

Chapter 5.

테이블형 수치데이터 다루기

모두의 데이터과학 with python

2018-01-30

김예은

이번 chapter에서는...

numpy 모듈에 대해서 공부한다.

이번 chapter에서 다루는 대부분의 함수 앞에
"numpy."나 "np."를 붙이는 것을 잊지 말자!

numpy.를 붙이지 않는 경우는 array객체에 대해 함수를 수행하는 경우
ex. `__array__.transpose()`

numpy모듈에 내장되어 있는 함수를 사용하는 것이기 때문!

NumPy module

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

[출처] <http://www.numpy.org/>

NumPy module

- Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
- NumPy is licensed under the [BSD license](http://www.numpy.org/licenses), enabling reuse with few restrictions.

[출처] <http://www.numpy.org/>

Unit 21. 배열 만들기

1. Numpy의 array vs Python 기본 list

1. Python의 list보다 간편하고 빠르다
2. 같은 type의 아이템(=원소)만 갖는다.

```
> list1 = [1, "2", 3]                                # [1, "2", 3]
```

```
> array1 = numpy.array(list1)                         #["1", "2", "3"]
```

Unit 21. 배열 만들기

[참고] Module에 들어있는 함수 보기

```
> Everything = dir(numpy)  
> print (everything)
```

Unit 21. 배열 만들기

[참고] Module aliasing(별명)

```
> import numpy as np
```

주의! Aliasing을 하면 원래 모듈명(numpy)은 python이 알아듣지 못한다.

Unit 21. 배열 만들기

2. Array 만드는 방법

: **list**나 **tuple**같은 item들의 collections을 변환

`array(object, dtype = None, copy = True,
order = "K", subok = False, ndim = 0)`

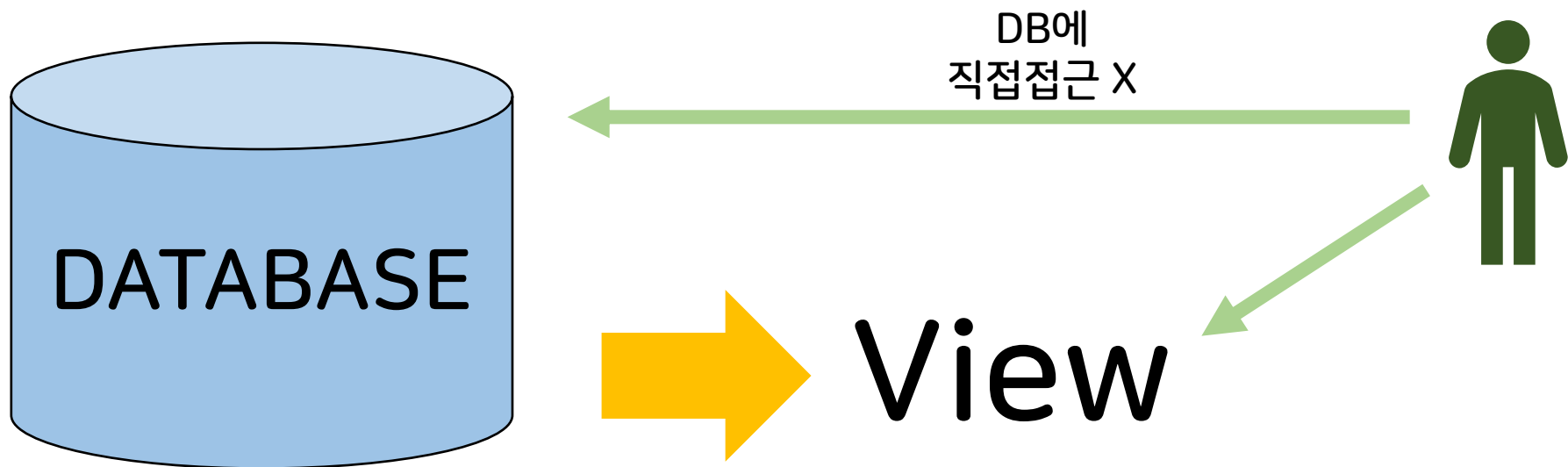
1. dtype :data dtype. 명시하지 않으면 python이 추론
2. copy: view or copy?(boolean, optional)
3. order, subok, ndim은 아래 링크 참고

출처: <https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.array.html>

Unit 21. 배열 만들기

[보충] View 란?

Database에 실제 데이터가 저장되어 있고,
View는 가상 데이터이다. 하지만, 모든 역할은 동일하게 수행가능
Raw data를 보호하기 위한 목적으로 사용된다.



Unit 21. 배열 만들기

3. 알아두면 편리한 Array

1. 1로 이루어진 array & 0으로 이루어진 array
ones([rowdim, coldim], dtype = 데이터 타입)
zeros([rowdim, coldim], dtype = 데이터 타입)
2. 빈 array : 내용이 항상 0인 것은 아니다
empty([rowdim, coldim], dtype = 데이터 타입)
3. 단위행렬 : eye(dim, k = 0)
4. 등차수열: arange(start_point, end_point, d)

Unit 21. 배열 만들기

4. Array의 data type변경 하기

`__array__.astype(dtype, copy = True)`

1. `dtype = {int, str, float,...}`

2. `copy` : view or copy?

5. Array 복사하기

`__array__.copy()`

Unit 22. 행렬 전환과 형태 변형하기

1. 배열 모양 바꾸기(행렬 형태로 변환)

array.reshape(n,m)

2. 전치 행렬

array.T

array.transpose()

array.swapaxes(0,1) #0번째 축과 1번째 축을 바꿈

Unit 22. 행열 전환과 형태 변형하기

3. transpose() : 축 순서 바꾸기

array.transpose(x,y,z)

0번째 축 자리에는 x번째 축을 넣고,

1번째 축 자리에는 y번째 축을 넣고,

2번째 축 자리에는 z번째 축을 넣고,

Unit 22. 행열 전환과 형태 변형하기

4. swapaxes() : 두 축 교환

array. swapaxes(x,y)

x번째 축과 y번째 축을 교환

Unit 22. 행열 전환과 형태 변형하기

[예제]

```
> T = range(1,28)
```

```
> Matrix = np.array(t)
```

```
> Matirx=Matrix.reshape(3,3,3)
```

Unit 22. 행열 전환과 형태 변형하기

[예제]

> Matrix.transpose(1,0,2)

> Matrix.swapaxes(0,1)

Unit 22. 행열 전환과 형태 변형하기

[예제]

> Matrix.transpose(2,1,0)

> Matrix.swapaxes(0,1); Matrix.swapaxes(1,2)

Unit 22. 행렬 전환과 형태 변형하기

[실습1] 회귀분석

다중 회귀분석을 수행하는 함수 regression()을 만들어보자

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

1. 함수 정의하기
`def multireg(...):`

2. 행렬의 곱셈
`matrix1 @ matrix2`

3. 역행렬
`from numpy.linalg import inv`
`inv(Matrix)`

Unit 22. 행렬 전환과 형태 변형하기

[실습1] 회귀분석

다중 회귀분석을 수행하는 함수 regression()을 만들어보자

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

함수1) multireg2(y, x1, x2)

- 설명변수가 두 개인 회귀모형: $(\beta_0 \quad \beta_1 \quad \beta_2)$ array리턴
- y, x1, x2 배열을 입력 받는다.
- 세 배열의 길이가 다르면 에러를 출력한다.

Unit 22. 행렬 전환과 형태 변형하기

[실습1] 회귀분석

다중 회귀분석을 수행하는 함수 regression()을 만들어보자

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

함수2) multireg2(y, x1, x2, x3)

- 설명변수가 세 개인 회귀모형 : $(\beta_0 \ \beta_1 \ \beta_2 \ \beta_3)$ array리턴
- y, x1, x2, x3 배열을 입력 받는다.
- 네 배열의 길이가 다르면 에러를 출력한다.

Unit 22. 행렬 전환과 형태 변형하기

[실습1] 회귀분석

다중 회귀분석을 수행하는 함수 regression()을 만들어보자

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

주의! X 행렬의 첫 번째 열은 1로 이루어져있다.(상수항)

$$\begin{pmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ 1 & x_{31} & x_{32} \end{pmatrix}$$

Unit 22. 행렬 전환과 형태 변형하기

[실습1] 회귀분석

다중 회귀분석을 수행하는 함수 regression()을 만들어보자

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

다음 코드를 실행하여 임시 데이터로 사용하자.

```
> x1 = 3 * np.random.random(10)
```

```
> x2 = np.arange(-1,1,0.2)
```

```
> x3 = np.random.random(10) + np.arange(0,1,0.1)
```

```
> y = 10*x1 + 3*x2 + np.random.random(10)
```

Unit 23. 인덱싱과 자르기

1. Boolean indexing

[책 예제]

```
> dirty = np.array([9,4,1,-0.01,-0.02, -0.001])
```

```
> dirty[dirty < 0] = 0
```

Unit 23. 인덱싱과 자르기

[참고] 계산처리순서

1. `&, |, !` : 비트단위 연산자
2. `<, ==` : 관계형 연산자
3. `or, and, not` : python의 bool연산

적절한 괄호사용이 중요!

Unit 23. 인덱싱과 자르기

2. Smart indexing, Smart slicing

1. 여러 행이나 열을 선택하는 방법은 두가지

✓ : 연속적으로 선택, 앞뒤에 숫자 쓰지 않으면 모두 선택을 뜻함

✓[, ,] 원하는 행이나 열 선택

✓다음 두 식에 대한 결과는 같다.

> sap2d[:,[0,1]]

> sap2d[:,0:2]

2. 위 두 방법을 사용하지 않으면 결과는 1차원 배열

ex. > sap2d[:,1]

Unit 24. 브로드캐스팅

1. 브로드캐스팅 = 배열에서 벡터 연산
수학적으로 가능한 차원과 스칼라에 대한 연산 가능
→ "선형결합"

[참고](list의 *) vs (array의 *)

```
> seq = range(1,6)
```

```
> seq * 5          #[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, ..., 1, 2, 3, 4, 5]
```

```
> seq_a = np.array(seq)
```

```
> seq_a * 5        #[5, 10, 15, 20, 25]
```

Unit 24. 브로드캐스팅

2. 사칙 연산을 이용해 노이즈 만들기

```
> noise = np.eye(4) + 0.01*np.ones(4,)
```

```
> noise = np.eye(4) + 0.01*np.random.random([4,4])
```

```
> np.round(noise,2)
```

Unit 25. 유니버설 함수 파헤치기

1. 유니버설함수: 브로드캐스팅의 함수형 버전
한번의 함수 호출로 배열의 모든 아이템(원소)을 처리

[책 예제] 주식이 떨어진 S&P이름 출력하기

MMM	ABT	ABBV	CAN	ACE	ATVI	ADBE	ADT
140.49	40.68	55.7	98.2	109.96	35.71	87.85	30.22
0.97	41.53	57.21	99.19	111.47	36.27	89.11	30.91

MMM은 주식이 140.49 → 0.97

ABT는 주식이 40.68 → 41.53

Unit 25. 유니버설 함수 파헤치기

[책 예제]

```
> stocks = np.array([140.49, 0.97, 40.68, 41.53, 55.7,  
57.21, 98.2, 99.19, 109.96, 111.47, 35.71, 36.27, 87.85,  
89.11, 30.22, 30.91])
```

140.49	0.97	40.68	41.53	...	87.85	89.11	30.22	30.91
--------	------	-------	-------	-----	-------	-------	-------	-------

(16 x 1)

Unit 25. 유니버설 함수 파헤치기

[책 예제]

```
> stocks = stocks.reshape(8,2).T
```

140.49	0.97
40.68	41.53
55.7	57.21
98.2	99.19
109.96	111.47
35.71	36.27
87.85	89.11
30.22	30.91

(8 x 2)

140.49	40.68	55.7
0.97	41.53	57.21

...

87.85	30.22
89.11	30.91

(2 x 8)

Unit 25. 유니버설 함수 파헤치기

[책 예제]

```
> fall = np.greater(stocks[0], stocks[1])
```

TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
------	-------	-------	-------	-------	-------	-------	-------

```
> sap[fall]
```

Unit 25. 유니버설 함수 파헤치기

2. nan (=not a number)

1. `isnan(_array_)`

: 각 원소(아이템)이 nan인지 boolean array를 리턴

Unit 25. 유니버설 함수 파헤치기

[실습2] 이상치 찾기 :code 1줄~3줄

다음 코드를 실행해 임시 데이터로 사용하자.

단, 이상치는 3시그마 밖에 있는 자료로 한다.

```
> data = np.random.random(20) * 10
```

```
> data[len(temp)-1] = np.random.random(1) * 50
```

```
[False, ..., False, True]
```

값을 갖는 outlier라는 이름의 array를 만들어보자.

Unit 26. 조건부 함수 이해하기

1. where(c,a,b)

✓삼항 연산자(if~else)

✓c : boolean array

✓a,b : array

✓c[i]가 true이면 a[i],false면 b[i]를 리턴

```
for i in range(0,len(a)):
    if a[i] > b[i] :
        x[i] = 1;
    else:
        x[i] = 0;
```

```
x = np.where(a > b, 1, 0)
```

Unit 26. 조건부 함수 이해하기

[참고] java의 3항 연산자
두 코드는 같은 역할을 수행한다.

```
if( a > b ){  
    x = 1;  
}else{  
    x = 0;  
}
```

```
x = (a > b) ? 1 : 0;
```

Unit 26. 조건부 함수 이해하기

- 2. `any()` : 배열 중 한 원소라도 `true`면 `true`를 반환
`all()`: 배열의 모든 원소가 `true`이면 `true`를 반환
- 3. `nonzero()`: 0이 아닌 모든 원소의 인덱스를 리턴

Unit 27. 배열집계와 정렬하기

1. 기초 통계량

1. `sum()`
2. `std()`, `mean()`
3. `min()`, `max()`

2. `cumsum()` : $array[i] = \sum_{i=0}^{len(array)} array[i]$

3. `cumprod()` : $array[i] = \prod_{i=0}^{len(array)} array[i]$

4. `sort()`: 정렬

Unit 27. 배열집계와 정렬하기

[책 예제] 단리와 복리 계산(이자율 3.75%, 30년)

```
> rate = 0.0375
```

```
> term = 30
```

```
> simple = (rate + np.ones(term)).cumsum()
```

```
> compound = ((1 + rate)*np.ones(term)).cumprod() - 1
```

Unit 27. 배열집계와 정렬하기

[실습3] 모평균 가설검정 : 함수내부 code 7줄
실습 2에서 사용한 데이터에서 이상치를 제외하고
" $H_0 : \mu = 5$ "이라는 가설을 검정해보자.

- ✓ 모평균 가설검정을 수행하는 함수(MUtest)를 만들고,
가설검정을 해보자
- ✓ 매개변수로 입력 받는 것은 데이터, 모평균(=5), 유의수준(=.05)
- ✓ 결과로 다음과 같이 출력할 것
ex> p-value = 0.36233로 0.025보다 크므로 " H_0 : 모평균은 5"라는
귀무가설을 기각하지 않는다.

Unit 28. 배열을 셋처럼 다루기

1. `unique(array)`

: array의 중복되지 않는 모든 원소로 구성된 array를 리턴
빈도는 세지 않는다.

[책 예제] 뉴클레오티드

```
> dna = "AGTCCGCGAATACAGGCTCGGT"
```

```
> dna_as_array = np.array(list(dna))
```

```
> np.unique(dna_as_array)
```


Unit 28. 배열을 셋처럼 다루기

2. `in1d(needle, haystack)`
: `needle`의 원소가 `haystack`안에 존재하는지 여부를
불 배열로 리턴
3. `union1(A,B)`: 합집합
[참고]set으로 리턴 `set(A).union(B)`
4. `intersec1d(A,B)`:교집합
[참고]set으로 리턴 `set(A).intersection(B)`

Unit 29. 배열 저장하고 읽기

1. `save(file, arr)` : .npy 파일에 배열 저장
`load(file)` : 배열을 읽어옴
→ 바이너리포맷. numpy만이 다룰수 있다
2. `loadtxt()`: 텍스트파일에서 테이블형 데이터를 불러옴
`savetxt()`: 배열을 텍스트파일에 저장

numpy는 필요하다면 파일을 자동으로 생성하고 연다.

.gz 파일을 자동으로 압축풀어줌

주석처리와 구분자를 다루고, 원치않는 행을 스킵하는 방식을
조정가능

Unit 30. 합성 사인파 만들기

[책 예제]

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import matplotlib
```

```
#신호, 잡음, '악기'정보를 상수로 정의한다.
```

```
SIG_AMPLITUDE = 10; SIG_OFFSET = 2; SIG_PERIOD =  
100
```

```
NOISE_AMPLITUDE = 3
```

```
N_SAMPLES = 5 * SIG_PERIOD
```

```
INSTRUMENT_RANGE = 9
```

Unit 30. 합성 사인파 만들기

#사인곡선을 구성하고 잡음을 섞어 넣는다.

```
times = np.arange(N_SAMPLES).astype(float)
```

```
signal = SIG_AMPLITUDE * np.sin(2 * np.pi * times /  
SIG_PERIOD) + SIG_OFFSET
```

```
noise = NOISE_AMPLITUDE * np.random.normal(size =  
N_SAMPLES)
```

```
signal += noise
```

Unit 30. 합성 사인파 만들기

#음역대를 벗어난 스파이크를 제거한다.

```
signal[signal > INSTRUMENT_RANGE] = INSTRUMENT_RANGE
```

```
signal[signal < -INSTRUMENT_RANGE] = -INSTRUMENT_RANGE
```

Unit 30. 합성 사인파 만들기

#결과를 플롯으로 시각화한다.

```
matplotlib.style.use("ggplot")
```

```
plt.plot(times, signal)
```

```
plt.title("Synthetic sine wave signal")
```

```
plt.xlabel("Time")
```


```
plt.ylabel("Signal + noise")
```

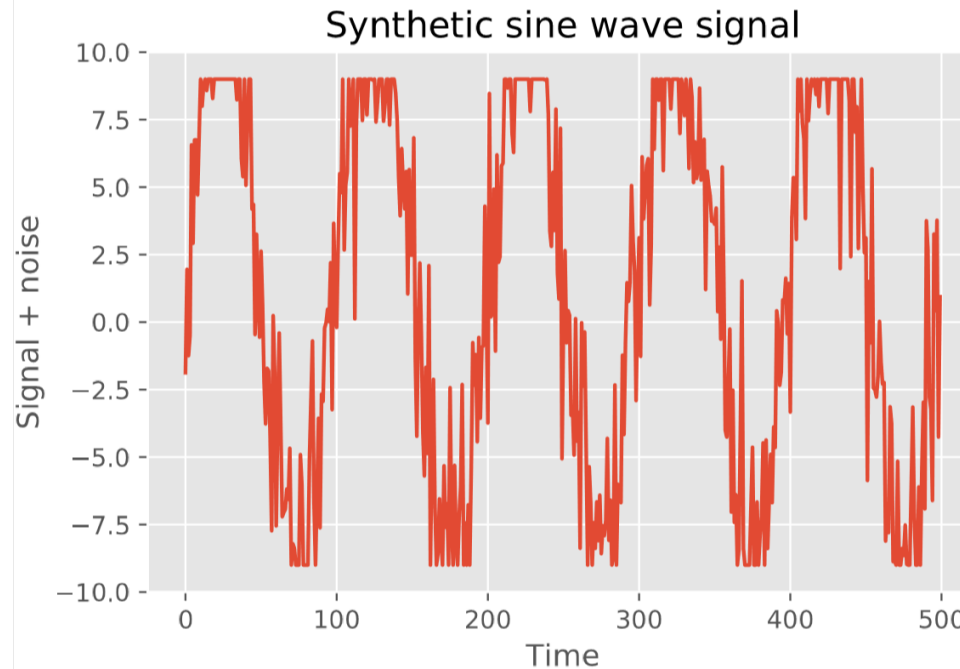
```
plt.ylim(ymin = -SIG_AMPLITUDE, ymax =  
SIG_AMPLITUDE)
```

Unit 30. 합성 사인파 만들기

#pdf파일로 저장

```
plt.savefig("signal.pdf")
```

 signal.pdf	2018-01-28 오후 2...	PDF 파일	16KB
--	--------------------	--------	------



Python공부 방법 추천

codecademy

<https://www.codecademy.com/learn>

실습

[실습1] 회귀분석 함수 만들기

```
def multireg2(y,x1,x2)  
def multireg3(y,x1,x2,x3)
```

[실습2] 이상치 찾기

```
outlier = [False, ..., False, True]
```

[실습3] 모평균 가설검정 함수 만들기

```
def MUtest(data, mu, alpha)
```