https://3months.tistory.com/292 : 간단히 정리된 사이트
pandas는 데이터분석 라이브러리.
(1)dataframe 객체 : 2차원 배열, 서로 다른 자료형을 사용할 수 있다. 행, 열
(2)series 객체 : 1차원 배열, 1차원 ndarray와 호환 , 2개이상은 series아니고 dataframe이다.

// pandas도 numpy를 가져와서 만든다.

1. Series 데이터 생성하기 (index, value) : 생성해서 사용할 일은 없지만
   type 확인시키기...

```
import pandas as pd
import numpy as np

s1 =pd.Series([1,2,3])
s1                    : index value 짝으로 구성된다. 기본적으로 type 생성

s2= pd.Series(['a','b','c'])
s2                              : type is object(문자이므로)

s3 = pd.Series(np.arange(200))
s3
```

//인덱스의 변화를 본다. (numpy의 인덱스는 0부터시작되고 자동으로 생성된다.)

```
s4= pd.Series([1,2,3],[100,200,300])  : value, index 순서다
s4
s5 = pd.Series([1,2,3], ['a','m','k'])
s5
s6 = pd.Series(np.arange(5), np.arange(100, 105), dtype =int)
s6                    : type 지정가능 기본은 OS에 따라 다르다.
s6.index              : index 확인
s6.values             : value 확인
s6[0]                 : 오류, 0 index는 없다.
s6[104]
s6[104] =70           : 값변경
s6[104]               : 변경값 확인
s6[105] = 90          : 인덱스와 값 추가 후 s6 확인해 보기

s7 = pd.Series(np.arange(7), s6.index)
```

: s6의 index 재사용. 에러이유 살펴보고 수정하기(개수가 맞지 않다.)

2. Series 타입의 이해, 분석

Series가 제공하는 여러 가지 함수 사용

```
import pandas as pd
import numpy as np
s = pd.Series([1,1,2,1,2,2,2,1,1,3,3,4,5,5,7, np.NaN])
```

// np.NaN : Not a Number 숫자가 아닌 것(누락이나 다른값이 들어온 경우), 실제 데이터에서는 이런 값이 굉장히 많다. (올바르지 않은 데이터로 인식하므로 보통은 제거한다.)

NaN이 들어가니 정수들인데도 float으로 잡힌다.

```
s
len(s)
s.size          //개수를 확인
s.unique() //유일한 값만 ndarray반환
s.count()  //pandas는 알아서 NaN을 제외한 개수를 반환
s.mean()   //NaN을 제외한 평균

a = np.array([2,2,2,2,np.NaN])
a.mean()  //np는 NaN이 있으면 구할수 없다. 위 Series와 비교해 보기

b= pd.Series(a)         //np로 구한 a를 Series로 변경
b.mean()  //Series는 NaN을 무시하고 구한다.

s.value_counts()   //NaN을 제외하고 각 값들의 빈도(나온개수)를 반환

s[[5,7,8]]
s[[5,7,8]].value_counts()  //여러값을 가져올수 있다.

s.head()  //상위 기본5개 출력 s.head(n=7) -> 7개
s.tail() //하위 5개
```

3. Series 데이터 연산하기 (같은 인덱스 끼리 연산한다.)

```
s1 = pd.Series([1,2,3,4],['a','b','c','d'])
s2 = pd.Series([6,3,2,1], ['d','c','b','a'])
s1
s2
s1+s2   //같은 인덱스 끼리 더함을 알 수 있다. numpy는 같은 순서끼리더함.
s1 ** 2
s1 ** s2  //인덱스가 같은 것끼리 연산된다. 인덱스가 맞지 않으면 NaN이 됨
```

s2= pd.Series([6,3,2],['f','c','b']) 수정하여 인덱스를 다르게 지정해서 s1+s2

4. Series 데이터 Boolean Selection으로 데이터 선택하기

```
s = pd.Series(np.arange(10), np.arange(10)+1)
s
s > 5
s[s>5]          //True 인것만 필터링
  >>  짝수인것만 가져와서 s_odd 변수에 넣기  // s_odd = s[s%2==0]
  >>  0이 포함되지 않게 수정하기        // s_odd = s[(s%2==0) & (s!=0)]
>> 5보다 크고 8보다 작은 값을 출력해 보자  //s_odd = s[(s>5) & (s<8)]
>> s에서 7보다 큰 거의 개수를 출력
>> s에서 7보다 큰 값들의 합을 출력  //(s[s>=7]).sum()  //sum(s[s>=7])

>>  index를 기준으로 필터링
  s.index >5  //index 값을 기준으로 비교(6부터 True)
  s[s.index>5]
  s=pd.Series(np.arange(100,105), ['s','f','c','d','e'])
  s[s.index>'c']
```

5. Series 데이터 변경 & 슬라이싱하기

```
s= pd.Series(np.arange(100,105), ['a','b','c','d','e'])
s
>> 'a' index의 값을 200 으로 수정
  s['a']=200
>> 'k' index에 300을 넣어 s에 추가
  s['k'] = 300

s.drop('k')  //삭제
s            //s가 삭제되는 것이 아니라 지운 것이 새로운 객체로 만들어진다.
s.drop('k', inplace=True)          //해당객체에 바로적용(주의)
s
s[['a','b']]     //여러값을 가져올수 있다.
s[['a','b']] = [300,900] //여러값을 동시에 수정가능하다.
```

//index가 숫자인 경우와 문자인 경우 슬라이싱 비교하기

```
s1 = pd.Series(np.arange(100,105))
print(s1)
s1[1:3]

s2= pd.Series(np.arange(100,105), ['a','b','c','d','e'])
print(s2)
print(s2[1:3])
s2['b':'d']  //숫자는 -1까지, 문자는 문자까지
```