데이터 처리 라이브러리
Pandas

학습 목표

(1) Pandas에 대해 이해한다.

(2) Pandas의 기본 자료 구조형에 대해 이해한다.

(3) Pandas의 기본 자료 구조형을 실습을 통해 생성할 수 있다.

목 차

1-1 Pandas에 대해 알아보기

(가) Pandas의 핵심 자료 구조는 Series와 DataFrame이다.

(나) 파이썬이 오픈소스 기반의 R과 더불어 빅데이터 분석 분야에서 인기가 높아진데는 여러가지 이유가 있지만, 그 중에 Pandas의 라이브러리의 역할이 크다.

1-2 Pandas에 앞서 알아보기

▶List(리스트)

In [2]:
```
myfood = ['banana', 'apple', 'candy']
print(myfood[0])
print(myfood[1])
print(myfood[2])
```

```
banana
apple
candy
```

In [3]:
```
for item in myfood:
    print(item)
```

```
banana
apple
candy
```

[리스트]
리스트를 이용하면 반복문을 통해 데이터 관리가 가능하다.

1-2 Pandas에 앞서 알아보기

▶ 딕셔너리(Dictionary)

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

```
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']}
```

1-3 Pandas 불러오기

▶Pandas 불러오기(모듈 불러오기)

```
import pandas as pd
```

```
from pandas import Series, DataFrame
```

1-3 Pandas 불러오기

▶Pandas의 Series 자료형 만들기

```
### 홍길동 팀의 5일간의 점수
### [1000, 14000, 3000, 3000, 1000]
score = Series( [1000, 14000, 3000, 3000, 1000] )
print(score)
```

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

1-4 Pandas Series 이해하기

▶ Pandas 기본(Series)

```
### 홍길동 팀의 5일간의 점수
### [1000, 14000, 3000, 3000, 1000]
score = Series( [1000, 14000, 3000, 3000, 1000] )
print(score)
```

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

Series	
Index	**Value**
0	**1000**
1	**14000**
2	**3000**
3	**3000**
4	**1000**

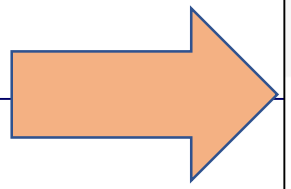
- Series 객체는 일차원 배열과 달리 각 값에 **연결된 값도 동시에 저장**
- Series 객체는 기본적으로 0으로 시작하는 정수 값을 사용하여 인덱싱을 한다.

1-5 Pandas Series

▶ Pandas Series index 값을 지정하기

```
### 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
```



```
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
```


1-5 Pandas Series

▶ Pandas Series index 값을 지정하기

```
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
```

인덱스을 **날짜로 지정**하여 **값의 접근 및 확인(출력)이 가능**하다.

1-5 Pandas Series

▶ 인덱스 확인하기, 인덱스에 해당되는 값을 확인하기

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

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

Series의 index인 날짜 확인

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

```
1000
14000
3000
```

해당 날짜에 해당하는 팀 점수(값) 확인

1-6 Pandas Series

▶ 인덱스 확인하기, 인덱스에 해당되는 값을 확인하기

- 길동팀의 3일간의 점수와 toto 팀의 3일간의 점수

```
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'] )
```


1-6 Pandas Series

▶ 데이터 확인해보기

```
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'] )
```

gildong

Series	
Index	**Value**
2019-05-01	1500
2019-05-02	3000
2019-05-03	2500

toto

Series	
Index	**Value**
2019-05-01	3000
2019-05-03	3000
2019-05-02	2000

1-6 Pandas Series

▶ Series 더하기 연산

gildong

Series	
Index	**Value**
2019-05-01	1500
2019-05-02	3000
2019-05-03	2500

toto

Series	
Index	**Value**
2019-05-01	3000
2019-05-03	3000
2019-05-02	2000

gildong + toto

```
gildong + toto
```

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

Index를 기준으로 값의 연산이 이루어지게 된다.
같은 index를 갖는 값이 더하기 연산을 수행.

1-7 DataFrame 객체

▶ DataFrame 의 이해

pandas의 Series가 1차원 형태의 자료구조라면 DataFrame은 여러 개의 컬럼(Column)으로 구성된 2차원 형태의 자료 구조

Series	
Index	**Value**
2019-05-01	1500
2019-05-02	3000
2019-05-03	2500

DataFrame		
	Series('col1')	Series('col2')
Index	**Value**	**Value**
2019-05-01	1000	1500
2019-05-02	2000	2000
2019-05-03	3300	3000

1-7 DataFrame 객체

DataFrame 의 객체 생성하기

- 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.

```
from pandas import DataFrame
```

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


1-7 DataFrame 객체

DataFrame 의 객체 생성하기

소스 코드

```
from pandas import DataFrame

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

실행 결과

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

1-7 DataFrame 객체

DataFrame의 column(열) 데이터 출력하기

'col1'의 열의 데이터를 출력하기

```
print(data['col1'])
```

```
0    1
1    2
2    3
3    4
Name: col1, dtype: int64
```

'col2'의 열의 데이터를 출력하기

```
print(data['col2'])
```

```
0    10
1    20
2    30
3    40
Name: col2, dtype: int64
```


1-8 DataFrame 객체의 내부 구조

▶ DataFrame 객체의 내부 구조

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

DataFrame			
	Series('col1')	**Series('col2')**	**Series('col3')**
Index	**Value**	**Value**	**Value**
0	**1**	**10**	**A**
1	**2**	**20**	**B**
2	**3**	**30**	**C**
3	**4**	**40**	**D**

(가) DataFrame 객체는 'col1', 'col2', 'col3'의 세 개의 Series 객체로 구성된다.

(나) Series 객체의 인덱스는 서로 동일하다.

1-9 DataFrame 알아보기

▶ DataFrame 만들기

```
from pandas import DataFrame

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

team_df = DataFrame(team_score)
team_df
```

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

(A) 파이썬 **딕셔너리 형태**로 각 컬럼은 데이터로 저장이 가능하다.

(B) DataFrame 클래스의 생성자로 넘겨준다.

(C) 컬럼의 순서가 다를 경우, **columns**를 이용하여 **컬럼의 순서를 지정**할 수 있다.

1-9 DataFrame 알아보기

▶ 데이터 프레임 객체 만들기 (index 추가, column 순서지정)

```
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
```

	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

(가) DataFrame 객체는 'catanddog', 'toto', 'apple', 'gildong'의 Series 객체로 구성된다.

(나) Series 객체의 인덱스는 date(날짜)로 서로 동일하다.

