

데이터 분석 라이브러리



numpy array 생성

- ndarray
 - 리스트를 바탕으로 생성되는 n-dimensional array

```
import numpy as np
a = [1, 2, 3, 4]
print(type(a),a)
b = np.array(a)
print(type(b),b)
<class 'list'> [1, 2, 3, 4]
<class 'numpy.ndarray'> [1 2 3 4]
```

- 인덱싱 및 슬라이싱

import numby as no a = np.array([1,2,3,4]) • 리스트와 동일 print(a[1],a[-1],a[1:],a[::2])

2 4 [2 3 4] [1 3]



numpy

- 과학 연산을 위해서 설계된 다차원 배열 처리를 위한 파 이썬 라이브러리
- 내부적으로는 C언어로 구현되어 있어서 숫자(특히 배열) 연산을 할 때 매우 빠르고 효율적임
- 사용 방법
 - import numpy
 - import numpy as np
 - from numpy import *



NumPy v1.20.0 Type annotation supp

https://numpy.org/



대표적인 numpy 메소드

• arange(): 리스트의 range와 동일

```
import numby as no
print(np.arange(10))
print(np.arange(1,10,2))
print(np.arange(1,2,0.1))
[0 1 2 3 4 5 6 7 8 9]
[1 3 5 7 9]
[1. 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9]
```

zeros(), ones(), eye()

```
import numby as no
print(np.zeros(2))
print(np.ones(2))
print(np.eye(2))
fo. o.1
[1. 1.]
[[1. 0.]
[0. 1.]]
```



대표적인 numpy 메소드

• sqrt(), sin(), cos()

```
import numpy as np
print(np.sqrt(2))
print(np.pi)
print(np.sin(np.pi/2))
print(np.cos(0))
1.4142135623730951
3.141592653589793
1.0
```

random.rand(), random.choice()

```
import numpy as np
print(np.random.rand(5))
print(np.random.choice(10,5))

[0.46052375 0.85205319 0.97010211 0.93133171 0.51178033]
[9 0 6 6 7]
```



행렬 연산

• 행렬 pointwise 연산자: +, -, *, /

```
import numpy as np
a = np.array([[1,2,3],[3,2,5]])
b = np.array([[-1,3,5],[1,4,2]])
print(a+b)
print(a-b)
print(a*b)
print(a/b)
[[0 5 8]
[4 6 7]]
[[ 2 -1 -2]
[ 2 -2 3]]
[[-1 6 15]
[3 8 10]]
[[-1.
               0.66666667 0.6
 [ 3.
                                     ]]
```

Edited by Harksoo Kim

*n*차원 행렬 만들기

• 원소가 10개인 2차원 행렬 만들기

```
import numby as no
       a = np.arange(10).reshape(2,5)
      print(a)
       print(a.shape)
      print(a.ndim)
      print(a.size)
       [[0 1 2 3 4]
      [5 6 7 8 9]]
(2, 5)
       10
                                                 import numpy as np
                                                 a = np.arange(10).reshape(2,5)
                                                 print(a)
• 2차원 행렬 인덱싱
                                                 print(a[1,2])
                                                 print(a[(0,3)])
                                                 [[0 1 2 3 4]
                                                  [56789]]
```



Edited by Harksoo Kim

행렬 연산

• 곱 및 전치

$$\mathbf{A}\mathbf{B}^{T} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 5 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 3 & 4 \\ 5 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \cdot (-1) + 2 \cdot 3 + 3 \cdot 5 & 1 \cdot 1 + 2 \cdot 4 + 3 \cdot 2 \\ 3 \cdot (-1) + 2 \cdot 3 + 5 \cdot 5 & 3 \cdot 1 + 2 \cdot 4 + 5 \cdot 2 \end{bmatrix}$$

$$= \begin{bmatrix} 20 & 15 \\ 28 & 21 \end{bmatrix},$$

$$\mathbf{A}^{T}\mathbf{B} = \begin{bmatrix} 1 & 3 \\ 2 & 2 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} -1 & 3 & 5 \\ 1 & 4 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \cdot (-1) + 3 \cdot 1 & 1 \cdot 3 + 3 \cdot 4 & 1 \cdot 5 + 3 \cdot 2 \\ 2 \cdot (-1) + 2 \cdot 1 & 2 \cdot 3 + 2 \cdot 4 & 2 \cdot 5 + 2 \cdot 2 \\ 3 \cdot (-1) + 5 \cdot 1 & 3 \cdot 3 + 5 \cdot 4 & 3 \cdot 5 + 5 \cdot 2 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 15 & 11 \\ 0 & 14 & 14 \\ 2 & 29 & 25 \end{bmatrix}.$$

$$\mathbf{A}^{T}\mathbf{B} = \begin{bmatrix} 1 \cdot (-1) + 3 \cdot 1 & 1 \cdot 3 + 3 \cdot 4 & 1 \cdot 5 + 3 \cdot 2 \\ 2 \cdot (-1) + 5 \cdot 1 & 3 \cdot 3 + 5 \cdot 4 & 3 \cdot 5 + 5 \cdot 2 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 15 & 11 \\ 0 & 14 & 14 \\ 2 & 29 & 25 \end{bmatrix}.$$



행렬 연산

• 한꺼번에 더하기/빼기

```
import numpy as np
a = np.zeros(5)+2
print(a)
b = np.ones(5)-2
print(b)
[2. 2. 2. 2. 2.]
[-1. -1. -1. -1.]
```

• 마스크 씌우기

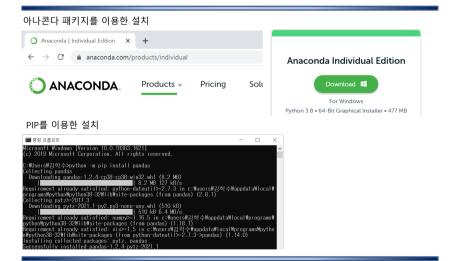
import numpy as np
a = np.arange(-3,3)
print(a(0))
print(a[a<0])
print(a[asca)>2])

[-3 -2 -1 0 1 2]
[True True True False False False]
[-3 -2 -1]
[-3]



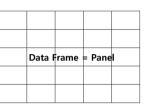
Edited by Harksoo Kim

pandas 설치



pandas

- 데이터 프레임을 효과적으로 다룰 수 있도록 도 와주는 라이브러리
- panel datas의 약자로 파이썬 기반 데이터 분석 에서 매우 많이 사용됨
- 사용 방법
 - import pandas
 - import pandas as pd





데이터 프레임 (Data Frame)

- 데이터 프레임
 - 행(인덱스)과 열(컬럼)로 구분
 - 인덱스는 별도로 지정을 하지 않으면 정수로 지정 (한번 설정된 인덱스는 불변)





데이터 프레임 만들기

- 생성할 데이터 프레임 구조
 - 크기: 3행 2열
 - 인덱스 명: m1, m2, m3
 - 컬럼 명: price, num
 - 값: 랜덤 숫자

```
import pandas as pd import numpy as np 

df = pd.DataFrame(np.random.rand(3,2), index=['m1','m2','m3'], columns=['price','num']) 

price num 
m1 0.496474 0.937166 
m2 0.311304 0.478858 
m3 0.953548 0.241274
```



행과 열 전치

- 전치(transpose) 메소드
 - 형식: 데이터프레임.T



특정 열 접근 및 마스킹

- 특정 열 접근
 - 형식: 데이터프레임[컬럼 명]
- 마스킹
 - 불리언 로직(True/False) 결과를 데이터 프레임에 입력

```
price num
m1 0.750011 0.751260
                                                                                  m2 0.203087 0.284539
 import pandas as pd
                                                                                  m3 0.509435 0.501892
import numby as no
                                                                                       0.751260
                                                                                       0.284539
df = pd.DataFrame(np.random.rand(3,2), index=['m1','m2','m3'], columns <math>\frac{m^2}{m^3}
                                                                                      0.501892
print(df,"#n")
                                                                                  Name: num, dtype: float64
print(df['num'],"\n")
print(df['num']>0.5,"\n")
                                                                                      False
df2 = df[df['num']>0.5]
                                                                                       True
print(df2)
                                                                                  Name: num, dtype: bool
                                                                                  m1 0.750011 0.751260
                                                                                  m3 0.509435 0.501892
```



컬럼(열) 간 연산

- 컬럼 가 연산
 - 컬럼 내 모든 값들 사이에 연산이 적용됨
 - 연산 결과를 새로운 컬럼에 저장할 수 있음



인덱스(행) 내 연산

- 인덱스 내 연산
 - numpy에서 기준 축을 지정하여 접근 가능
 - 기준 축을 기준으로 각 행이 numpy array로 취급

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.rand(3,2), index=['m1','m2','m3'], columns=['price','num'])
df['sum']=np.sum(df, axis=1)
df

price num sum
m1 0.306168 0.591926 0.898094
m2 0.607162 0.610202 1.217364
m3 0.984488 0.268280 1.252768
```



실습

- 다음과 같은 절차에 따라 총 가격이 가장 비싼 가계를 출력하는 프로그램을 작성 하시오.
 - 단가와 개수로 이루어진 데이터 프레임 생성

	unit price	number
store1	1000	25
store2	280	120
store3	900	30

- 단가와 개수를 곱한 총 가격이 추가된 데이터 프레임 생성

	unit price	number	total price
store1	1000	25	25000
store2	280	120	33600
store3	900	30	27000

- 총 가격이 가장 비싼 가계 출력

	unit price	number	total price
store2	280	120	33600



실습

• 2차원(3*6) 행렬 형태로 표현된 3개의 문서와 1차원 벡터 형태의 질의 사이의 코사인 유사도를 계산하여 출력하시오.



Edited by Harksoo Kim

실습

?



질의응답



Homepage: http://nlp.konkuk.ac.kr E-mail: nlpdrkim@konkuk.ac.kr

