

map for series

map for series

- map을 사용하면 Series의 각 element에 특정 함수를 적용할 수 있음
- 사용 방법
 - `sr1.map(f1)`
 - `sr1`: Series 객체, `f1`은 특정 함수

```
def f1(x):  
    return x**2
```

```
sr1=pd.Series(np.arange(7))
```

```
sr1 (a)
```

```
sr1.map(f1) (b)
```

함수의 x parameter에는 Series의 각 element가 입력

```
Out[2]:  
(a) 0 0  
    1 1  
    2 2  
    3 3  
    4 4  
    5 5  
    6 6  
dtype: int32
```

```
Out[2]:  
(b) 0 0  
    1 1  
    2 4  
    3 9  
    4 16  
    5 25  
    6 36  
dtype: int64
```

map for series 예제2

- 조금 더 복잡한 함수

```
def f1(x):  
    if x>4:  
        return 'L'  
    elif x>2:  
        return 'M'  
    else:  
        return 'S'  
  
sr1=pd.Series(np.arange(7))  
  
sr1 (a)  
sr1.map(f1) (b)
```

```
Out [8]:  
(a) 0 0  
     1 1  
     2 2  
     3 3  
     4 4  
     5 5  
     6 6  
     dtype: int32  
  
Out [8]:  
(b) 0 S  
     1 S  
     2 S  
     3 M  
     4 M  
     5 L  
     6 L  
     dtype: object
```

map for series, lambda 사용

- 간단한 함수는 lambda를 활용하여 바로 함수 정의가 가능
- 사칙연산 수준의 간단한 operation는 Series객체에 직접 operation 적용가능 (broadcasting for Series)

```
def f1(x):  
    return x%3  
  
sr1=pd.Series(np.arange(7))  
  
sr1.map(f1) #1  
sr1.map(lambda x: x%3) #2  
sr1%3 #3 (Series에 직접 적용, Series 강의자료에 등장)
```

#1, #2, #3 결과

Out [9]:

0	0
1	1
2	2
3	0
4	1
5	2
6	0

dtype: int64

```
def f1(x):  
    return x+2  
  
sr1=pd.Series(np.arange(7))  
  
sr1.map(f1) #1  
sr1.map(lambda x: x+2) #2  
sr1+2 #3 (Series에 직접 적용, Series 강의자료에 등장)
```

#1, #2, #3 결과

Out [11]:

0	2
1	3
2	4
3	5
4	6
5	7
6	8

dtype: int64

map for series, lambda 사용

- 아래와 같이 여러 라인 필요 함수를 적용하려면 `lambda` 또는 `Series`에 직접 `operation` 적용 불가능 (전 슬라이드의 #2, #3 불가능)

```
def f1(x):  
    if x>4:  
        return 'L'  
    elif x>2:  
        return 'M'  
    else:  
        return 'S'  
  
sr1=pd.Series(np.arange(7))  
  
sr1  
sr1.map(f1)
```