# 파이썬 실습(1)

## 중복되는 원소로 구성된 튜플 출력하기

- 예를 들어 (9,2,4,5)인 경우
  - {{9},{9,2},{9,2,4},{9,2,4,5}} 로 표현할 수 있음
  - 집합의 원소 순서는 바뀌어도 상관 없음, 즉 {{9,2,4,5},{9,2},{9},{9,2,4}} 로 표현되어도 모두 같은 튜플인 (9,2,4,5)를 나타냄
- 특정 튜플을 표현하는 집합인 담긴 문자열이 매개 변수로 주어질 때(ts), ts가 표현하는 튜플을 리스트에 담아 반환하는 함수를 작성
- 함수를 호출하는 메인코드까지 작성할 것

```
if __name__ == "__main__":
    ts = input("중복없는 튜플의 집합 형태 입력 >> ")
    print(tuple_ss(ts))
```

# 파이썬 실습(2)

- 영어의 끝말잇기 게임 프로그램 작성
  - 리스트로 주어진 영어 단어집 작성
    - 모두 소문자로 작성
    - 단어집의 단어 개수는 3이상 50이하로 구성
    - 단어의 길이는 2이상 30이하로 작성
  - 참여자 수 입력(2 ~ 5명)
  - 몇 번째 참여자가 몇 번째 차례에서 탈락하는지를 출력으로 함
  - 끝말잇기의 올바른 단어가 아니거나, 앞선 사람이 이미 말한 단어의 중복일 경우 탈락이며, 주어진 단어집의 끝까지 탈락자가 없을 경우는 [0,0]을 반환함

# 파이썬 실습(3)

- **2**차원 행렬 **A**와 **B**를 입력 받아 두 행렬의 곱을 구하는 함수를 작성하고 메인 코드까지 작성
  - 행렬의 곱은 내적의 곱을 의미하며
  - A*B 의 경우 A의 열의 사이즈와 B의 행의 사이즈가 동일해야 곱을 할 수 있음
  - A와 B 행렬을 구성하는 원소는 -10이상 20이하인 자연수로 구성
  - A와 B 행렬의 행과 열의 길이는 3이상 5이하로 할 것
- 행렬의 곱의 결과물의 **Transpose Matrix** 코드 작성
  - 전치행렬은 행과 열이 바뀐 행렬을 의미함
  - [[1,2],[3,4]] 의 전치행렬은 [[1,3],[2,4]] 가 됨

# 파이썬 데이터처리 및 분석

## Numpy 모듈

**2020.07. 15.** 수요일

최회련

En-CORE

Data Science Edu.

# 데이터 분석 분야의 수학

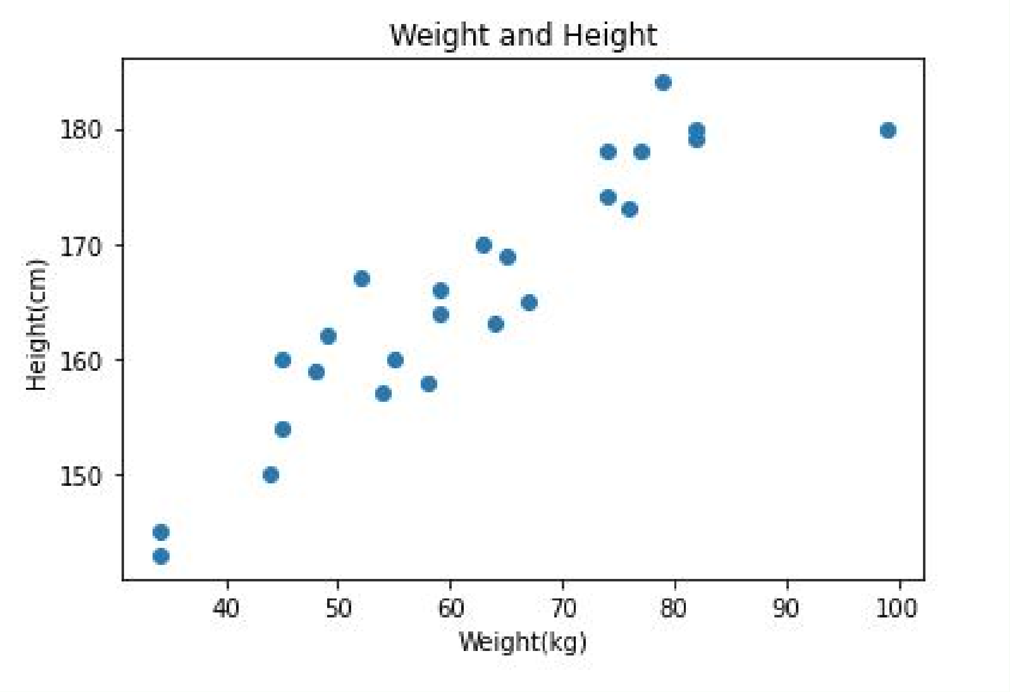
- **확률/통계 및 선형대수학(linear algebra)**
- 선형대수학의 필요성
  - 데이터를 벡터(vector)로 표현
  - 벡터 모임인 행렬을 통한 방정식 연산의 빠른 수행
- 데이터를 벡터로 표현
  - 데이터들을 표현하는 특징(feature)을 선형대수학의 기본 단위인 벡터로 표현
    - A 지역의 강수량 변화 - 연도, 월, 강수량의 데이터 표현
    - 영화 선호도 – B영화의 평점, C 영화의 평점, D 영화의 평점 데이터 표현
    - 영화 데이터 표현 – 장르, 감독, 출연배우, 개봉일 등
- 데이터 표현식을 찾기 위해 필요
  - 좌표계를 이용하여 데이터들을 표현하는 식을 찾는데 유용
  - 데이터를 가장 잘 표현하는 확률분포의 파라미터들을 축으로 좌표계를 생성 후 오차를 축소시키는 방향으로 이동

# 데이터 분석 분야의 수학

## ■ 행렬 연산의 필요성

- 예측을 위한 회귀분석
  - 몸무게와 키와의 관계 → 몸무게가 주어지면 키값의 예측 가능

| | weight | height |
|---|---|---|
| 0 | 44 | 150 |
| 1 | 54 | 157 |
| 2 | 67 | 165 |
| 3 | 34 | 145 |
| 4 | 59 | 166 |

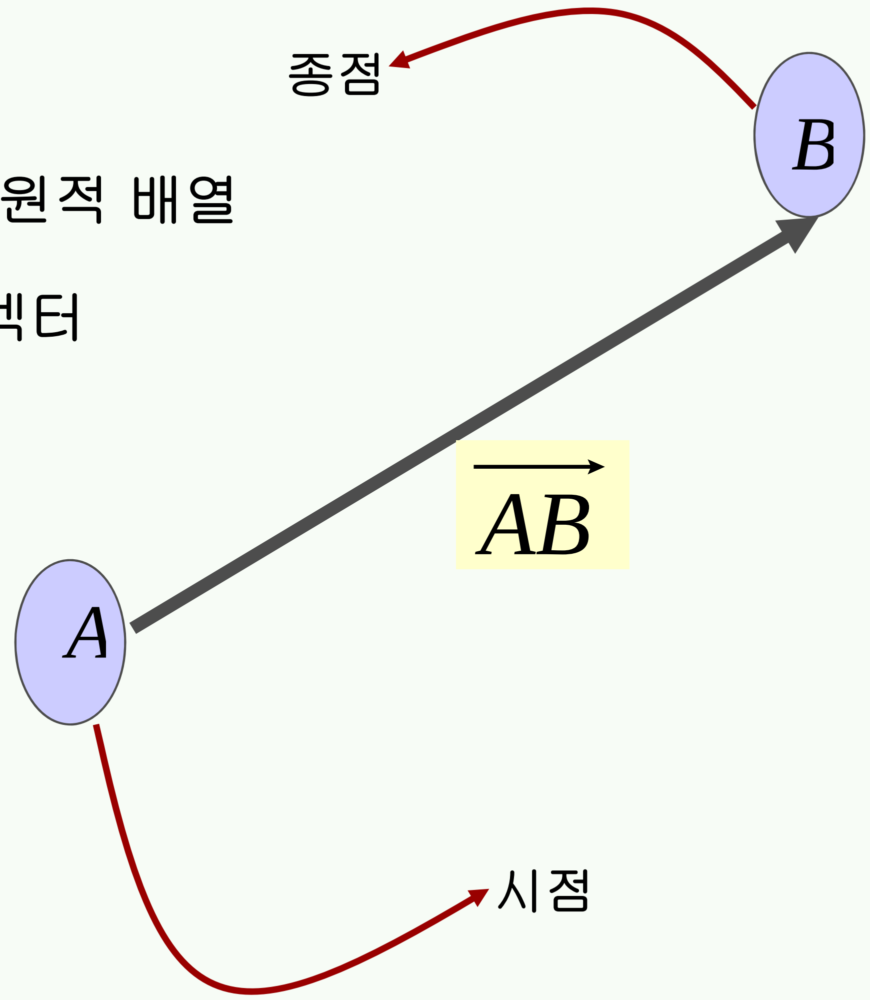


# 벡터(vector)

- 크기와 방향을 가진 요소
  - 예) 속도, 평행이동, 힘의 작용, ...
- **벡터(vector)**는 요소(수 또는 변수)들의 **1**차원적 배열
- 벡터는 일종의 순서쌍, **a**와 **b** 는 서로 다른 벡터

$$\vec{\boldsymbol{a}} = (\mathbf{1}, \mathbf{3}, -\mathbf{1})$$

$$\vec{\boldsymbol{b}} = (\mathbf{1}, -\mathbf{1}, \mathbf{3})$$

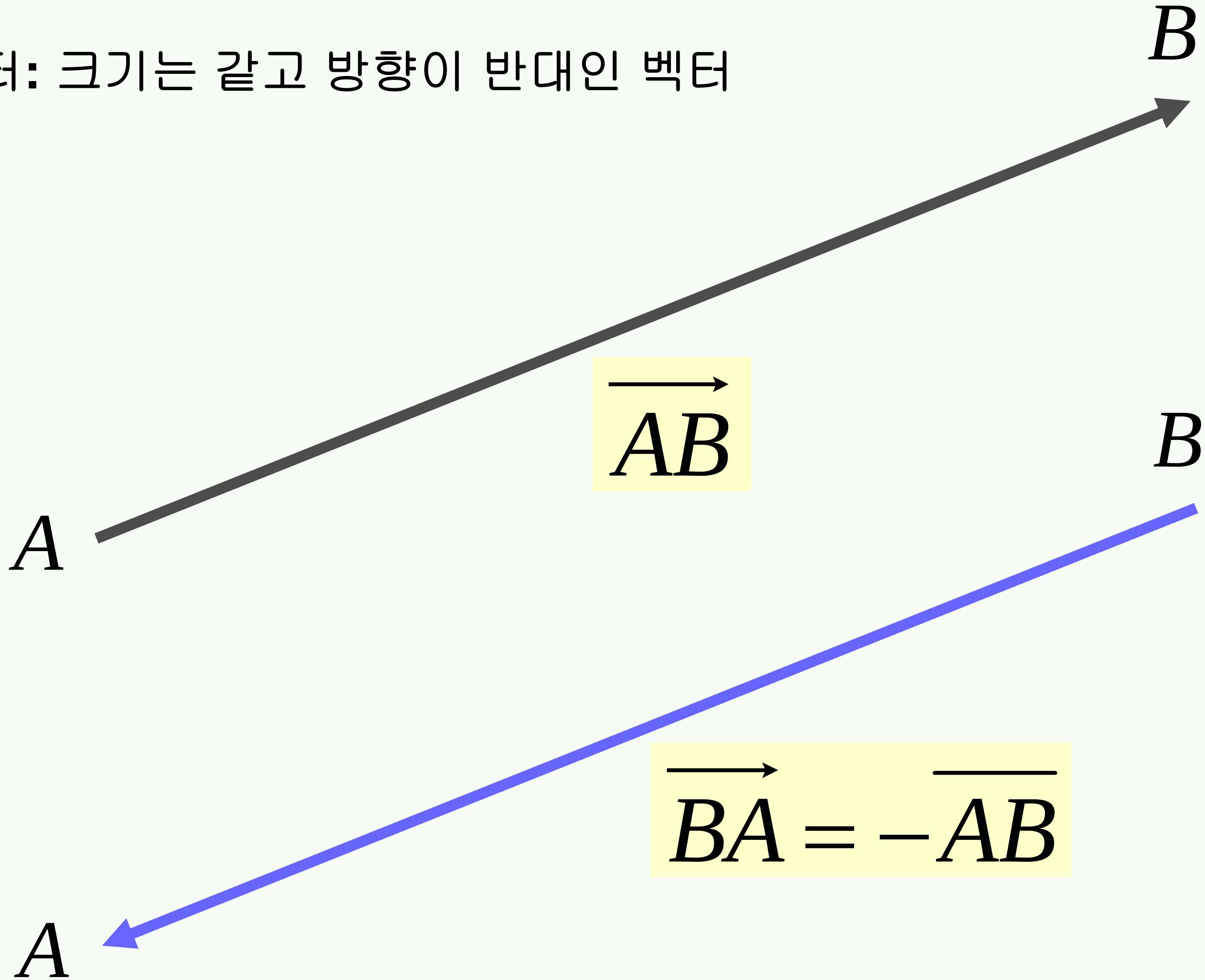



**cf)** 스칼라**(scalar)**

- 크기만을 가진 요소
- 예) 속력, 무게, 길이, …

# 벡터(vector)

- 역벡터: 크기는 같고 방향이 반대인 벡터




$$\overrightarrow{AB}$$

$$\overrightarrow{BA} = -\overline{AB}$$

# 벡터(vector)

- 벡터(**vector**)는 요소(수 또는 변수)들의 **1**차원적 배열로 정의 되며
- 열벡터(**column vector**): 세로로 배열된 벡터

$$\begin{bmatrix} 1 \\ x \end{bmatrix} \quad \begin{bmatrix} 2 \\ y \end{bmatrix} \quad \begin{bmatrix} 3 \\ z \end{bmatrix}$$

- 행벡터(**row vector**): 가로로 배열된 벡터

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

# 전치(transpose)

- 한 열벡터 → 행벡터로 변형

$$v = \begin{bmatrix} 1 \\ x \end{bmatrix} \quad \Rightarrow \quad v^T = [\,1, \quad x\,]$$

# 벡터의 연산(1)

## 벡터의 덧셈과 뺄셈

두 벡터 $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ 와 $y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$ 의 덧셈과 뺄셈 은 $x \pm y = \begin{bmatrix} x_1 & \pm & y_1 \\ & \vdots & \\ x_n & \pm & y_n \end{bmatrix}$

## 벡터의 동일성

- 동일 위치에 있는 요소들이 모두 같은 경우
- 두 벡터의 차는 영벡터(zero vector)

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 2 & - & y_1 \\ 3 & - & y_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

# 벡터의 연산(2)

- 벡터에 대한 스칼라 곱

벡터 $x = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$ 에 스칼라 $k$ 를 곱하면 $kx = k\begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2k \\ 3k \end{bmatrix}$

- 벡터 간의 곱셈: 벡터간의 내적(**inner product**) → 결과는 스칼라

두 벡터 $x = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}$ 와 $y = \begin{bmatrix} y_1 & \cdots & y_2 \end{bmatrix}$ 의 내적 $x \cdot y$ 는

$$x \cdot y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

# 벡터의 연산(3)

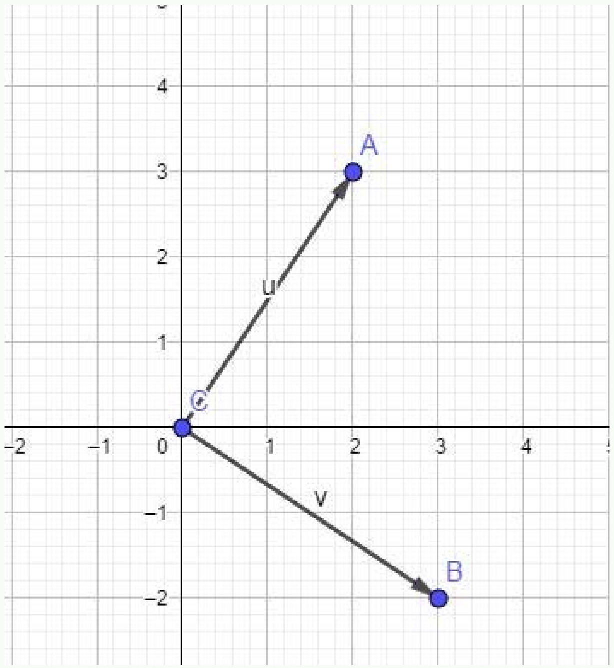
- $\boldsymbol{k}(\vec{\boldsymbol{a}}+\vec{\boldsymbol{b}})=\boldsymbol{k}\vec{\boldsymbol{a}}+\boldsymbol{k}\vec{\boldsymbol{b}}$ 임을 증명**, a** 와 **b** 는 **3**차원**,** 즉 열의 개수가 **3**개

# 벡터의 연산(4)

- 벡터의 내적 예
  - 세 과목의 점수는 (80,97,88) 이고, 각 과목의 성적 반영률이 (0.3,0.4,0.3)으로 주어진 경우의 최종 점수는 얼마인가?

# 벡터와 좌표

- **P(2,3)과 O(0,0), Q(3,-2)**




좌표평면 작성 무료 툴
https://geogebra.softonic.kr/download

# 벡터와 함수

- 두 이차함수의 연산

$$f(x) = -2x^2 + 3x + 1, \qquad g(x) = 3x^2 - x + 2$$

- **f(x) + g(x)**를 연산하면**?**

$$f(x) + g(x) = -2x^2 + 3x + 1 + 3x^2 - x + 2$$

$$= \quad (-2+3)x^2 + (3-1)x + (1+2) = x^2 + 2x + 3$$

- 즉, $\vec{a} = (-2, 3, 1)$, $\vec{b} = (3, -1, 2)$으로 표현 할 수 있음

# 행렬(matrix) 기본개념

## 행렬(matrix)- 01

- 요소(수 또는 변수)들이 2차원적으로 배열된 것
- 요소는 행렬 내에서의 위치를 하첨자로 부기함

• 행렬의 크기
  • 행의 수와 열의 수로 정해짐
•B행렬의 크기 → 2×3(2 by 3)
•행렬 $A$ 는 $A_{m\times n}$ 또는 $[a]_{m\times n}$ 표현

- 행렬: $B = \begin{bmatrix} 1 & 2 & 3 \\ x & y & z \end{bmatrix}$
- 요소: $b_{13} = 3$
- 행렬의 일반적 표현: $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ & \vdots & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$

# 기본 개념(계속)

■ **행렬 - 02**

$m \times n$ 행렬 $A = [a_{ij}]$ 와 $r \times s$ 행렬 $B = [b_{ij}]$ 가 있을 때

*m*=*r*, *n*=*s* 이고 $1 \le i \le m$ , $1 \le j \le n$ 인 모든 *i*, *j* 에 대하여

$a_{ij} = b_{ij}$ 이면 *A* 와 *B* 는 같다고 함

***A*=*B*** 로 나타냄

# 행렬의 연산

- 행렬의 덧셈과 뺄셈

$m \times n$행렬 $A = [a_{ij}]$와 $B = [b_{ij}]$가 있을 때

- *A* 와 *B* 의 두 행렬의 합(matrix addition)

$$A + B = [a_{ij} + b_{ij}]$$

- A와 B의 두 행렬의 차(matrix subtraction)

$$A - B = [a_{ij} - b_{ij}]$$

- 실수(스칼라)의 곱

$m \times n$행렬 $A = [a_{ij}]$ 와 실수(스칼라) $k$ 가 있을 때

- *A* 와 *k* 의 스칼라곱(scalar multiplication)

$$Ak = kA = [ka_{ij}]$$

# 행렬의 연산(계속)

- **[예제]**

행렬 *A*, *B*, *C* 가 다음과 같을 때 *A*+*B*, *A*+*C*, *A*+2*B* 를 구하여라.

$$A = \begin{bmatrix} 1 & 2 \\ -2 & 5 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 4 \\ 6 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} -1 & 8 & 0 \\ 2 & 0 & 4 \end{bmatrix}$$

**[풀이]**

# EF시 교통국 사례 - 행렬의 덧셈

- **EF시의 구역: 구시가지(구역 1)와 신시가지(구역 2)**
  - 하루에 각 구역 간을 이동하는 인구 조사
  - 교통수단별로 이동인구 조사하고 구역 간 이동인구를 두 행렬 A, B로 표현
  - 행렬 A는 차량, B는 전철을 이용하는 이동인구를 나타냄
  - 이동수단에 관계없이 두 구역 간을 이동하는 총 인구수는 유형별로 얼마인가?
  - 구 시가지 내에서의 이동 인구는 얼마인가?

# EF시 교통국 사례 - 행렬의 덧셈

$$A = \begin{bmatrix} 20 & 4 \\ 5 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}$$

- 행렬 A
  - 차량을 이용하여 구역 2에서 구역 1로 이동하는 인구: 5만(행렬 A요소 중 $a_{21}$)
  - 차량을 이용하여 구역 1에서 구역 2로 이동하는 인구: 4만(행렬 A요소 중 $a_{12}$)

- 행렬 B
  - 전철로 구역 2 내부에서 이동하는 인구: 4만(행렬 A요소 중 $b_{22}$)

# 행렬의 연산(계속)

- 행렬의 곱(matrix multiplication)

$m \times n$ 행렬 $A = [a_{ij}]$ 와 $r \times s$ 행렬 $B = [b_{ij}]$가 있을 때

- $n$=$r$ 이면 $A$ 와 $B$ 의 행렬의 곱(matrix multiplication) 연산 가능 → $AB$로 나타냄

$$AB = [c_{ij}] \qquad m \times s \text{ 행렬}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

행렬의 곱은 벡터의 내적에서 출발 → 차원의 일치

벡터 **a** 와 벡터 **b**의 내적 → **a•b = $\mathbf{a_1b_1 + a_2b_2}$**

**•(dot)**를 ×의 형태로 변형

$$\begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = a_1 \times b_1 + a_2 \times b_2$$

# 행렬의 연산(계속)

- 예제

다음 두 행렬의 곱 *AB* 와 *BA* 를 구하여라.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

**[풀이]**

# 행렬의 연산(계속)

- 행렬 $A$, $B$, $C$ 는 행렬이고 $c$, $d$ 는 스칼라라고 할 때 행렬의 합과 스칼라 곱에 대한 행렬의 성질($O$는 행렬의 모든 원소가 **0**인 행렬)

(1) $A+B=B+A$

(2) $A+(B+C)=(A+B)+C$

(3) $A+O=A=O+A$

(4) $A+(-A)=O=(-A)+A$

(5) $(-1)A=-A$

(6) $c(A+B)=cA+cB$

(7) $(c+d)A=cA+dA$

(8) $(cd)A=c(dA)$

# 행렬의 연산(계속)

- 세 개의 행렬 $A$, $B$, $C$ 와 스칼라 $k$ 에 대하여 행렬의 곱이 갖는 성질

  (1) $(AB)C=A(BC)$

  (2) $A(B+C)=AB+AC$

  (3) $(B+C)A=BA+CA$

  (4) $k(AB)=(kA)B=A(kB)$

# 행렬의 연산(계속)

■ 예제

다음 선형방정식을 *AX=B* 의 형태로 나타내어라.

$$
\begin{aligned}
x_1 + 3x_2 + x_3 &= 1 \\
2x_1 - x_2 + x_3 &= 4 \\
x_2 - x_3 &= 2
\end{aligned}
$$

**[풀이]**

# 국내 초고속통신망 시장 사례

- 시장 주도 기업: **1, 2, 3** 세 기업
  - 행렬 R: 시장점유율 $R = [0.52, \quad 0.375, \quad 0.105]$
  - 행렬 P: 소비자들의 이동성향
  - 행렬 내의 요소 $p_{ij}$ 는 기업 i의 통신망을 사용하다가 다음 해에 기업 j의 통신망으로 바꿀 확률
  - 행렬 P에서 행의 합은 반드시 1이어야 하나 열의 합은 반드시 1일 필요 없음

$$P = \begin{bmatrix} 0.9 & 0.07 & 0.03 \\ 0.125 & 0.8 & 0.075 \\ 0.2 & 0.2 & 0.6 \end{bmatrix} = \begin{bmatrix} P^1 & P^2 & P^3 \end{bmatrix}$$

# 국내 초고속통신망 시장 사례

- 차년도 세 기업의 시장점유율은 각각 얼마인가?

$$\text{기업 1의 차년도 점유율} = 0.52 \times 0.9 + 0.375 \times 0.125 + 0.105 \times 0.2 = 0.536$$

이는 $R$과 $P^1$의 내적을 구한 것과 동일

$$\text{기업 1의 차년도 점유율} = RP^1 = 0.536$$

$$\therefore P(\text{차년도}) = \begin{bmatrix} RP^1 & RP^2 & RP^3 \end{bmatrix} = R\begin{bmatrix} P^1 & P^2 & P^3 \end{bmatrix} = RP$$

$$= \begin{bmatrix} 0.52 & 0.375 & 0.105 \end{bmatrix} \begin{bmatrix} 0.9 & 0.07 & 0.03 \\ 0.125 & 0.8 & 0.075 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$

$$= \begin{bmatrix} 0.536 & 0.357 & 0.107 \end{bmatrix}$$

# 여러 가지 행렬

- **영행렬(zero matrix)**
  - $m \times n$ 행렬 $A = [a_{ij}]$ 가 있을 때 모든 *i*, *j* 에 대하여 $a_{ij} = 0$ 인 행렬
  - *A*=*O* 라고 나타냄

$$O = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

# 여러 가지 행렬(계속)

- **정방행렬(square matrix)**
  - $m \times n$ 행렬 $A = [a_{ij}]$ 가 있을 때 *m=n* 인 행렬
  - $n \times n$ 행렬로 나타냄

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

# 여러 가지 행렬(계속)

- *n* 차 정방행렬
  - *n* 개의 행과 *n* 개의 열로 구성된 정방행렬
  - *n* 값: 행렬의 차수(order)
  - 대각원소(diagonal element)
    - 대각선상의 원소 $a_{11}, a_{22}, \ldots, a_{nn}$
  - 대각행렬(diagonal matrix)
    - 대각원소 이외의 모든 원소가 0인 행렬

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

# 여러 가지 행렬(계속)

- **단위행렬(unit matrix, identity matrix)**
  - 정방행렬이면서 대각원소가 모두 1이고 그 외의 모든 원소가 0인 대각행렬
  - 행렬곱의 항등행렬
  - *I* 로 나타냄

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

- 정방행렬 *A* 에 대한 단위행렬 *I* 의 성질

$$IA = A = AI$$

# 여러 가지 행렬(계속)

- **전치행렬(transpose matrix)**
  - $m \times n$행렬 $A = [a_{ij}]$ 가 있을 때 *A* 의 행과 열을 서로 바꾼 행렬
  - $A^T$로 나타냄
- **대칭행렬(symmetric matrix)**
  - $A^T=A$ 인 행렬
- $m \times n$행렬 *A*, *B* 에 대한 전치행렬의 성질

$$(1)(A^T)^T = A$$

$$(2)(A+B)^T = A^T + B^T$$

$$(3)(kA)^T = kA^T$$

$$(4)(AB)^T = B^T A^T$$

# 여러 가지 행렬(계속)

■ 예제

다음 행렬에 대한 전치행렬을 구하고 대칭행렬인지 판별하여라.

(1)

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

(2)

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

(3)

$$A = \begin{bmatrix} 5 & 0 & 2 & -6 \\ 0 & 4 & 3 & 1 \\ 2 & 3 & 1 & 4 \\ -6 & 1 & 4 & 2 \end{bmatrix}$$

# 여러 가지 행렬(계속)

## 역행렬

- 임의의 정방행렬 $A_{n\times n}$에 대하여 정방행렬 $B_{n\times n}$ 가 다음의 식을 만족하면,

$$AB = BA = I_{n\times n}$$

- B를 A의 역행렬이라 하고, $A^{-1}$로 표기한다.
- 행렬곱의 역원

## 역행렬의 성질

$$AA^{-1} = A^{-1}A = I$$

$$(A^{-1})^{-1} = A \qquad (AB)^{-1} = B^{-1}A^{-1}$$

$$(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$$

$$(A^n)^{-1} = (A^{-1})^n \qquad (kA)^{-1} = \frac{1}{k}A^{-1}$$

# 행렬식(Determinant)

- 원래 연립**1**차방정식의 해를 구하기 위해 라이프니쯔**(Leibnitx,G.W.)**에 의해 시작
- 크래머**(cramer)**에 의해 완전한 체계를 갖춤
- 다변수연립방정식의 해를 일괄적으로 풀 수 있다는 장점
- 기하학에서는 넓이와 부피를 구하는데 사용
- 또한 역행렬의 유무 판별에도 사용
- **2,3**차 이상의 행렬식은 라플라스**(**또는 여인수전개**)** 또는 크래머방식을 이용하여 구함
- **2**차 → **A=** $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ **=> det(A) =** $ad - bc$

# 행렬식

- **3차 B=** $\begin{bmatrix} a1 & a2 & a3 \\ b1 & b2 & b3 \\ c1 & c2 & c3 \end{bmatrix}$ **=> det(B)**

  **= a1b2c3+b1c2a3+c1b3a3 - a3b2c1 - b3c2a1 - c3a2b1**

- 역행렬이 존재하지 않는 연립방정식은 해가 없거나, 무수히 많은 경우에 해당
  - 두 식이 같으면 (배수포함) 해가 무수히 많음을 의미
- 다음 연립일차방정식이 주어졌을 때, **x=y=0** 이외의 해를 가진다고 했을 때, **a**의 값은?

$$\begin{pmatrix} 2 & 3 \\ 4 & a \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

# 연습문제 1

- 하이야트 맥주주식회사의 새로운 양조장 입지 선정 문제
  - 두 후보지 A, B에 대한 평가척도는 다음 표와 같음
  - 각 척도별 평가점수는 10점 만점

| 평가척도 | 가중치(%) | 평가점수 | |
|---|---|---|---|
| | | A | B |
| 건설비용 | 10 | 8 | 5 |
| 사회기반시설 | 10 | 7 | 7 |
| 사업여건 | 10 | 4 | 7 |
| 부동산 가격 | 20 | 7 | 4 |
| 삶의 질 | 20 | 4 | 8 |
| 교통 편의성 | 30 | 7 | 6 |

# 연습문제 1

- 평가척도의 가중치를 고려할 때 두 후보지 중 어느 곳이 더 바람직한가?

| 평가척도 | 가중치(%) | 평가점수 | | 가중치 평가점수 | |
|---|---|---|---|---|---|
| | | A | B | A | B |
| 건설비용 | 10 | 8 | 5 | 80 | 50 |
| 사회기반시설 | 10 | 7 | 7 | 70 | 70 |
| 사업여건 | 10 | 4 | 7 | 40 | 70 |
| 부동산 가격 | 20 | 7 | 4 | 140 | 80 |
| 삶의 질 | 20 | 4 | 8 | 80 | 160 |
| 교통 편의성 | 30 | 7 | 6 | 210 | 180 |
| 합계 | 100 | 37 | 37 | 620 | 610 |

# 연습문제 1

- 벡터의 내적 개념을 이용하여 대안별 가중평가점수의 합계를 구하는 식 기술

# 연습문제 2

- 다음 행렬의 역행렬을 구하시오

$$A = \begin{bmatrix} 2 & 3 \\ 0 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 1 \\ -1 & 0 \end{bmatrix}$$

# 연습문제3

- 연립일차방정식이 다음과 같이 주어졌을 때, **x=y=0** 이외의 해를 가지는 경우의 람다**(λ)** 의 값을 구하시오

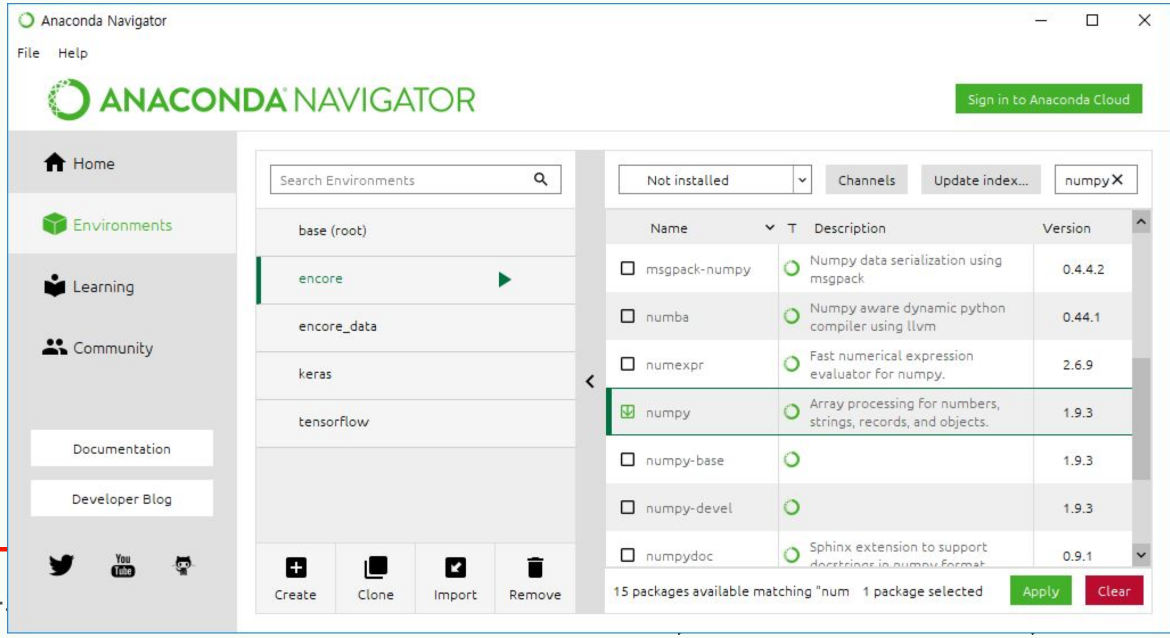
$$\begin{pmatrix} \mathbf{3} & \mathbf{1} \\ \mathbf{1} & \mathbf{3} \end{pmatrix}\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{\lambda}\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \rightarrow \mathbf{Ax} = \boldsymbol{\lambda}\mathbf{x}$$

# 고유값과 고유벡터(eigenvalue and eigenvector)

- **Ax = λx** 에서 이 등식을 만족하는 특별한 상수 **λ**를 행렬 **A**의 고유값이라 함
- 열벡터 **x**를 고유값 **λ**에 대응하는 고유벡터라고 함
- 고유값이 같은 벡터들은 서로 실수배 차이밖에 나지 않으므로 같은 벡터로 취급 함

# NumPy 설치

- **Numpy 모듈이 설치가 안되어 있는 경우**
  - Command 창에서 가상 환경으로 activate 한 후 pip install numpy 를 실행 (pip3 install numpy)
  - Anaconda Navigator 실행 후, [Environments] 실행
    - 생성시킨 가상환경 중 작업할 환경을 선택
    - 오른쪽에 이미 설치된 패키지 및 모듈의 목록이 보여짐
    - 목록에 없는 패키지 및 모듈은 상단의 Not Installed 에서 검색 후
    - 목록을 선택 후 [Apply] 를 클릭 함



# NumPy 패키지

- 파이썬의 리스트는 원하는 타입들을 채울 수 있는 유연성을 가지는 대신, 저장공간 낭비 등이 발생
- **NumPy**의 배열은 유연성은 부족하지만, 데이터 저장과 가공에 효율적
- 계산 또는 분석에 이용되어지는 라이브러리, 선형대수의 기본 원리
- 다차원의 수치까지 다룰 수 있음
- **list** 보다 빠른 계산 구조를 가진 **array** 생성
- **array** 형태이므로, 구성되는 값들은 모두 같은 자료형으로 구성

파이썬의 패키지 설치 위치
**C:\Python\Python37-32\Lib\site-packages**

# NumPy 기본

- **list 와 array 의 차이**

```
import numpy as np

list1 = [1,2,3]
print("list*2 : ", list1*2)
ary1 = np.array([1,2,3])
print("array*2 : ", ary1*2)
```

**list*2 : [1, 2, 3, 1, 2, 3]**
**array*2 : [2 4 6]**

```
import numpy as np

list1 = [1,2,3,'b']
print(list1)
ary1 = np.array([1,2,3,'b'])
print(ary1)
ary2 = np.array([1,2, 2.3 ,3])
print(ary2)
```

```
[1 2 3 'b']
['1' '2' '3' 'b']
[1. 2. 2.3 3.]
```

# NumPy 기본

- **Array 생성**

```
import numpy as np
a1 = np.array([1,2,3]) #벡터형식
print(a1, a1.shape, a1.ndim)
print(type(a1))
print(a1[0], a1[1], a1[2])
print()
a2 = np.array([[1,2,3],[4,5,6]]) #배열형식
print(a2, a2.shape, a2.ndim)
print(a2[0],a2[1])
print(a2[0,0],a2[0][1],a2[0,2])
```

# NumPy 기본

■ 데이터 기본 타입

| 데이터 타입 | 설명 |
|---|---|
| **bool_** | **1**바이트의 참 또는 거짓 |
| **int8** | **1byte** 정수 |
| **int16, int32, int64** | **2,4,8 byte** 정수 |
| **uint8, uint16,uint32,uint64** | **1,2,4,8 byte** 양의 정수 |
| **float16, float32, float64** | **2,4,8 byte** 실수 |
| **complex64, complex128** | **8, 16 byte** 복소수 |

```
import numpy as np

ary1 = np.zeros(10, dtype=np.float32)
ary2 = np.array([range(i,i+2) for i in range(2,5)], dtype='int32')
ary3 =  np.array([range(i, i+4) for i in range(2,5)],dtype='float32')
print(ary1)
print(ary2)
print(ary3)
```

# NumPy 기본

- zeros, ones, full, eye, linspace

```
import numpy as np
a = np.zeros((2,2));print(a)
b = np.ones((3,3));print(b)
c = np.full((2,3), 5)
print(c)
d = np.eye(3) #단위행렬 생성
print(d)
e = np.linspace(0,1,5) #0~1사이에서 균등하게 5개 생성
print(e)
f = np.arange(10) # 0 ~9까지의 1씩 증가하는 정수 생성
print(f)                # np.arange(10, dtype='float32') ??
```

```
import numpy as np
a = np.array([[1,2,3],[4,5,6]]);print(a)
b = np.ones_like(a); print(b)
c = np.full_like(a, 5);print(c)
d = np.zeros_like(a);print(d)
```

# NumPy 기본

## NumPy 의 랜덤함수 사용

- random, normal, randint ,randn,exponential,uniform

```
import numpy as np

ary1 = np.random.exponential(6, 2)#mean(scale)=6,size=2
ary2 = np.random.random((3,3)) #0~1사이의 실수 균등분포, 3 by 3
ary3 = np.random.normal(2,1,(3,2)) #평균:2,편차:1,정규분포 3by2
ary4 = np.random.randn(2,3)  #표준정규분포, 평균:0, 편차:1
ary5 = np.random.randint(0,10,(2,2))
ary6 = np.random.uniform(1,5,2) # 1 ~ 5사이의 uniform 분포
print(ary1)
print(ary2)
print(ary3)
print(ary4)
print(ary5)
print(ary6)
```

# NumPy 기본

- **random.seed**

```
import numpy as np

ary1 = np.random.randint(1,10, 5)
ary2 = np.random.poisson(lam=2.0, size=5)

print(ary1)
print(ary2)
```

```
import numpy as np

np.random.seed(10)
ary1 = np.random.randint(1,10,5)
ary2 = np.random.poisson(lam=2.0,size=5)

print(ary1)
print(ary2)
```

# NumPy 기본

- 평균 **0,** 편차 **1**인 정규분포값을 **10000** 발생 후 도표로 확인

```
import numpy as np
import matplotlib.pyplot as plt
#matplotlib inline
data = np.random.normal(0, 1, 10000)
plt.hist(data, alpha=0.5, bins = 100)#bins:나누기양, 즉, 막대기 수
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
data1 = np.random.normal(2,1, 10000)
data2 = np.random.normal(0,1,10000)
plt.hist(data1, bins=50, color="green")
plt.hist(data2, alpha=0.5, bins = 50)
plt.show()
```

**matplotlib 설치 - pip install matplotlib**
**tutorial site::**
**https://matplotlib.org/tutorials/index.html**
**Jupyter Notebook 사용자는 import 작성 후**
**%matplotlib inline 의 코드를 추가 작성**

# Matplotlib.pyplot

- **Sigmoid Relu function의**

  분포 그리기

```
import numpy as np
import matplotlib.pyplot as plt

def sigmoid_f(x):
    return 1 / (1 + np.exp(-x))

def relu_f(x):
    return np.maximum(0, x)

x = np.arange(-5.0, 5.0, 0.1)
y_relu = relu_f(x)
plt.plot(x, y_relu)
plt.show()

y_sigmoid = sigmoid_f(x)
plt.plot(x,y_sigmoid)
plt.show()
```

# NumPy 기본

```
import numpy as np

ary1 = np.array([1,2,2,3,4,4,5,6,7,8,8,8])
np.random.shuffle(ary1) # 랜덤하게 섞기

ary2 = np.random.choice(ary1,3)
#ary2 = np.random.choice(ary1,3,
         p=[0.2,0.05,0.05,0.1,0.1,0.1,0.1,0.1,0.05,0.05,0.05,0.05])
#choice(선택값, 크기, replace=True, p=각데이터가 선택될 수 있는 확률)

ary3 = np.unique(ary1) # 중복 제거

print(ary1)
print(ary2)
print(ary3)
```

# NumPy 기본

- **Array indexing: slicing** **a[start:stop:step]**

```
import numpy as np
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
b = a # b = a[:] 동일함
a[0,0] = 100 # a[0][0] = 100
print(a)
print(b)
```

**Cf) b = a.copy() 인 경우?**

```
import numpy as np
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
b = a[:2,1:3] # a[ : , :] 첫자리는 행, 두번째는 열
a[0,1] = 100
b[0,1 ] = 300
print(a, a.ndim)
print(b, b.ndim)
```

# NumPy 기본

## a[ , ]과 a[ : , : ] 차이?

```
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

row_r1 = a[1, :]
row_r2 = a[1:2,:]
print(row_r1, row_r1.shape)
print(row_r2, row_r2.shape)
print()

temp = a[1:, :3]
imsi = a[1:2,2]
print(imsi)
print(temp)
```

```
import numpy as np
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

col_r1 = a[:, 1]
col_r2 = a[:,1:2]
print(col_r1, col_r1.shape, col_r1.ndim)
print(col_r2, col_r2.shape, col_r2.ndim)
```

# NumPy 기본

- **Integer array indexing**

```
import numpy as np
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

print(a[[0,1,2],[0,1,0]]) #a[0,0],a[1,1],a[2,0]

b = np.array([a[0,0],a[1,1],a[2,0]])
print(b, b.shape)

print(a[[0,0],[1,1]]) #a[0,1],a[0,1]
```

# NumPy 기본

```
import numpy as np
a = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
print(a)
print()
b = np.array([0,2,0,1])

c = a[np.arange(4),b] # a[0,0], a[1,2], a[2,0],a[3,1]
print(c, c.shape)
print()

a[np.arange(4),b] += 10
print(a)
```

# NumPy 기본

- **Boolean array indexing**

```
import numpy as np
a = np.array([[1,2],[3,4],[5,6]])

bool_index = (a>2)
print(bool_index)

print(a[bool_index])
print(a[a>2])

even_a = (a % 2 == 0)
print(even_a)

print(a[even_a])
print(a[a%2==0])
```

# NumPy 기본

- **Data type**

```
import numpy as np
a = np.array([3,4,5])
print(a.dtype)
b = np.array([3.0,4.0])
print(b.dtype)

c = np.array([1,2], dtype=np.float32)
print(c.dtype)
print(c)
```

# quiz

- 다음 프로그램의 결과는?

```
import numpy as np
a = np.array([[1,2,3,4],
              [5,6,7,8],
              [9,10,11,12]])
b = a[1:, ::2]
print(b, b.ndim,b.shape)
b[1,0] = 88
print(a)
```

```
import numpy as np
a = np.array([[1,2,3,4],
              [5,6,7,8],
              [9,10,11,12]])
b = a[ ::-1, ::-1]
print(b, b.ndim,b.shape)
```

## 배열의 재구조화

- reshape 사용

```
import numpy as np

ary1 = np.arange(1,10) # range 와 같은 역할
print(ary1, ary1.ndim)
ary2 = ary1.reshape((3,3))
print(ary2,ary2.ndim)

ary3 = np.array([[1,2,3,4],
                 [5,6,7,8]])
re_ary3 = ary3.reshape((4,2))
print(re_ary3)
```

# NumPy 기본

■ 배열의 연결 및 분할

- 연결: concatenate, vstack, hstack
- 분할: split, hsplit, vsplit

```
import numpy as np

ary1 = np.array([[1,2,3,4],
                 [5,6,7,8]])
ary2 = np.array([[10,11,12,13],
                 [14,15,16,17]])

ary3 = np.concatenate([ary1,ary2]) # 열의 수가 같아야 함,세로붙이기
ary4 = np.concatenate([ary1,ary2], axis=1)#행의 수가 같아야 함
print(ary3, ary3.shape)
print(ary4, ary4.shape)
```

# NumPy 기본

- math

```
import numpy as np
a = np.array([[1,2],[3,4]],dtype=np.float64)
b = np.array([[5,6],[7,8]],dtype=np.float64)

print(a + b)
print(np.add(a,b))
print()
print(a-b);print(np.subtract(a,b));print()
print(a*b);print(np.multiply(a,b));print()
print(a/b);print(np.divide(a,b));print()
print(np.prod(a,axis=0));print()
print(np.sqrt(a));print()
```

```
print(a.dot(b))
print(np.dot(a,b));print()
print(b.dot(a))
print(np.dot(b,a))

print(np.sum(a))#total sum
print(np.sum(a,axis=0))
print(np.sum(a,axis=1))

a_t = a.T
print(a_t)
```

# NumPy 기본

```
import numpy as np
a = np.array([[1,2],[3,4]],dtype=np.float64)

a_det = np.linalg.det(a) #행렬식계산
print(a_det);print()
a_inv = np.linalg.inv(a) #역행렬계산
print(a_inv);print()

#Ax = B의 방정식을 해결하는 메서드
a = np.array([[2,3],[4,1]])
b = np.array([[2],[5]])
x = np.linalg.solve(a,b)
print(x)
```

# NumPy 기본

- **broadcasting**
  - rule 1 : 두 배열의 차원 수가 다르면 작은 차원을 가진 배열의 형상 앞쪽을 1로
  - rule 2 : 두 배열의 형상이 모두 다르면, 1을 가진 배열이 다른 배열과 동일
  - rule 3 : 임의의 차원에서 크기가 일치하지않고 1도 아니면 오류

```
import numpy as np
a = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
b = np.array([1,0,1])
c = np.empty_like(a) # np.empty([4,3])
print(c)
for i in range(4):
    c[i,:] = a[i,:] + b

print(c)

x = np.array([1,2,3])
w = np.array([4,5])
print(np.reshape(x,(3,1))*w)
```

예를 들어,
**a:(3, 1), b: (3, )** 일 때,
**a + b** 의 계산을 위해
**rule 1**에 의해 **b** 가 **(1,3)**으로
**rule 2**에 의해 **a**가 **(3,3)**, **b**가 **(3,3)** 이 되어
계산되는 것을 **broadcasting** 이라 함

# 실습

- **2014**년 시애틀의 강수량 데이터를 활용

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

rainfall = pd.read_csv('Seattle2014.csv')['PRCP'].values
inches = rainfall/25.4 # 25.4mm == 1 inches
plt.hist(inches, 40)
plt.show()
print("Number days without rain: ", np.sum(inches==0))
print("Number days with rain: ", np.sum(inches!=0))
print("Days with more than 0.5 inches: ", np.sum(inches > 0.5))
print("Rainy days with < 0.1 inches : ", np.sum((inches >0) & (inches < 0.2)))
```