PART 1.판다스 자료구조

**2-1. 시리즈**

시리즈: 데이터가 순차적으로 나열된 1차원 배열, 열벡터

인덱스와 데이터값이 일대일 대응(인덱스 통해 데이터 값에 직접 접근 가능)

파이썬 딕셔너리: (key: value)가 시리즈의 index와 value에 각각 매칭

∙시리즈 만들기

딕셔너리->시리즈 객체 변환: pandas.Series(딕셔너리)

리스트->시리즈 변환 시 동일하나, 키 값 별도 정의하지 않으면 정수형 위치 인덱스

튜플->시리즈 변환 시 리스트와 동일,

위의 두 경우 Series() 함수의 두번째 인자로 index=[“~~”] 로 인덱스 이름 설정 가능

자료형 확인 함수: type(객체)

∙인덱스 구조

인덱스: 해당 데이터값의 순서와 주소를 저장, 종류: 정수형 위치 인덱스, 인덱스 이름(인덱스 라벨)

인덱스 배열만 선택: Series객체.index

정수형 위치 인덱스의 경우 RangeIndex 객체로 표시되며, 범위의 마지막 값은 포함치 않음

출력 시 결과 예시: RangeIndex(start=0, stop=5, step=1)

데이터 배열만 선택: Series객체.values

∙원소 선택

정수형 위치 인덱스 사용: Series객체[정수]

인덱스 라벨 사용: Series객체[“문자열”]

여러 개의 원소 선택:

여러 개의 인덱스를 리스트 형태로 입력: Sr[[1,3]] 또는 Sr[[“생년월일”,”성별”]]

인덱스 범위를 지정: sr[1:2] 또는 sr[“생년월일”:”성별”]

정수형 위치 인덱스 사용 시 범위의 끝은 포함되지 않음.

인덱스 이름 선택 시 범위의 끝 포함됨.

**2-2. 데이터 프레임**

데이터 프레임: 2차원 배열, 여러 개의 열벡터들이 같은 행 인덱스를 기준으로 결합된 2차원 벡터 또는 행렬

행 인덱스(row index), 열 이름(column name or column label)

행: 개별 관측 대상 observation, 열: 공통의 속성을 갖는 일련의 데이터 record

∙데이터프레임 만들기

딕셔너리->데이터 프레임 변환: pd.Dataframe(딕셔너리 객체)

주로 여러 개의 시리즈를 원소로 갖는 딕셔너리가 전달됨: 딕셔너리의 키가 df의 column name이 되고, 값에 해당하는 각 리스트가 열이 됨.

이 경우 리스트가 열로 변환.

Df의 원소로 들어가는 객체: 2차원 배열: 여러 개의 리스트(투플)을 원소로 갖는 리스트(투플) 등

이 경우 리스트가 행으로 변환

행 인덱스/열 이름 설정

행 인덱스/열 직접 설정: pd.Dataframe(2차원 배열, index=행 인덱스 배열, columns= 열이름 배열)

행 인덱스 확인: df.index

열 이름 확인: df.columns

행 인덱스 변경: df.index= 새로운 행 인덱스 배열

열 이름 변경: df.columns=새로운 열 이름 배열

df.columns=[”연령”, ”이름”, ”소속”]

행 인덱스 또는 열 이름의 일부를 선택적 변경:

행 인덱스 변경: df.rename(index={기존 인덱스: 새인덱스, ‘’’})

열 이름 변경: df.rename(columns={기존 이름:새이름, ‘’’})

단, 새로운 객체 반환

원본 객체 수정하려면 두번째 인자로 inplace=True 추가

행/열 삭제

행 삭제: df.drop(행 인덱스 또는 배열, axis =0)

열 삭제: df.drop(열 이름 또는 배열, axis=1)

여러 개의 행 또는 열 삭제시 배열로 전달

Df.drop([“우현, 인아”], inplace= True)

Axis 설정 별도로 없다면 기본적으로 행 삭제.

새로운 객체 반환.

원본 수정 시 axis 다음 인자로 inplace=True

Df 복제하여 다른 변수에 저장:

모든 범위 복제: df2= df[:]

Df4= df.copy()

일부만 복제: df3=df.drop([“우현”,”인아”])

행 선택

인덱스 이름 기준 행 선택: df. loc[“a”]

정수형 위치 인덱스 사용: df. iloc[ 3 ]

행 범위 지정 가능

하나의 행만 선택 시 행 이름 제외, 행이 시리즈 객체(열 벡터)로 반환

여러 행 선택 시 :

2개 이상의 행 인덱스를 리스트 형태로 입력 : label2= df.loc[[“서준”, “우현”]]

Position2= df.iloc[[0,1]]

범위로 지정(슬라이싱): label3= df.loc[“서준”: “우현”]

Position3=df.iloc[0:1]

단, 인덱스 이름은 끝 지점 포함, 정수형 위치 인덱스는 끝 지점 포함치 않음.

열 선택

1개만 선택:

Df[“열이름”] 또는 df.열이름

두번째 방법은 열 이름이 문자열일때만 가능

시리즈 객체 반환

Math1=df[“수학”]

English=df.영어

열 n개 선택(데이터 프레임 생성)

Df[ [ 열1, 열2, ‘’’, 열n] ]

N=1이든 아니든, 2중 대괄호(대괄호 안에 열이름의 리스트 입력)는 데이터 프레임 반환 ( type(math2)) 로 확인 가능

범위 슬라이싱의 고급 활용

Df.iloc[ 시작인덱스 : 끝 인덱스: 슬라이싱 간격]

Df.iloc[ : :2] 모든 행에 대해 2간격으로 슬라이싱

Df.iloc[0:3:2] 0행부터 2행까지, 2간격으로 슬라이싱

0, 2행 추출

Df. iloc[ : : -1] 역순으로 인덱싱 하려면 음수 간격

원소 선택

인덱스 이름 이용: df.loc[행 인덱스, 열 이름]

정수 위치 인덱스 이용: df.iloc[행 번호, 열 번호]

행, 열의 좌표 입력 시 해당 위치의 원소 값 반환

원소 1개만 선택

df.loc[0,2] 또는 df.iloc[“서준”,”음악”]

이 경우 아무 인덱스도 가져오지 않음

1개의 행/열과 2개 이상의 열/행 선택 시 시리즈 객체 반환

df.loc[“서준”, [“음악”,”체육”]] 또는 df.loc[“서준”, “음악”:”체육”]

df.iloc[0, [2,3]] 또는 df.iloc[0, 2:4]

이 경우 2개 이상인 행/열의 인덱스만 가져옴, 시리즈이므로

2개 이상의 행과 2개 이상의 열 선택 시 데이터 프레임 반환 (또는 df의 일부분 선택)

df.loc[[“서준”,”우현”], [“음악”,”체육”]] 또는 위와 동일 형식

df.iloc[[0,1], [2,3]] 또는 위와 동일 형식

이 경우 해당하는 행과 열 인덱스를 가져옴

df의 하나의 열을 df의 인덱스로 설정: df.set\_index(“이름”, inplace = True)

다음장에서 자세히 배움

해당 열의 열 이름은 행 인덱스로 쓰이지 않음.(p29)

각 좌표 자리에 : 만 넣는 경우 모든 행/열 의미

범위 지정 가능.

열 추가

데이터 프레임의 마지막 열에 덧붙이듯 새로운 열(시리즈) 추가

기존의 df에 새로운 열 추가하여 해당 df 반환 (새 객체 아님)

열추가: df[“추가하려는 열 이름”] = 데이터 값

Df[‘국어’] = 80 시 기존 df의 모든 행(관측값)에 대해 동일하게 80이 입력됨

행 추가

행 추가: df. loc[“새로운 행 이름”] = 데이터 값(또는 배열)

하나의 데이터 값을 입력하는 경우 행의 모든 원소에 동일 값이 추가

열의 개수에 맞게 배열의 형태로 입력하는 경우 해당 위치에 해당 값이 추가

기존 행 벡터를 복사하여 새 행에 추가 가능

Df.loc[“행5”] = df.loc[3]

기존의 df에 새로운 열 추가하여 해당 df 반환 (새 객체 아님)

추가하려는 행 인덱스가 기존 인덱스와 중복 시 기존 행의 원소 변경

행 인덱스를 지정 시, 기존 인덱스의 순서와 상관 없이 지정 가능

?? 행인덱스가 정수형이 아니라 숫자이지만 문자로 입력된 상태???? 왜 loc로 함??그러려면 loc[“3”] 아님ㅁ??

원소 값 변경

원소 값 변경: df의 일부분 또는 원소 선택 = 새로운 값

원소 선택:

1개 선택: Df.iloc[0][3] = 80

Df.loc[‘서준’,’체육’]=90 또는 df.loc[‘서준’][‘체육’] =90

여러 개 선택:

동일 값으로 변경: df.loc[‘서준’, [‘음악’, ‘체육’]] = 50

각각 다른 값으로 변경(배열 이용): df.loc[‘서준’, [‘음악’, ‘체육’]] =100,50

행,열의 위치 바꾸기

선형대수학의 전치행렬과 같은 개념

행/열의 인덱스도 전치

두 번 전치 시 최초의 원본과 동일 형태

새로운 객체 반환

기존 객체 변경하려면 df=df.transpose() 메소드 이용

df=df.T 클래스 속성 이용

기존 객체에 새로운 전치된 객체 할당

**3. 인덱스 활용**

특정 열을 행 인덱스로 설정

원본 df 바꾸지 않고 새 df 객체 반환

기존의 행 인덱스는 삭제됨 (df에 있던 열이 행 인덱스로 이동한 꼴)

특정 열을 행 인덱스로 반환: df. set\_index( [‘열 이름’] 또는 ‘열이름’ )

원본 변경하려면 두번째 인수로 inplace= True 입력

두 개의 열을 행 인덱스로 지정: df.set\_index( [ ‘수학’, ’음악’ ] )

멀티 인덱스(Multi Index), 6장

행 인덱스 재배열

기존의 객체 변경 않고, 새로운 df 반환

새로운 배열로 행 인덱스를 재지정: df. reindex( 새로운 인덱스 배열 )

New\_index=[‘r0’,’r1’, ‘r2’, ‘r3’, ‘r4’]

Df.reindex[new\_index]

기존 df에 존재하지 않는 행 인덱스가 새롭게 추가되는 경우( 기존의 행 개수를 넘어서는 행 인덱스 개수가 지정된 경우)

해당하는 모든 열에 대해 NaN 입력 (Not a Number, 데이터가 존재하지 않는 상태)

NaN 이외의 유효값으로 채우려면 Reindex의 두번째 인자로 fill\_value=0 입력

행 인덱스 초기화

행 인덱스를 정수형 위치 인덱스로 초기화

기존의 행 인덱스는 첫번째 열로 이동함.

새 df 객체 반환

정수형 위치 인덱스로 초기화: df. reset\_index()

행 인덱스를 기준으로 데이터 프레임 정렬

행 인덱스 기준 정렬: df.sort\_index()

새 df 객체 반환

Ascending 옵션을 이용 정렬:

ascending = False 내림차순

ascending = True 내림차순

예시) r0 r1 r2 순으로 정렬

?????해보기: 정수, 알파벳, 한글, 정수와 알파벳 섞인 경우

특정 열의 데이터 값을 기준으로 데이터 프레임 정렬

열 기준 정렬: df. sort\_values()

새 df 객체 반환

Ascending 옵션 이용 정렬, True시 오름차순 False 시 내림차순

Ndf=df.sort\_valuse( by= ‘기준 하려는 열 이름’, ascending = False)

**4. 산술 연산**

판다스 객체의 3단계 산술연산 프로세스:

1. 행/열 인덱스 기준 모든 원소 정렬
2. 동일 위치 원소끼리 일대일 대응
3. 대응되는 원소끼리 연산 처리, 대응되는 원소 없으면 NaN 처리

**4-1. 시리즈 연산**

시리즈 vs 숫자

시리즈와 숫자 연산: series객체 + 연산자(+, -, \*, / ) + 숫자

연산 결과를 기존 시리즈의 인덱스와 순서대로 매칭하여 새 시리즈 반환

각각의 원소에 동일 산술 실시

예시) 정수 값 시리즈/200 시 실수 값 시리즈 반환

시리즈 vs 시리즈

시리즈와 시리즈 연산: series1 + 연산자(+, -, \*, / ) + series2

새로운 시리즈 객체 반환

시리즈의 모든 인덱스에 대하여 동일 인덱스 가진 원소끼리 계산

같은 인덱스(이름, 예를 들어 과목명)에 대해 각 시리즈마다 순서가 다르더라도, 판다스에서는 같은 인덱스를 찾아 정열한 후 연산

딕셔너리 데이터로 시리즈 만들기:

student1=pd.Series( { ‘국어’: 100, ‘영어’: 80, ‘수학’: 90 } )

여러 개의 시리즈를 데이터 프레임으로 합치기:

Result = pd. Dataframe([addition, substraction, multiplication, division], index = [ ‘덧셈’, ‘뺄셈’, ‘곱셈’, ‘나눗셈’])

정상적 연산이 불가능한 경우: NaN으로 입력.

연산 되는 두 시리즈의 원소 개수가 다르거나, 시리즈의 크기가 같지만 인덱스 값이 같은 경우

동일 인덱스가 모두 대응되더라도 어느 한쪽의 데이터 값이 NaN인 경우

시리즈 생성 시 NaN 입력:

Import numpy as np

Student1=pd.Series( {“국어”: np.nan, ‘영어’: 80, ‘수학’: 90} )

연산 메소드

연산 메소드 사용(시리즈와 시리즈) 연산 결과의 NaN을 다른 유효 숫자로 대체, 예시 0

Series. add(Series2, fill\_value=0)

Series. sub(Series2, fill\_value=0)

Series. mul(Series2, fill\_value=0)

Series. div(Series2, fill\_value=0)

시리즈와 시리즈 연산 시 특정 위치 데이터에 대하여, 정수/0 의 연산 결과: inf

무한대

**4-2. 데이터 프레임 연산**

Df는 여러 시리즈가 모인 것이므로, 시리즈 연산을 확장

행/열 인덱스 기준 정렬 후, 대응 원소끼리 연산

데이터프레임 vs 숫자

새 데이터프레임 객체 반환

기존 df의 형태는 유지한 채 모든 원소에 사칙 연산 수행

데이터 프레임과 숫자 연산: df + 연산자(+, -, \*, /) + 숫자

Seaborn 라이브러리가 제공하는 데이터셋 이용

Import Seaborn as sns

Titanic= sns. Load\_dataset(‘titanic’)

Df=titanic.loc[ : , [ ‘age’, ‘fare ’] ]

Titanic 데이터셋에서 age, fare의 두개 열 선택하여 새 df만들기

Df.head()

첫 5행만 표시

Df.tail()

마지막 5행만 표시

데이터프레임 vs 데이터프레임

각 DF의 동일 위치(동일 행, 동일 열)에 있는 원소끼리 계산,

새 데이터 프레임 반환

어느 한쪽 원소가 존재하지 않거나 NaN인 경우 연산 결과는 NaN