22-05-05    4000
Name: catanddog, dtype: int64
------
            catanddog  toto
22-05-01         7000  1500
22-05-02         5000  3000
22-05-03         3000  5000
22-05-04         5000  7000
22-05-05         4000  5500
------
            catanddog  toto  apple
22-05-01         7000  1500   4000
22-05-02         5000  3000   5000
22-05-03         3000  5000   6000
22-05-04         5000  7000   5500
22-05-05         4000  5500   4500
------
            catanddog  toto  apple
22-05-01         7000  1500   4000
22-05-02         5000  3000   5000
22-05-03         3000  5000   6000
22-05-04         5000  7000   5500
22-05-05         4000  5500   4500
```

팀별 총합 및 평균 등의 통계는 얼마나 될까?

In [33]:

```
print(team_df.sum() )
print("----")
print(team_df.mean() )
print("----")
```

```
catanddog    24000
toto         22000
apple        25000
gildong      14500
dtype: int64
----
catanddog    4800.0
toto         4400.0
apple        5000.0
gildong      2900.0
dtype: float64
----
```

팀별 요약값을 보고 싶다.

In [34]:

```
team_df.describe()
```

Out[34]:

	catanddog	toto	apple	gildong
count	5.000000	5.000000	5.000000	5.000000
mean	4800.000000	4400.000000	5000.000000	2900.000000
std	1483.239697	2162.174831	790.569415	741.619849
min	3000.000000	1500.000000	4000.000000	2000.000000
25%	4000.000000	3000.000000	4500.000000	2500.000000
50%	5000.000000	5000.000000	5000.000000	3000.000000
75%	5000.000000	5500.000000	5500.000000	3000.000000
max	7000.000000	7000.000000	6000.000000	4000.000000

In [35]:

```
## 날짜별 누적 통계
team_df.cumsum()
```

Out[35]:

	catanddog	toto	apple	gildong
22-05-01	7000	1500	4000	2000
22-05-02	12000	4500	9000	4500
22-05-03	15000	9500	15000	7500
22-05-04	20000	16500	20500	11500
22-05-05	24000	22000	25000	14500

In [36]:

```
## 날짜별 합계
print(team_df.sum(axis=1))
```

```
22-05-01    14500
22-05-02    15500
22-05-03    17000
22-05-04    21500
22-05-05    17000
dtype: int64
```

In [37]:

```
rowsum = team_df.sum(axis=1)
print(type(rowsum))
```

```
<class 'pandas.core.series.Series'>
```

In [38]:

```
team_df['rowsum'] = team_df.sum(axis=1)
team_df
```

Out[38]:

	catanddog	toto	apple	gildong	rowsum
22-05-01	7000	1500	4000	2000	14500
22-05-02	5000	3000	5000	2500	15500
22-05-03	3000	5000	6000	3000	17000
22-05-04	5000	7000	5500	4000	21500
22-05-05	4000	5500	4500	3000	17000

점수가 높은 날짜별로 확인해 보자.

In [39]:

```
team_df.rowsum.sort_values(ascending=False)
```

Out[39]:

```
22-05-04    21500
22-05-05    17000
22-05-03    17000
22-05-02    15500
22-05-01    14500
Name: rowsum, dtype: int64
```

조건을 걸어 일정 이상의 팀점수의 날만 확인해 보자.

- 17000이상인 날만 확인해 보기

In [40]:

```
team_df[ team_df.rowsum >= 17000]
```

Out[40]:

	catanddog	toto	apple	gildong	rowsum
22-05-03	3000	5000	6000	3000	17000
22-05-04	5000	7000	5500	4000	21500
22-05-05	4000	5500	4500	3000	17000

In [41]:

```
team_df
```

Out[41]:

	catanddog	toto	apple	gildong	rowsum
22-05-01	7000	1500	4000	2000	14500
22-05-02	5000	3000	5000	2500	15500
22-05-03	3000	5000	6000	3000	17000
22-05-04	5000	7000	5500	4000	21500
22-05-05	4000	5500	4500	3000	17000

합계 점수가 1등 2등만 선택해 보자.

In [42]:

```
team_df.sum()
```

Out[42]:

```
catanddog    24000
toto         22000
apple        25000
gildong      14500
rowsum       85500
dtype: int64
```

In [43]:

```
team_df.drop(['toto', 'gildong' ], axis=1)
```

Out[43]:

	catanddog	apple	rowsum
22-05-01	7000	4000	14500
22-05-02	5000	5000	15500
22-05-03	3000	6000	17000
22-05-04	5000	5500	21500
22-05-05	4000	4500	17000

In [44]:

```
team_12 = team_df.drop(['toto', 'gildong' ], axis=1)
team_12
```

Out[44]:

	catanddog	apple	rowsum
22-05-01	7000	4000	14500
22-05-02	5000	5000	15500
22-05-03	3000	6000	17000
22-05-04	5000	5500	21500
22-05-05	4000	4500	17000

In [45]:

```
team_12.to_csv("team_12.csv", index=False)
team_12.to_excel("team_12.xlsx", index=False)
```

In [46]:

```
!dir
```

```
 D 드라이브의 볼륨: wjv_backup
 볼륨 일련 번호: 1E42-9FBD

 D:₩Github₩PythonBasic₩part02_library 디렉터리

2022-04-29  오후 11:14    <DIR>          .
2022-04-29  오후 11:14    <DIR>          ..
2022-04-25  오후 06:24             8,196 .DS_Store
2022-04-29  오후 11:10    <DIR>          .ipynb_checkpoints
2022-04-06  오전 11:08            90,717 C1_2B_1_matplotlib_datavis_corona.ipynb
2022-04-06  오전 11:08           765,170 C1_2_1_matplotlib.ipynb
2022-04-06  오전 11:08           660,295 C1_3_1_Seaborn_Basic.ipynb
2022-04-06  오전 11:08           267,152 C1_3_2_titanic_datavis.ipynb
2022-04-06  오전 11:08           345,816 C1_3_3_Titanic_EDA_DataPreprocessing.ipynb
2022-04-29  오후 11:12            60,127 C1_4_1_pandas_01.ipynb
2022-04-06  오전 11:08           450,281 C1_4_2_pandas_02_california.ipynb
2022-04-06  오전 11:08           224,102 C1_4_3_titanic_dataset.ipynb
2022-04-06  오전 11:08             9,392 C1_4_4_화장품관련키워드분석_ing.ipynb
2022-04-06  오전 11:08           360,931 C1_4_5_titanic_dataset_pandas_etc.ipynb
2022-04-06  오전 11:08         1,825,759 C1_4_6_corona_analysis.ipynb
2022-04-27  오후 01:01         1,778,360 C1_5_1_folium_local.ipynb
2022-04-06  오전 11:08         2,197,027 C1_5_1_folium_withColab.ipynb
2022-04-28  오후 01:47           717,253 C1_5_2_folium_seoul_data.ipynb
2022-04-06  오전 11:08           974,968 C1_6_1_plotly_basic_iplot.ipynb
2022-04-06  오전 11:08         2,456,472 C1_6_2_plotly_express_v11.ipynb
2022-04-06  오전 11:08    <DIR>          data
2022-04-06  오전 11:08    <DIR>          html_pdf
2022-04-27  오후 12:55            53,345 map.html
2022-04-27  오후 01:00            52,045 map_circle.html
2022-04-20  오후 07:21    <DIR>          part2_1_1_datavis
2022-04-06  오전 11:08    <DIR>          part2_1_2B_matplotlib
2022-04-06  오전 11:08    <DIR>          part2_1_2_matplotlib
2022-04-20  오후 06:03    <DIR>          part2_1_3_seaborn
2022-04-29  오후 08:23    <DIR>          part2_1_4A_pandas
2022-04-06  오전 11:08    <DIR>          part2_1_4B_pandas
2022-04-06  오전 11:08    <DIR>          part2_1_5B_folium_project
2022-04-06  오전 11:08    <DIR>          part2_1_5C_folium_실습결과물
2022-04-27  오후 01:06    <DIR>          part2_1_5_folium
2022-04-06  오전 11:08    <DIR>          part2_1_6B_plotly
2022-04-06  오전 11:08    <DIR>          part2_1_6C_실습결과물
2022-04-06  오전 11:08    <DIR>          part2_1_6_plotly
2022-04-27  오전 02:11    <DIR>          part2_2_1A_konlpy
2022-04-06  오전 11:08    <DIR>          part2_2_1B_konlpy_pratice
2022-04-06  오전 11:08    <DIR>          part2_2_1C_konlpy_결과물
2022-04-27  오후 12:55            75,899 Plugins_1.html
2022-04-28  오후 12:19           159,139 seoul_data.html
2022-04-28  오후 12:31           159,114 seoul_data_c.html
2022-04-29  오후 11:14               109 team_12.csv
2022-04-29  오후 11:14             5,492 team_12.xlsx
2022-04-28  오후 12:30           151,958 서울시공공화장실위치.html
              25개 파일          13,849,119 바이트
              20개 디렉터리  38,745,702,400 바이트 남음
```

REF

https://jakevdp.github.io/PythonDataScienceHandbook/04.08-multiple-subplots.html
(https://jakevdp.github.io/PythonDataScienceHandbook/04.08-multiple-subplots.html)

In [ ]:

```
```