Lambda & MapReduce

Pythonic Code

최성철 교수
Director of TEAMLAB

Lambda

Lambda

- 함수 이름 없이, 함수처럼 쓸 수 있는 익명함수
- 수학의 람다 대수에서 유래함

General function

```
def f(x, y):
    return x + y

print(f(1, 4))
```

Lambda function

```
f = lambda x, y: x + y
print(f(1, 4))
```

Lambda

- Python 3부터는 권장하지는 않으나 여전히 많이 쓰임

```
f = lambda x, y: x + y
print(f(1, 4))
```

```
f = lambda x: x ** 2
print(f(3))
```

```
f = lambda x: x / 2
print(f(3))
```

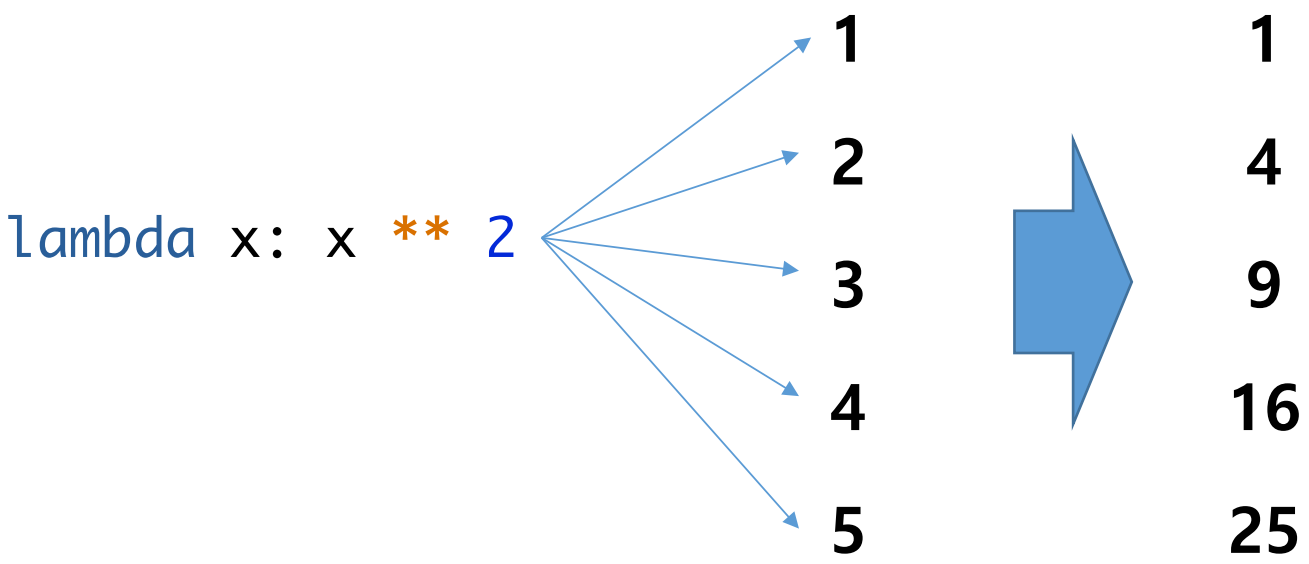
```
print((lambda x: x +1)(5))
```

Map & Reduce

Map function

- Sequence 자료형 각 element에 동일한 function을 적용함

```
map(function_name, list_data)
```




```
ex = [1,2,3,4,5]
f = lambda x: x ** 2
print(list(map(f, ex)))


f = lambda x, y: x + y
print(list(map(f, ex, ex)))
```

Map function

- 두 개 이상의 list에도 적용 가능함, if filter도 사용가능

```
ex = [1,2,3,4,5]
f = lambda x, y: x + y
print(list(map(f, ex, ex)))
```

```
list(
   map(
   lambda x: x ** 2 if x % 2 == 0
   else x,
   ex)
)
```

Map function

- python3 는 iteration을 생성 → list을 붙여줘야 list 사용가능
- 실행시점의 값을 생성, 메모리 효율적

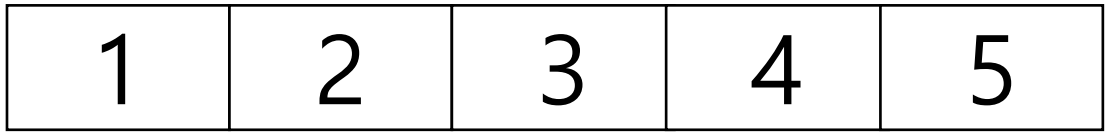
```
ex = [1,2,3,4,5]
print(list(map(lambda x: x+x, ex)))
print((map(lambda x: x+x, ex)))

f = lambda x: x ** 2
print(map(f, ex))
for i in map(f, ex):
    print(i)
```

```
result = map(f, ex)
print(next(result))
```

Reduce function

- map function과 달리 list에 똑같은 함수를 적용해서 통합

```
from functools import reduce
print(reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]))
```



Summary

- Lambda, map, reduce는 간단한 코드로 다양한 기능을 제공

- 그러나 코드의 직관성이 떨어져서 lambda나 reduce는 python3에서 사용을 권장하지 않음

- Legacy library나 다양한 머신러닝 코드에서 여전히 사용중

Human knowledge belongs to the world.

Asterisk

Pythonic Code

최성철 교수
Director of TEAMLAB

Asterisk

- 흔히 알고 있는 * 를 의미함

- 단순 곱셈, 제곱연산, 가변 인자 활용 등 다양한게 사용됨

***args**

```
def asterisk_test(a, *args):
    print(a, args)
    print(type(args))


asterisk_test(1,2,3,4,5,6)
```

****kargs**

```
def asterisk_test(a, **kargs):
    print(a, kargs)
    print(type(kargs))

asterisk_test(1, b=2, c=3,
d=4, e=5, f=6)
```

Asterisk – unpacking a container

- tuple, dict 등 자료형에 들어가 있는 값을 unpacking
- 함수의 입력값, zip 등에 유용하게 사용가능

```
def asterisk_test(a, *args):
    print(a, args)
    print(type(args))


asterisk_test(1, *(2,3,4,5,6))
```

```
def asterisk_test(a, args):
    print(a, *args)
    print(type(args))


asterisk_test(1, (2,3,4,5,6))
```

Asterisk – unpacking a container

```
a, b, c = ([1, 2], [3, 4], [5, 6])
print(a, b, c)


data = ([1, 2], [3, 4], [5, 6])
print(*data)
```

```
def asterisk_test(a, b, c, d,):
    print(a, b, c, d)
data = {"b":1 , "c":2, "d":3}
asterisk_test(10, **data)
```

```
for data in zip(*([1, 2], [3, 4], [5, 6])):
    print(data)
```

Human knowledge belongs to the world.

Linear algebra concepts

Pythonic Code

최성철 교수
Director of TEAMLAB

Linear Algebra
선형 대수

Linear Algebra

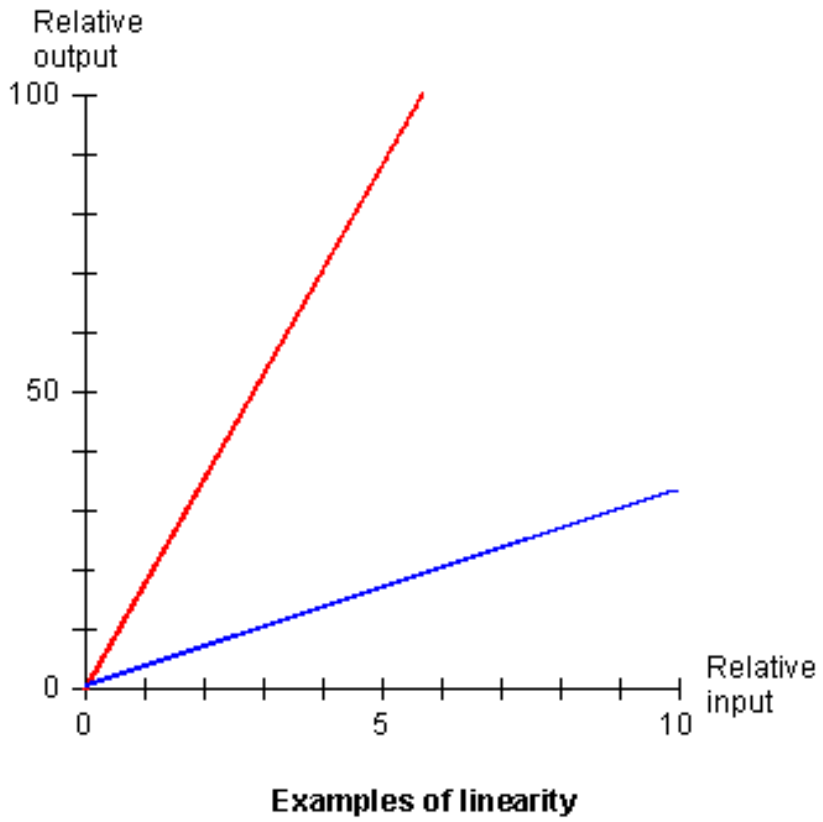
선형(線型)

- 직선 또는 그와 비슷한 성질을 가진 대상
- 일반적으로 1차함수 형태 = 선형성

대수학(代數學)

- 문자에 숫자를 대입하여 푸는 문제
- 1개 이상의 변수를 가진 다항방정식을 푸는 문제

Linear Algebra



Examples of linearity

Examples:

1) $3a + 6y$

$= 3 (a + 2y)$

2) $fs + qr - fr - qs$

$= fs - fr + qr - qs$

$= f (s - r) + q (r - s)$

$= (f - q) (s - r)$

Vector

Vector

- “배달, 운반하다”의 의미를 가진 단어가 어원

- 크기와 방향을 모두 가지는 것을 벡터(vector)

- 크기만 가지는 경우 스칼라(scalar)라고 지칭함

Vector

$$u = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, v = \begin{bmatrix} 1 \\ -3 \end{bmatrix}, w = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

일반적인 vector의 표현방법

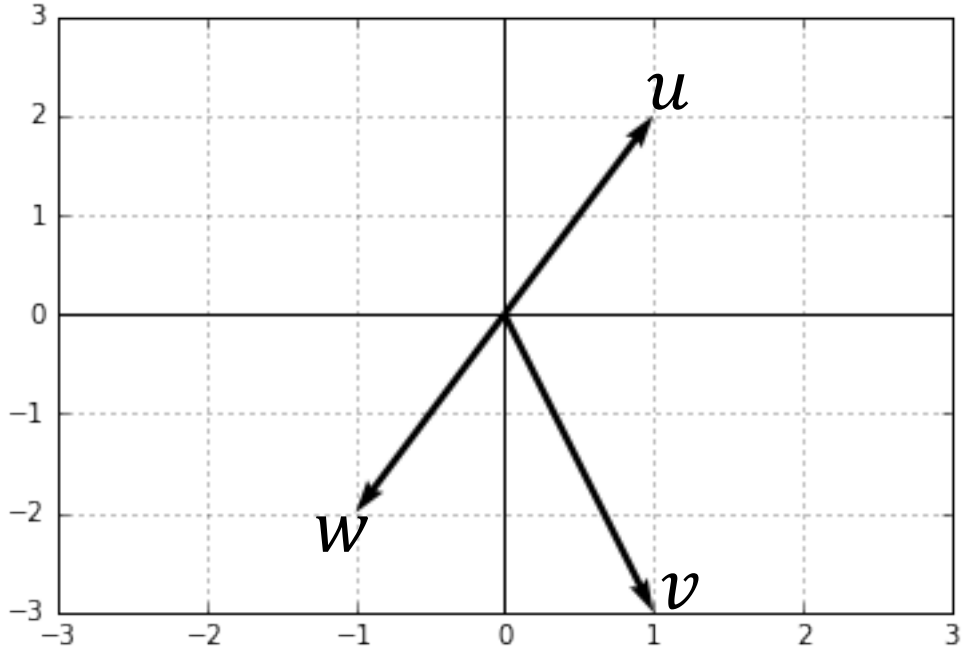
Vector

- 고등학교때는 평면 좌표 위의 Vector를 배움
- Vector라 하면 다음과 같이 상상됨

$$u = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$v = \begin{bmatrix} 1 \\ -3 \end{bmatrix}$$

$$w = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$



Vector

- 그러나 일반적인 연산에서는 2개 이상의 변수를 처리
- 이 변수 하나하나가 vector의 element로 표시됨

$$\mathbb{R}^4 \ni [1,2,-1.0,3.14] \quad \text{또는} \quad \mathbb{R}^4 \ni \begin{bmatrix} 1 \\ 2 \\ -1.0 \\ 3.14 \end{bmatrix}$$

Vector

- $[1,2,-1.0,3.14]$은 R상의 4-Vector 또는 4-dimentional vector
- 이 변수 하나하나가 vector의 element로 표시됨
- $\mathbb{R}^4$: 4개의 실수를 원소를 가지는 벡터 집합

$$\mathbb{R}^4 \ni [1,2,-1.0,3.14] \quad \text{또는} \quad \mathbb{R}^4 \ni \begin{bmatrix} 1 \\ 2 \\ -1.0 \\ 3.14 \end{bmatrix}$$

($\mathbb{R}^n$은 실수집합)

Vector의 표현

3-dimensional column vector $\begin{bmatrix} 9 \\ 2 \\ 3 \end{bmatrix}$

3-dimensional row vector $[9,2,3]$

3-dimensional zero vector $[0,0,0]$

Vector의 계산

n-vector들의 덧셈

- Element 위치를 대응하는 원소들의 덧셈으로 정의됨

$$[u_1, u_2, \dots, u_n] + [v_1, v_2, \dots, v_n] = [u_1 + v_1, u_2 + v_2, \dots, u_n + v_n]$$

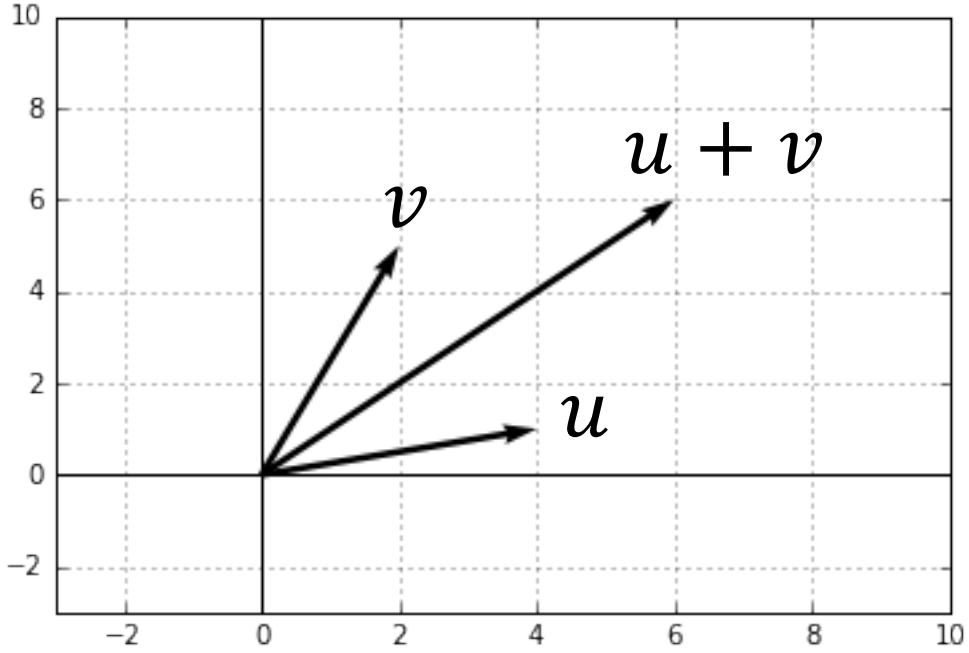
Example

$$[2, 2] + [2, 3] + [3, 5] = [7, 10]$$

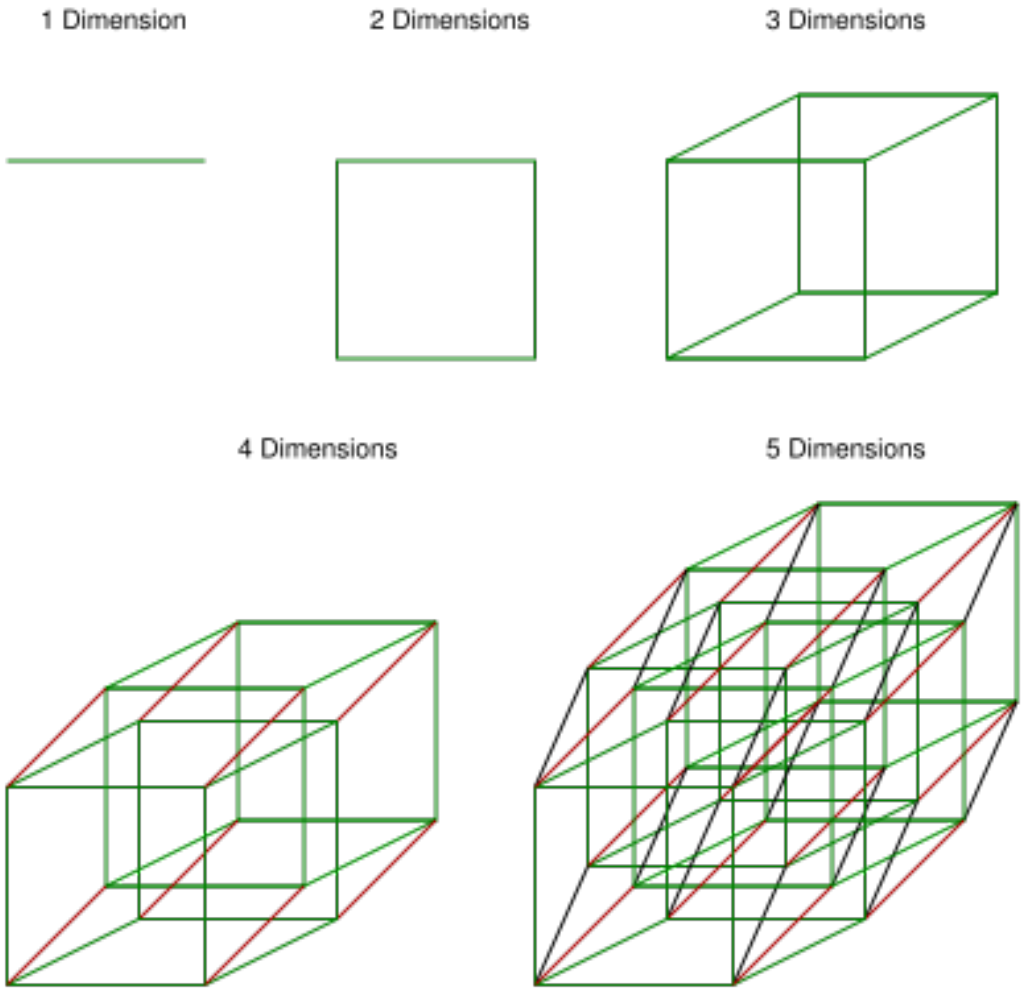
Vector의 계산

$$u = \begin{bmatrix} 4 \\ 1 \end{bmatrix} \quad v = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$[4, 1] + [2, 5] = [6, 6]$$



Vector의 계산


1 Dimension
2 Dimensions
3 Dimensions
4 Dimensions
5 Dimensions

Associativity / Commutativity

임의의 벡터 u, v, w은 아래와 같이 결합 법칙과 교환 법칙이 성립됨

결합 법칙 $$(u + v) + w = u + (v + w)$$

교환 법칙 $$u + v = v + u$$

Example

$$([2, 2] + [2, 3]) + [3, 5] = [2, 2] + ([2, 3] + [3, 5]) = [7, 10]$$

$$[2, 2] + [2, 3] = [2, 3] + [2, 2]$$

Scalar-Vector product

Proposition

$$\alpha(u + v) = \alpha u + \alpha v$$

Example

$$2([1, 2, 3] + [4, 4, 4]) = 2[4, 6, 7] = [8, 12, 14]$$

Matrix


THE
MATRIX
MUSIC FROM THE MOTION PICTURE
PARENTAL
ADVISORY
EXPLICIT CONTENT

Matrix

- 격자
- 수학에서는 사각형으로 된 수의 배열을 지칭
- 한 개 이상의 벡터(vector) = Matrix

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}, \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}, \begin{bmatrix}1\\ -2\end{bmatrix}, \begin{bmatrix}2 & 1\end{bmatrix}$$

일반적인 matrix의 표현방법

Matrix Representation

- Matrix는 m개의 행(row) , n개의 열(column)로 구성
- $m \times n$행렬이라고 함 (m by n 이라고 읽음)

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Matrix Representation

행렬 A의 i행, j열의 값을 A의 ij번째 element라 함 a_{ij}로 표시함

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$a_{11} = 1$$
$$a_{23} = 6$$
$$a_{31} = 7$$

Matrix Equal(등치)

- 두개의 행렬 A와 B가

  1) A와 B의 같은 크기이면서

  2) 모든 i, j 에 대하여 같은 값을 가지면,

  즉 $a_{ij} = b_{ij}$ 이면 등치라고 한다

Matrix Equal(등치)

$$A = \begin{bmatrix} 3 & 6 \\ 4 & 5 \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

이면 A = B 이기 위한 조건은 아래와 같다.

$$b_{11} = 3, b_{12} = 6, b_{21} = 4, b_{22} = 5$$

Matrix Addition

두개의 행렬 A와 B가 같은 크기

➔ C = A + B가 $a_{ij} + b_{ij}$ 로 구성됨

$$A = \begin{bmatrix} 3 & 6 \\ 4 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 8 \\ 6 & 7 \end{bmatrix}$$

$$C = A + B = \begin{bmatrix} 3 & 6 \\ 4 & 5 \end{bmatrix} + \begin{bmatrix} 5 & 8 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 8 & 14 \\ 10 & 12 \end{bmatrix}$$

Scalar-Matrix Product

주어진 행렬 A에 Scalar c 를 곱하면
모든 ij에 대해 $\mathbf{c} \times a_{ij}$ 로 구성된 행렬이 구성됨

$$A = \begin{bmatrix} 3 & 6 \\ 4 & 5 \end{bmatrix} \qquad c = 4$$

$$\mathbf{c} \times A = 4 \times \begin{bmatrix} 3 & 6 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 12 & 24 \\ 16 & 20 \end{bmatrix}$$

Matrix Transpose (전치 행렬)

- 주어진 $m \times n$ 의 행과 열을 바꾸어 만든 행렬
- 행렬 A의 전치 행렬은 A^T로 표시

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad A^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Matrix Product (행렬 곱셈)

- 앞 행렬의 열과 뒤 행렬의 행을 dot product
- 앞 행렬 열의 수와 뒤 행렬의 행 수가 동일 해야만 계산 가능

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 56 & \\ & \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad N = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad M \times N = ?$$

Human knowledge belongs to the world.

Linear algebra codes

Pythonic Code

최성철 교수
Director of TEAMLAB

Vector representation of python

- Vector를 파이썬으로 표시하는 다양한 방법 존재

```
vector_a = [1, 2, 10] # List로 표현했을 경우
vector_b= (1, 2, 10) # Tuple로 표현했을 경우
vector_c = {'x': 1, 'y': 1, 'z': 10} # dict 표현했을 경우

print(vector_a, vector_b, vector_c)
```

- 최선의 방법은 없음
- 값의 변경 유무, 속성값 유무에 따라 선택할 수 있음
- 본 수업에서는 기본적으로 list로 vector 연산 실시

Vector의 계산

```
u = [2, 2]
v = [2, 3]
z = [3, 5]
result = []
for i in range(len(u)):
    result.append(u[i] + v[i] + z[i])

print(result)
```

[2, 2]+[2, 3]+[3, 5]=[7, 10]

이런 코드는 쓰면 안됨
파이썬 답지 못하고
안 아름다움

Vector handling with python

- Python은 특유의 간결성이 최대의 장점
- Vector와 같은 수학 연산을 복잡하게 표현한다면 사용이 어려움
- 최대한 파이썬만의 특징을 살려서 간단하게 연산을 표시
- Comprehension과 zip 같은 pythonic technique을 적극 활용

https://goo.gl/ATmcYo

Vector의 계산

```
u = [2, 2]
v = [2, 3]
z = [3, 5]

result = [sum(t) for t in zip(u,v,z)]
print (result)
```

[2, 2]+[2, 3]+[3, 5]=[7, 10]

Vector의 계산: Scalar-Vector product

```
u = [1, 2, 3]
v = [4, 4, 4]
alpha = 2

result = [alpha*sum(t) for t in zip(u,v)]
print(result)
```

$$2([1,2,3]+[4,4,4]) = 2[5,6,7] = [10,12,14]$$

Matrix representation of python

- Matrix 역시 Python으로 표시하는 다양한 방법이 존재

```
matrix_a = [[3, 6], [4, 5]] # List로 표현했을 경우
matrix_b = [(3, 6), (4, 5)] # Tuple로 표현했을 경우
matrix_c = {(0 ,0): 3, (0 ,1): 6, (1 ,0): 4, (1 ,1): 5} # dict 표현했을 경우
```

- 특히 dict로 표현할 때는 무궁무진한 방법이 있음

- 본 수업에서는 기본적으로 two-dimensional list 형태로 표현함

- [[1번째 row], [2번째 row], [3번째 row]]

Matrix의 계산: Matrix addition

$$C = A + B = \begin{bmatrix} 3 & 6 \\ 4 & 5 \end{bmatrix} + \begin{bmatrix} 5 & 8 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 8 & 14 \\ 10 & 12 \end{bmatrix}$$

```
matrix_a = [[3, 6], [4, 5]]
matrix_b = [[5, 8], [6, 7]]
result = [[sum(row) for row in zip(*t)] for t in zip(matrix_a, matrix_b)]

print(result)
```

Matrix의 계산: Scalar-Matrix Product

$$\alpha \times A = 4 \times \begin{bmatrix} 3 & 6 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 12 & 24 \\ 16 & 20 \end{bmatrix}$$

```
matrix_a = [[3, 6], [4, 5]]
alpha = 4
result = [[alpha * element for element in t] for t in matrix_a]

print(result)
```

Matrix의 계산: Matrix Transpose

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
matrix_a = [[1, 2, 3], [4, 5, 6]]
result = [ [element for element in t] for t in zip(*matrix_a) ]
print (result)
```

Matrix의 계산: Matrix Product

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{4} & \mathbf{5} & \mathbf{6} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \mathbf{1} & \mathbf{4} \\ \mathbf{2} & \mathbf{5} \\ \mathbf{3} & \mathbf{6} \end{bmatrix} \text{ 이면 } \quad C = A \times B = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 5 & 8 \\ 5 & 6 \end{bmatrix}$$

```
matrix_a = [[1, 1, 2], [2, 1, 1]]
matrix_b = [[1, 1], [2, 1], [1, 3]]
result = [[sum(a * b for a, b in zip(row_a, column_b)) ₩
               for column_b in zip(*matrix_b)] for row_a in matrix_a]

print(result)
```

두 개이상의
Argument가 존재할 때는?

Human knowledge belongs to the world.