1. NUMPY

# NUMPY에서 활용되는 함수

널값을 주는 법 🡪 np.nan, None

Ndarray 생성하기 🡪 Np.array (ndarray란 넘파이에서 가장 강력한 무기로 칭송 받고 있는 N차원의 배열 객체. Ndarray는 기존 파이썬과 다르게 오직 같은 종류의 데이터만을 배열에 담을 수 있다. 넘파이라는 라이브러리를 활용하여 만든 배열을 ndarray라 부름.)

자료형 구하기 🡪 .dtype

자료형 변환(사이즈 크기) 🡪 .astype

배열 차원 및 행열 조정 🡪 .shape

몇 차원인지 알아보기 🡪 .ndim

배열에 수 랜덤(1~N) 배치 🡪 Np.arange

배열에 0 배치 🡪 Np.zeros

배열에 1 배치 🡪 Np.ones

배열 행열 사이즈 조정 🡪 .reshape

# 인덱싱 INDEXING

특정 데이터 추출 🡪 [,] [], arr.ix[](대괄호 안에 들어가는 건 행,열을 나타내는 숫자이거나 문자—명칭 or 위치 기반 인덱싱, arr.iloc[](대괄호 안에 들어가는 건 오로지 숫자 기반—위치기반 인덱싱), arr.loc[](대괄호 안에 들어가는 건 오로지 문자 기반—명칭 기반 인덱싱)

슬라이싱 🡪 [n:m], [n, m:u]

불리언 인덱싱 🡪 arr3[arr3>25]

# 정렬

오름차순, 내림차순 🡪 오름차순 같은 경우에는 sort, 내림차순 경우에는 reverse = True를 적용하면 됨. 🡪 np.sort(), array.sort()

수를 기준으로 정렬하되, 가시적으로는 인덱스가 출력. 🡪 np.argsort(), array.argsort()

# 선형대수 연산 – 행렬 내적과 전치 행렬 구하기

행렬 내적 연산 🡪 np.dot()

전치 행렬 🡪 np.transpose()

2. PANDAS

Numpy가 같은 데이터 타입의 배열만 처리할 수 있는 데 반해 pandas는 데이터 타입이 다양하게 섞여 있을 때도 처리할 수 있다.

# Pandas에서 활용되는 함수

Pd.series (series 데이터는 index와 values를 분리해서 가져올 수 있다.)

Pd.dataframe(series는 1차원 데이터 생성, dataframe은 2차원 데이터 생성, 밸류, 인덱스, 컬럼리스트를 다 정해주고 알 수 있음🡪 df.values(), df.columns(), df.values())

\*\* 파이썬의 리스트와 numpy의 배열과 달리 pandas의 데이터끼리는 서로 크기가 달라도 연산할 수 있다. 비는 경우는 nan으로 표시됨.

\*\* 연산 수행 시, 인자axis가 0이면 DataFrame의 values에서 열별(1열끼리 쫙 계산)로 연산을 수행하고, 1이면 행별(1행끼리 쫙 계산)로 연산을 수행한다. Axis 인자를 설정하지 않으면 기본값으로 0이 설정된다. 드롭할 때는, axis가 1이면 열 삭제. 0이면 행삭제. 약간 연산수행이랑 드롭이랑은 반대 느낌?

Data.head(), data.tail() 🡪 head & tail은 각 데이터프레임의 맨 윗부분, 맨 아랫부분을 출력

Data.shape 🡪 row와 column출력

Data.describe 🡪 데이터프레임에 대한 대략적인 정보를 출력

Data.value\_counts() 🡪 column 내부에 어떤 value가 몇 개 있는지를 출력해준다.

# 데이터 셀렉션 및 필터링

[]

🡪Df[‘col2’]

Df[0:3]

Iloc[]연산자: row와 column의 index순서(숫자)로 추출하기

Df.iloc[0,1]

Loc[]연산자: row의 index value와 column의 이름으로 추출하기

Df.loc[1,’col2’]

Df2.loc[‘a’,’col3’]

Ref) dataframe데이터 중 하나의 원소만 선택하려면 다음 방법 중 하나를 이용하면 된다.

Dataframe\_data.loc[index\_name][column\_name]

Dataframe\_data.loc[index\_name,column\_name]

Dataframe\_data[column\_name][index\_name]

Dataframe\_data[column\_name][index\_pos]

Dataframe\_data[column\_name].loc[index\_name]

# Dataframe\_data1.append(dataframe\_data2 [,ignore\_index=True])

🡪 append는 데이터2를 데이터1 밑으로 붙이기

🡪 ignore\_index=True의 의미는 원래 기존의 데이터2의 데이터들이 갖고 있는 인덱스를 지워주고 1에 맞춰서 새롭게 생성 시켜주는 것.

# Dataframe\_data1.join(dataframe\_data2)

🡪 join은 데이터2를 데이터1 옆으로 붙이기.

# Dataframe\_left\_data.merge(dataframe\_right\_data, how =merge\_method, on=key\_label)

How에는 left, right, outer,inner가 들어가는데, sql이랑 비슷한 개념임

On에는 기준 교집합 열이 들어가면 됨.

# 정렬,결손 데이터 처리하기

Df.sort\_values(by=[‘col1’])

pd.isnull(data).sum(0) 🡪 0이면 데이터 프레임의 열로 널값을 센다. 1이면 데이터 프레임의 행으로 널값을 센다.

데이터 프레임 생성, 외부 파일 읽어오기 🡪 pd.read.csv()

데이터 프레임의 윗부분, 아랫부분 출력 🡪 df.head(), df.tail()

데이터 프레임의 모양 표현 🡪 df.shape

데이터 프레임에 대한 대략적인 정보 출력 🡪 df.describe(), r의 str(), dplyr::glimpse()와 유사

# 결측값

데이터 프레임에 널값이 있는지 확인하는 법 🡪 isnull, isna (여기 뒤에 sum()함수를 붙이거나, mean()함수를 붙일 수 있다. 널값이 총 몇개인지 확인 가능.)

널값 메꾸기 🡪 fillna()

\*\* list, ndarray, dataframe등의 타입을 알고자 할때는 type(변수)이렇게 하고, list, ndarray, dataframe안에 있는 데이터의 타입을 확인하려면 변수.type()이렇게 확인할 것.

추측이지만, 리스트도 그렇고 리스트 자체가 아니라, 리스트 안에 있는 원소의 특징을 알고자 할 때는 함수를 뒤에 변수.함수() 이렇게 쓰는 듯?