Series 기초

Pandas의 Series는 1차원 배열과 같은 자료 구조.
파이썬 리스트와 튜플도 1차원 배열과 같은 자료구조

사실 Pandas의 Series는
어떤 면에서는 파이썬의 리스트와 비슷하고
어떤 면에서는 파이썬의 딕셔너리와 닮은 자료구조

Series를 사용하기에 앞서
Pandas라는 모듈에서 직접 Seires와 DataFrame을
로컬 네임스페이스로 import

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

Series를 직접 로컬 네임스페이스로 import한 경우에는 pandas는 생략하고 바로 Series 라고만 적으면 된다.

```
from pandas import Series, DataFrame

kakao = Series([92600, 92400, 92100, 94300, 92300])
print(kakao)
```

Series 객체는 일차원 배열과 달리 값뿐만 아니라 각 값에 연결된 인덱스 값도 동시에 저장.

Series 객체 생성 시에
인덱스 값을 따로 지정하지 않으면
기본적으로 Series 객체는 0부터 시작하는 정숫값을 사용하여 인덱싱.

```
print(kakao[0])
print(kakao[1])
print(kakao[2])
```

```
print(kakao[0])
print(kakao[1])
```

```
print(kakao[2])

kakao2 = Series([92600, 92400, 92100, 94300, 92300],
                index=['2016-02-19', '2016-02-18', '2016-02-17',
                       '2016-02-16','2016-02-15'])
print(kakao2)
```

kakao2라는 Series 객체는
인덱스값으로 날짜에 해당하는 문자열을 지정했기 때문에
정수값으로 인덱싱 하는 것 대신 날짜를 의미하는 문자열을 사용하여 각 날짜에
대한 종가를 바로 얻어올 수 있다.

```
print(kakao2['2016-02-19'])
print(kakao2['2016-02-18'])
# 시리즈는 인덱스 저장영역과 값 저장영역이 1:1로 매칭되어서 같이 저장된다.
```

Series 객체의
index와 value라는 이름의 속성을 통해 접근할 수 있다.

```
for date in kakao2.index:
    print(date)

for ending_price in kakao2.values:
    print(ending_price)
```

pandas의 Series는 서로 다르게 인덱싱된 데이터에 대해서도 알아서 덧셈 연산을 처리해 준다.

```
mine = Series([10, 20, 30], index=['naver', 'sk', 'kt'])
friend = Series([40, 50, 60], index=['kt', 'naver', 'sk'])

merge = mine + friend
print(merge)
```