DALC 공통기초스터디 5주차

데이터는 그 형태나 속성이 매우 다양

->컴퓨터를 통해 분석하기 위해서는 일관성 있게 동일한 형태로 통합할 필요

판다스: 서로 다른 유형의 데이터를 공통의 형식으로 정리

판다스는 핵심 데이터 구조로서 1차원 배열 형태를 가진 Series와 2차원 배열 형태를 가진 DataFrame을 제공하며

판다스 버전확인: import pandas / pandas.__version

넘파이와 판다스 모두 임포트

시리즈-데이터가 순차적으로 나열된 1차원 배열의 형태

시리즈에서 인덱스는 데이터값과 일대일로 대응

(+인덱스는 데이터값의 위치를 나타내는 주소의 역할.)

데이터프레임- 2차원 배열 구조

여러 개의 열벡터들이 순서대로 결합된 2차원 행렬

Series

넘파이의 1차원 array와 유사

Data1=pd.Series([0.25,0.5,0.75,1.0])

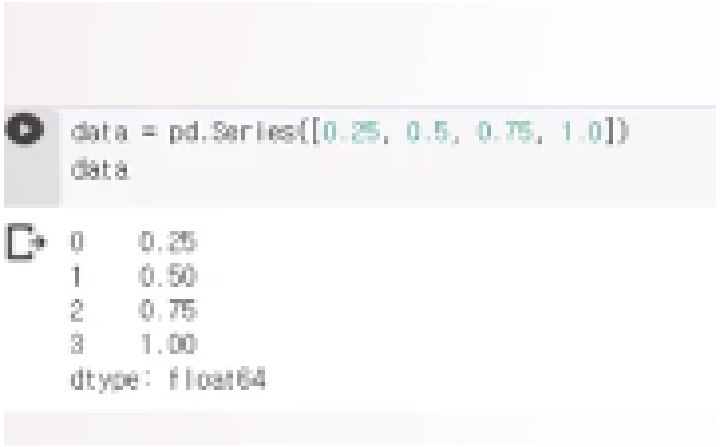
Data1.values

Data1.index

Data1[1]→0.5 출력

Data2=pd.Series([0.25,0.5,0.75,1.0], index=('a','b','c','d'))

Data2['a]→0.25출력



딕셔너리 구조

딕셔너리는 값에 키를 매핑하는 구조

->보통 판다스 Series는 딕셔너리를 이용하여 생성

Fruit_dict={‘apple’:1,’banana’:2,’watermelon’:3}

Fruit=pd.Series(Fruit_dict)

→apple 1

banana 2

watermelon 3

Fruit[‘apple’:’banana’] →apple 1

banana 2

딕셔너리 vs 시리즈

```
dic_data = {2:'a', 1:'b', 3:'c'}
dic_data
```

```
{1: 'b', 2: 'a', 3: 'c'}
```

딕셔너리-오름차순정렬

```
pd.Series({2:'a', 1:'b', 3:'c'})
```

```
2    a
1    b
3    c
dtype: object
```

시리즈: 입력한 순서대로

Pd.Serie({2:'a',1:'b',3:'c'},index=[3.2])

데이터 프레임

행과 열과 구성, 2차원 배열

행 인덱스는 시리즈의 인덱스와 동일한 역할

열인덱스: 열벡터 혹은 열이름

```
Fruit_dict={‘apple’:1,’banana’:2,’watermelon’:3}
Fruit=pd. DataFrame(Fruit_dict)
```

변수이름.columns

변수이름.index

```
Fruit=pd. DataFrame(Fruit_dict,colums=[‘rank’])
pd. DataFrame(Fruit_dict,index=[‘watermelon’],colums=[‘rank’])
```

랜덤값을 가진 데이터를 사용

넘파이 필요

DataFrame(np.random.rand(3,2))

인덱스 객체

A=pd.Index([2,3,5,7,11])

A[2]->A라는 인덱스 객체에서 인덱스 2인

요솟값을 참조

변경불가능

집합 연산

교집합 A&B

합집합 A|B

차집합 A^B

인덱서

Loc : 인덱스 이름 기준, 코드작성자 범위지정가능 범위의 끝포함

['a':'c']→a,b,c

변수이름.loc[]

Iloc: 정수형 인덱스 기준, 정수형 위치인덱스 사용, 범위의 끝 제외

[3:7]→3,4,5,6

변수이름.iloc[]

데이터프레임에 연산을 이용한 새로운 열추가

DF['A-B']=DF['A']-DF['B']

연산메소드

+: add()

-: sub()

*: mul()

/: div()

그리고 연산 결과로 NaN 값을 갖는 것을 피하기 위해 fill_value 옵션을 각 메소드에서 사용가능

Studentsum=sudent1.add(student2,fill_value=0)

학생1과 학생2를 더하되, 누락값을 0으로 설정

실습시간