판다스를 활용한 데이터 이해

In [1]:

```
myfood = ['banana', 'apple', 'candy']
print(myfood[0])
print(myfood[1])
print(myfood[2])
print(myfood[1:3]) # 첫번째 두번째 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 [7]:

```
import pandas as pd  # pandas를 불러오고 pd로 약자로서 쓰겠다.
```

In [8]:

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

In [9]:

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

```
pandas 버전 :  0.24.2
```

홍길동 팀별 대항 게임 5일간의 점수

In [10]:

```
score = Series( [1000,14000, 3000, 3000, 1000] )
print(score)
```

```
0     1000
1    14000
2     3000
3     3000
4     1000
dtype: int64
```

In [11]:

```
print("자료형 확인 : ", type(score))
```

```
자료형 확인 :  <class 'pandas.core.series.Series'>
```

In [12]:

```
## Series 인덱스 확인
```

In [13]:

```
print(score.index)
```

```
RangeIndex(start=0, stop=5, step=1)
```

In [14]:

```
## Series 값 확인
print(score.values)
```

```
[ 1000 14000  3000  3000  1000]
```

판다스 시리즈 인덱스 지정

In [15]:

```
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 [16]:

```
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 [17]:

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

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

In [18]:

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

```
1000
14000
3000
```

두 팀의 팀점수 합산해보기

- 길동팀의 3일간의 점수와 toto 팀의 3일간의 점수

In [19]:

```
from pandas import Series
```

In [20]:

```
gildong = Series([1500, 3000, 2500],
    index = ['2019-05-01', '2019-05-02', '2019-05-03'] )
toto = Series([3000, 3000, 2000],
    index = ['2019-05-01', '2019-05-03', '2019-05-02'] )
```

In [21]:

```
gildong + toto
```

Out[21]:

```
2019-05-01    4500
2019-05-02    5000
2019-05-03    5500
dtype: int64
```

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 [22]:

```
from pandas import DataFrame
```

In [23]:

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

Out[23]:

	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 [24]:

```
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[24]:

	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 [25]:

```
date = ['19-05-01','19-05-02', '19-05-03', '19-05-04', '19-05-05']
team_df = DataFrame(team_score,
    columns=['catanddog', 'toto', 'apple', 'gildong'],
    index=date)
team_df
```

Out[25]:

	catanddog	toto	apple	gildong
19-05-01	7000	1500	4000	2000
19-05-02	5000	3000	5000	2500
19-05-03	3000	5000	6000	3000
19-05-04	5000	7000	5500	4000
19-05-05	4000	5500	4500	3000

In [26]:

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

Out[26]:

```
19-05-01    1500
19-05-02    3000
19-05-03    5000
19-05-04    7000
19-05-05    5500
Name: toto, dtype: int64
```

In [27]:

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

Out[27]:

	toto	gildong
19-05-01	1500	2000
19-05-02	3000	2500
19-05-03	5000	3000
19-05-04	7000	4000
19-05-05	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 [28]:

```
print(team_df.loc[ '19-05-02' ] ) # 19-05-02 일
print("-----------")
print(team_df.loc[ ['19-05-02', '19-05-03'] ]) # 5월 2일, 3일
print("-----------")
print(team_df.loc[ '19-05-02': ]) # 5월 2일 이후 전체 데이터 가져오기
```

```
catanddog    5000
toto         3000
apple        5000
gildong      2500
Name: 19-05-02, dtype: int64
-----------
          catanddog  toto  apple  gildong
19-05-02       5000  3000   5000     2500
19-05-03       3000  5000   6000     3000
-----------
          catanddog  toto  apple  gildong
19-05-02       5000  3000   5000     2500
19-05-03       3000  5000   6000     3000
19-05-04       5000  7000   5500     4000
19-05-05       4000  5500   4500     3000
```

loc를 이용한 접근

In [29]:

```
## 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')
-----
19-05-01    1500
19-05-02    3000
19-05-03    5000
19-05-04    7000
19-05-05    5500
Name: toto, dtype: int64
-----
            toto  gildong
19-05-01  1500      2000
19-05-02  3000      2500
19-05-03  5000      3000
19-05-04  7000      4000
19-05-05  5500      3000
-----
            toto  apple  gildong
19-05-01  1500   4000      2000
19-05-02  3000   5000      2500
19-05-03  5000   6000      3000
19-05-04  7000   5500      4000
19-05-05  5500   4500      3000
```

iloc 속성을 이용한 행, 열, 데이터 접근하기

In [30]:

```
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: 19-05-01, dtype: int64
------
            catanddog  toto  apple  gildong
19-05-01         7000  1500   4000     2000
19-05-02         5000  3000   5000     2500
------
            catanddog  toto  apple  gildong
19-05-01         7000  1500   4000     2000
19-05-02         5000  3000   5000     2500
19-05-03         3000  5000   6000     3000
------
            catanddog  toto  apple  gildong
19-05-01         7000  1500   4000     2000
19-05-02         5000  3000   5000     2500
19-05-03         3000  5000   6000     3000
```

In [31]:

```
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
```

```
19-05-01    7000
19-05-02    5000
19-05-03    3000
19-05-04    5000
19-05-05    4000
Name: catanddog, dtype: int64
------
            catanddog  toto
19-05-01         7000  1500
19-05-02         5000  3000
19-05-03         3000  5000
19-05-04         5000  7000
19-05-05         4000  5500
------
            catanddog  toto  apple
19-05-01         7000  1500   4000
19-05-02         5000  3000   5000
19-05-03         3000  5000   6000
19-05-04         5000  7000   5500
19-05-05         4000  5500   4500
------
            catanddog  toto  apple
19-05-01         7000  1500   4000
19-05-02         5000  3000   5000
19-05-03         3000  5000   6000
19-05-04         5000  7000   5500
19-05-05         4000  5500   4500
```

In [32]:

```
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 [33]:

```
team_df.describe()
```

Out[33]:

	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 [34]:

```
## 날짜별 누적 통계
team_df.cumsum()
```

Out[34]:

	catanddog	toto	apple	gildong
19-05-01	7000	1500	4000	2000
19-05-02	12000	4500	9000	4500
19-05-03	15000	9500	15000	7500
19-05-04	20000	16500	20500	11500
19-05-05	24000	22000	25000	14500

날짜별 합계

In [35]:

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

```
19-05-01    14500
19-05-02    15500
19-05-03    17000
19-05-04    21500
19-05-05    17000
dtype: int64
```

In [36]:

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

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

In [37]:

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

Out[37]:

	catanddog	toto	apple	gildong	rowsum
19-05-01	7000	1500	4000	2000	14500
19-05-02	5000	3000	5000	2500	15500
19-05-03	3000	5000	6000	3000	17000
19-05-04	5000	7000	5500	4000	21500
19-05-05	4000	5500	4500	3000	17000

점수가 높은 날짜별로 확인해보자

In [38]:

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

Out[38]:

```
19-05-04    21500
19-05-05    17000
19-05-03    17000
19-05-02    15500
19-05-01    14500
Name: rowsum, dtype: int64
```

조건을 걸어 일정 이상의 팀 점수의 날만 확인해 보자.

- 17000이상인 날만 확인해 보기

In [39]:

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

Out[39]:

	catanddog	toto	apple	gildong	rowsum
19-05-03	3000	5000	6000	3000	17000
19-05-04	5000	7000	5500	4000	21500
19-05-05	4000	5500	4500	3000	17000

In [40]:

```
team_df
```

Out[40]:

	catanddog	toto	apple	gildong	rowsum
19-05-01	7000	1500	4000	2000	14500
19-05-02	5000	3000	5000	2500	15500
19-05-03	3000	5000	6000	3000	17000
19-05-04	5000	7000	5500	4000	21500
19-05-05	4000	5500	4500	3000	17000

In [41]:

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

Out[41]:

	catanddog	apple	rowsum
19-05-01	7000	4000	14500
19-05-02	5000	5000	15500
19-05-03	3000	6000	17000
19-05-04	5000	5500	21500
19-05-05	4000	4500	17000

In [42]:

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

Out[42]:

	catanddog	apple	rowsum
19-05-01	7000	4000	14500
19-05-02	5000	5000	15500
19-05-03	3000	6000	17000
19-05-04	5000	5500	21500
19-05-05	4000	4500	17000

In [43]:

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

In [46]:

```
# window의 경우
!dir *team*

# 리눅스 OS의 경우
# !ls *team*
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: EAD2-35E2

 C:₩Users₩toto₩Documents₩GitHub₩seoul_ktm04 디렉터리

2020-05-12  오전 07:56               109 team_12.csv
2020-05-12  오전 07:56             5,493 team_12.xlsx
               2개 파일               5,602 바이트
               0개 디렉터리  139,344,519,168 바이트 남음
```

REF

- pandas 공식 사이트 : https://pandas.pydata.org/ (https://pandas.pydata.org/) (https://pandas.pydata.org/ (https://pandas.pydata.org/))
- pandas 10 minute tutorial : https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html (https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html) (https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html (https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html))

In [ ]:

```
```