05 판다스를 활용한 데이터 이해

학습 내용

- 판다스를 이해하고 실습을 통해 알아본다.

01 파이썬 기본 다지기

리스트

In [40]:

```
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 [41]:

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

```
banana
apple
candy
```

딕셔너리(Dictionary)

In [42]:

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

In [43]:

```
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 [44]:

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

하나
둘
['a', 'b', 'c']

판다스 모듈 불러오기

In [45]:

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

In [46]:

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

In [47]:

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

pandas 버전 : 1.2.4

홍길동 팀별 대항 게임 5일간의 점수

[1000, 14000, 3000, 3000, 1000]

In [48]:

```
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 [49]:

```
## 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 [50]:

```
### 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 [51]:

```
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 [52]:

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

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

In [53]:

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

```
1000
14000
3000
```

두 팀의 팀점수 합산해보기

- 길동팀의 3일간의 점수와 toto 팀의 3일간의 점수

In [54]:

```
from pandas import Series
```

In [55]:

```
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 [56]:

```
gildong + toto
```

Out[56]:

```
2019-05-01    4500
2019-05-02    5000
2019-05-03    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 [57]:

```
from pandas import DataFrame
```

In [58]:

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

Out[58]:

	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 [59]:

```
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[59]:

	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 [60]:

```
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[60]:

	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

toto팀의 날짜별 점수를 확인해 보자.

- 팀별 컬럼명을 이용하여 접근이 가능하다.

In [61]:

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

Out[61]:

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

- toto와 gildong 팀 확인

In [62]:

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

Out[62]:

	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 [63]:

```
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 열에 접근

In [64]:

```
## 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 [65]:

```
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 [66]:

```
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
```

팀별 총합 및 평균 등의 통계는 얼마나 될까?

In [67]:

```
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 [68]:

```
team_df.describe()
```

Out[68]:

	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 [69]:

```
## 날짜별 누적 통계
team_df.cumsum()
```

Out[69]:

	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

In [70]:

```
## 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 [71]:

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

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

In [72]:

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

Out[72]:

	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 확인해 보자.

In [73]:

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

Out[73]:

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

조건을 걸어 일정 이상의 팀점수의 날만 확인해 보자.

- 17000이상인 날만 확인해 보기

In [74]:

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

Out[74]:

	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 [75]:

```
team_df
```

Out[75]:

	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

합계 점수가 1등 2등만 선택해 보자.

In [76]:

```
team_df.sum()
```

Out[76]:

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

In [77]:

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

Out[77]:

	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 [78]:

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

Out[78]:

	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 [79]:

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

In [81]:

```
!dir
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: 426B-0850

 C:₩Users₩toto₩Documents₩Github₩PythonBasic₩01_unit01_03_pandas 디렉터리

2021-09-15  오후 09:59    <DIR>          .
2021-09-15  오후 09:59    <DIR>          ..
2021-09-15  오후 08:11    <DIR>          .ipynb_checkpoints
2021-04-14  오후 05:43           312,855 01_02_Pandas알아보기_v10.pdf
2021-04-14  오후 05:43           363,656 01_03_pandas_01.html
2021-09-15  오후 09:58            55,745 01_03_pandas_01.ipynb
2021-06-28  오전 08:28         1,019,200 01_03_pandas_02.html
2021-06-28  오전 08:28           953,447 01_03_pandas_02.pdf
2021-09-15  오후 08:08           426,010 01_03_pandas_02_california.ipynb
2021-06-28  오전 08:32           824,995 01_04_titanic_dataset.html
2021-09-15  오후 08:10           221,769 01_04_titanic_dataset.ipynb
2021-06-28  오전 08:32           873,834 01_04_titanic_dataset.pdf
2021-04-14  오후 05:43             9,392 01_04_화장품관련키워드분석_ing.ipynb
2021-07-04  오후 11:53           950,935 01_05_titanic_dataset_pandas_etc.html
2021-09-15  오후 08:11           358,245 01_05_titanic_dataset_pandas_etc.ipynb
2021-07-04  오후 11:53         1,069,867 01_05_titanic_dataset_pandas_etc.pdf
2021-06-29  오전 09:02         1,779,353 01_unit01_03_pandas.zip
2021-04-14  오후 05:43           304,142 california_housing_test.csv
2021-04-14  오후 05:43         1,723,431 california_housing_train.csv
2021-07-02  오전 08:46               210 debug.log
2021-07-02  오전 08:34             3,258 decision_first_model.csv
2021-04-14  오후 05:43         1,279,001 pandas_신재용_0115.pdf
2021-09-15  오후 09:59               109 team_12.csv
2021-09-15  오후 09:59             5,493 team_12.xlsx
2021-04-14  오후 05:43           958,464 yelp_pandas_허수경.pdf
2021-04-14  오후 05:43           181,076 [REF]PyClass02_Bike_Model_Pandas.ipynb
2021-04-14  오후 05:43            17,986 상위키워드리스트_20210108.xlsx
              24개 파일          13,692,473 바이트
               3개 디렉터리  123,321,692,160 바이트 남음
```

REF

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

In [ ]:

```

```