

[TM_Python_Pandas] 01. Pandas 데이터 분석 기초

- 학습 목표
 - Pandas의 자료형 Series에 대해 이해할 수 있다.
 - 간단한 Series의 자료형을 만들어 보고 실습해 본다.
- 학습 내용
 - 1-1 Pandas 이해
 - (가) 핵심 자료 구조(Series, DataFrame)
 - (나) Series에 대해 알아보기
 - (다) Series와 리스트가 뭐가 다른가?
 - (라) 네이버, sk, kt의 주식 Series로 지정해 보기

1-1 Pandas 이해

파이썬이 오픈소스 기반의 통계언어 R과 더불어 빅데이터 분석 분야에서 인기가 높아진데는 여러가지 이유가 있다. 그 중에 pandas라는 라이브러리의 역할이 크다.

(가) 핵심 자료 구조

- Series
- DataFrame

```
In [49]: mystock = ['kakao', 'naver', 'daum']
          print(mystock[0])
          print(mystock[1])
          print(mystock[2])
```

```
kakao
naver
daum
```

```
In [50]: for stock in mystock:
          print(stock)
```

```
kakao
naver
daum
```

- 리스트, 튜플, 딕셔너리의 장점
 - (1) 리스트를 이용하면 반복문을 통해 데이터 관리가 가능하다.
 - (2) 튜플을 이용하면 () 을 이용. 리스트에 비해 속도가 빠르다.
 - (3) 딕셔너리를 이용하면 {}을 이용. 키값을 이용하여 값을 빠르게 찾을 수 있다.

(나) Series

모듈 불러오기

- 방법 1


```
from pandas import Series, DataFrame
```
- 방법 2


```
import pandas
print(pandas.Series)
```

```
In [51]: stock = Series([92600,92400,92100,94300,92300])
         print(stock)
```

```
0    92600
1    92400
2    92100
3    94300
4    92300
dtype: int64
```

Series	
Index	Value
0	92600
1	92400
2	92100
3	94300
4	92300

Series 객체는 일차원 배열과 달리 각 값에 연결된 인덱스 값도 동시에 저장합니다.

Series 객체를 생성할 때, 인덱스 값을 지정하지 않으면 기본적으로 **Series** 객체는 **0**으로 시작하는 정수값을 사용해 인덱싱한다.

그림 13.2 Series 객체의 내부 구조

```
In [52]: print(stock[0])
         print(stock[1])
         print(stock[3])
         print(stock[4])
```

```
92600
92400
94300
92300
```

(다) Series와 리스트가 뭐가 다른가?

```
In [53]: stockP = [92600,92400,92100,94300,92300]
         stock2 = Series(stockP, index=['2016-02-19',
                                       '2016-02-18',
                                       '2016-02-17',
                                       '2016-02-16',
                                       '2016-02-15'])
```

```
In [54]: print(stock2)
```

```
2016-02-19    92600
2016-02-18    92400
2016-02-17    92100
2016-02-16    94300
2016-02-15    92300
dtype: int64
```

Series	
Index	Value
'2016-02-19'	92600
'2016-02-18'	92400
'2016-02-17'	92100
'2016-02-16'	94300
'2016-02-15'	92300

인덱스 값으로 날짜에 해당하는 문자열을 지정.
정수값으로 인덱싱하는 대신 날짜를 의미하는 문자열 사용

```
print(stock2['2016-02-19'])
print(stock2['2016-02-18'])
```

그림 13.3 Series 객체의 내부 구조(2)

```
In [55]: for date in stock2.index:
         print(date)
```

```
2016-02-19
2016-02-18
2016-02-17
2016-02-16
2016-02-15
```

```
In [56]: for price in stock2.values:
         print(price)
```

```
92600
92400
92100
94300
92300
```

(라) 네이버, sk, kt의 주식 Series로 지정해 보기

```
In [57]: from pandas import Series, DataFrame

mine = Series([10, 20, 30], index=['naver', 'sk', 'kt'])
friend = Series([10, 30, 20], index=['kt', 'naver', 'sk'])
```

mine		friend	
Series		Series	
Index	Value	Index	Value
'naver'	10	'kt'	10
'sk'	20	'naver'	30
'kt'	30	'sk'	20

그림 13.4 Series 객체의 구조(3)

```
In [58]: ## pandas의 경우 '+' 연산을 수행하면 자기가 알아서 Index가 같은 것 끼리 연산한다.  
plus = mine + friend  
print(plus)  
  
kt      40  
naver   40  
sk       40  
dtype: int64
```