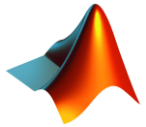
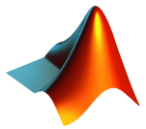
MATLAB 프로그래밍 및 실습

2강. MATLAB 기본문법

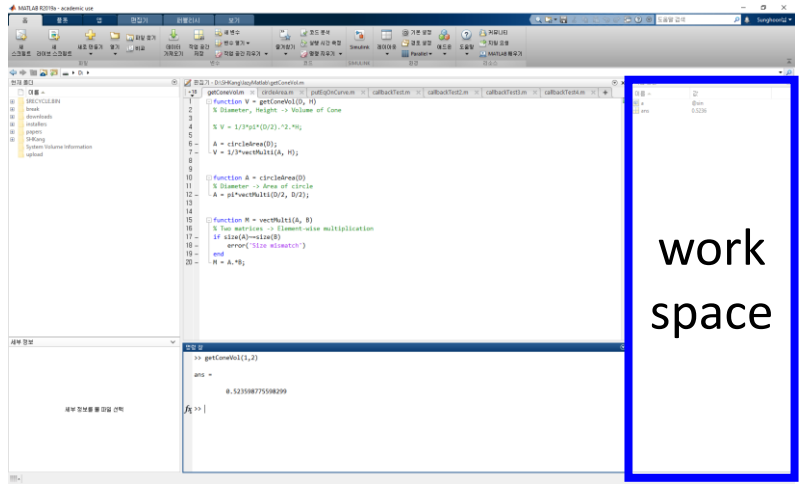


변수



변수란?

- 변수란?
  - 값을 저장하고, 저장된 값을 바꿀 수 있으며,
    이름으로 값을 읽어올 수 있는 프로그래밍의 기본 요소
- 변수명이란?
  - 변수가 저장된 메모리 공간에 붙은 이름
  - 변수명으로 변수에 저장된 값을 불러올 수 있음



작업공간 (workspace)
- 메모리 상에 올라와 있는 변수의 목록을 보여주는 공간

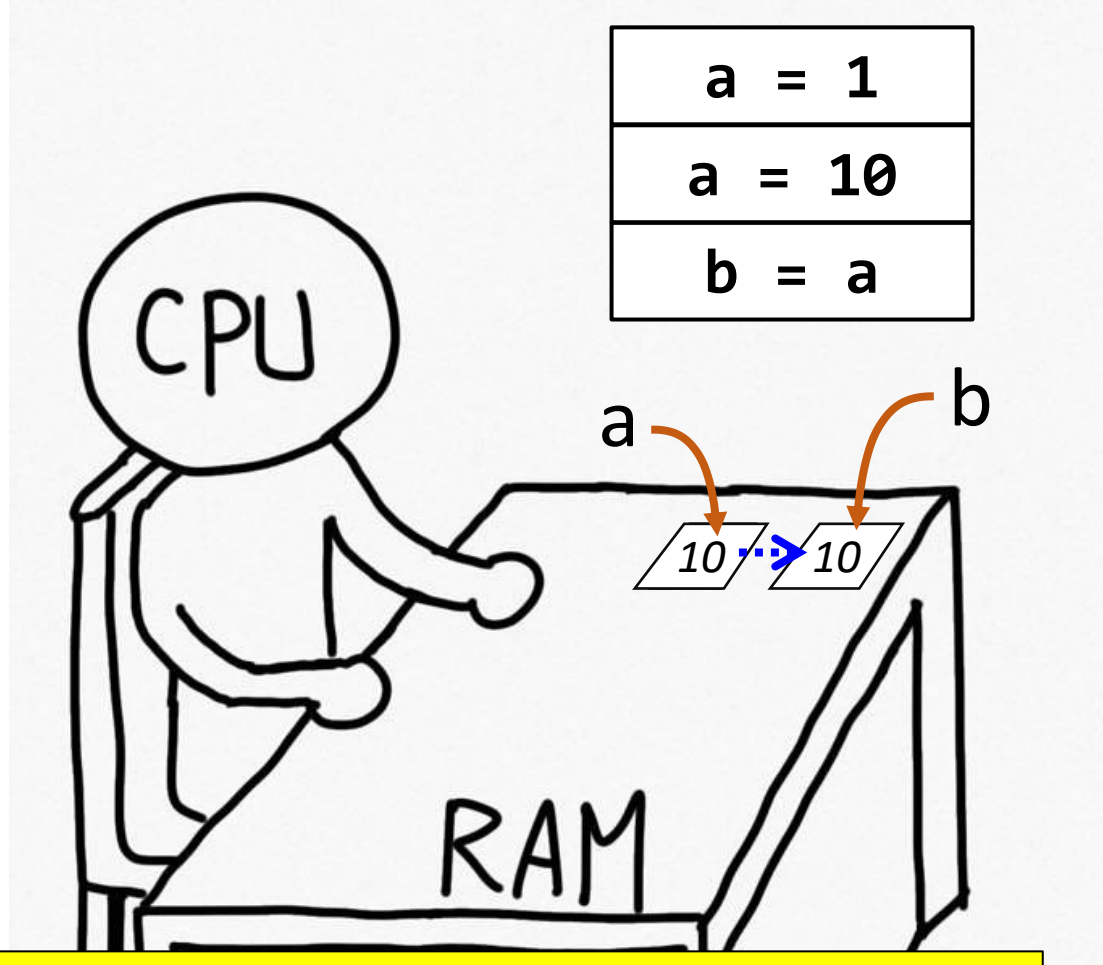
변수 **이름** → a = 1 ← **미리 "정의"된** 값, 표현식, 변수

(값이든 변수든 미리 **정의**된 것)

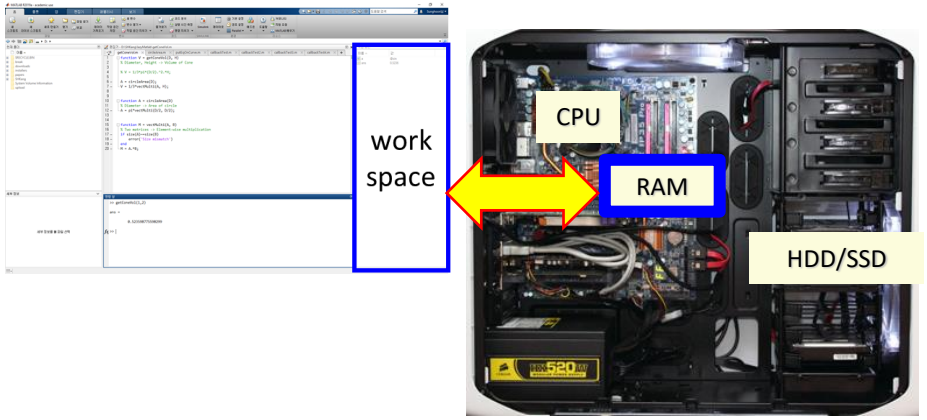
대입연산자 (수학의 등호가 아님!)

변수를 "정의"한다.

ex. a=a+1



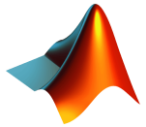
RAM (메모리)
- CPU가 현재 작업 중인 것들을 올려두는 공간
- CPU가 빠르게 접근 가능
- 컴퓨터의 **RAM** = 매트랩의 **"작업 공간"**
- CPU가 현재 작업중인 것 = 매트랩의 **"변수"**
- 컴퓨터 재부팅 → RAM 초기화
- 매트랩 종료 → 작업공간 비움



- 모든 것은 **정의**에서부터 시작된다.
- 변수명으로 변수 값을 가져올 수 있는 이유?
  → 변수 값이 메모리에 저장되고,
  그 메모리 공간에 이름(변수명)이 붙기 때문
- 정의한 적이 **있는** 변수명
  → 이름으로 접근 가능
- 정의한 적이 **없는** 변수명
  → "정의되지 않은 변수입니다."

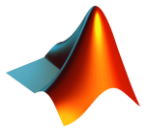
변수명 규칙

- 변수명은…
  - 최대 63글자
  - 영어, 숫자, 밑줄(_)만 허용
    - 첫 글자는 영어만 허용
    - 한글 X, 마침표 X, 쉼표 X, 콜론 X, 세미콜론 X, 빈칸 X
  - 매트랩 키워드 X
    - 키워드 확인: iskeyword
  - 대소문자 구분함
  - 내장 함수, 내장 상수, 내장 변수 등은 지양
    - ans, sqrt, pi, …
  - 이해하기 쉬운 이름으로 작성



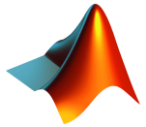
자주 사용하는 기본 함수 및 기능

- cd → 현재 폴더
- 현재 폴더를 바꾸는 3가지 방법
  1) 주소창에 붙여넣기
  2) 주소창과 현재 폴더 창을 탐색기처럼 이용
  3) cd 폴더명, 또는 cd('폴더명') → 폴더명에 빈칸 주의
- clc: 화면만 지울 뿐, 변수(workspace)를 지우지는 않음
- clear
  - clear x y z
  - clear x*
- 문(statement) 끝에 세미콜론이 있고 없음의 차이?
  - a = 1+2+3 vs a = 1+2+3;
  - 1+2+3; vs a = 1+2+3;

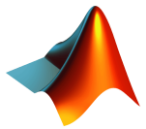


자주 사용하는 기본 함수 및 기능

- 명령 창 코드 중 일부만 실행하고 싶을 때
  - F9의 사용
  - 섹션 나누기 + 일부 섹션만 실행
- workspace 저장하기, 불러오기
  - 마우스 이용
  - load, save 함수로 하는 방법
  - 일부 변수만 저장하는 것도 가능
- format
  - short, short e, shorte
  - long, long e, longe
  - compact, loose



수식 작성



간단한 수식

- 사칙 연산
- 연산순서 (좌→우)
  1) 괄호
  2) 거듭제곱 (a^b → a^b)
  3) 곱하기, 나누기
  4) 더하기, 빼기

※ $x \times y$ 는 x*y
(xy는 이름이 xy인 변수)

```
>> x=15
x =
    15

>> x=3*x-12
x =
    33
>>
```

The number 15 is assigned to the variable x.

MATLAB displays the variable name and its assigned value.

A new value is assigned to x. The new value is 3 times the previous value of x minus 12.

```
>> a=12
a =
    12
>> B=4
B =
     4
>> C=(a-B)+40-a/B*10
C =
    18
```

Assign 12 to a.

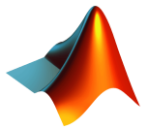
Assign 4 to B.

Assign the value of the expression on the right-hand side to the variable C.

```
>> a=12;
>> B=4;
>> C=(a-B)+40-a/B*10;
>> C
C =
    18
```

The variables a, B, and C are defined but are not displayed, since a semicolon is typed at the end of each statement.

The value of the variable C is displayed by typing the name of the variable.



좀 복잡한 수식

$$7 \times 3.1 + \frac{\sqrt{120}}{5} - 15^{5/3}$$

$$\left(\frac{1}{\sqrt{75}} + \frac{73}{3.1^3}\right)^{1/4} + 55 \times 0.41$$

$$B = P\left(1 + \frac{r}{n}\right)^{nt}$$

$$T = T_S + (T_0 - T_S)e^{-kt}$$

$$I_0 = I \cdot \ln\left(-\frac{Q}{RT}\right)$$

$$t' = \frac{t}{\sqrt{1 - \frac{v^2}{c^2}}}$$

$$F_g = \frac{Gm_1m_2}{r^2}$$

$$E = 1.74 \times 10^{19} \times 10^{1.44M}$$

$$V = \frac{\pi H}{3}\left(\frac{D}{2}\right)^2$$

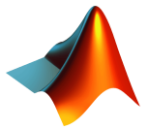
$$f = \frac{1}{\frac{1}{f_i} + \frac{1}{f_o}}$$

수식이 너무 길어지면?
1) …를 쓴다.
2) 파트를 나눈다.
3) 괄호를 아끼지 않는다.

자주 쓰는 내장함수

함수명	동작	예시	참고
sqrt	제곱근	>> sqrt(2) ans = 1.4142	
exp	e^x	>> exp(1) ans = 2.7183	exp^가 아님
abs	절대값	>> abs(-10) ans = 10	복소수 입력 가능
log	자연로그	>> log(exp(1)) ans = 1	
log10	$\log_{10}$	>> log10(100) ans = 2	

함수명	동작	예시	참고
round	반올림	>> round(pi) ans = 3	
ceil	올림	>> ceil(pi) ans = 4	
floor	버림	>> floor(-pi) ans = -4	
fix	0 방향으로 가장 가까운 정수	>> fix(-pi) ans = -3	양수: floor와 같음 음수: ceil과 같음
rem	나눗셈의 나머지	>> rem(10,3) ans = 1	
sign	부호	>> sign(pi) ans = 1	

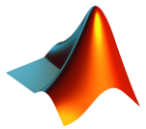


자주 쓰는 내장함수

※ 제곱표현 및 제곱 위치 주의 : $\sin^2(x)$

sin(x)	Finds the sine of **x** when **x** is expressed in radians.	**sin(0)** **ans = 0**
cos(x)	Finds the cosine of **x** when **x** is expressed in radians.	**cos(pi)** **ans = −1**
tan(x)	Finds the tangent of **x** when **x** is expressed in radians.	**tan(pi)** **ans =** **−1.2246** **e−016**
asin(x)	Finds the arcsine, or inverse sine, of **x**, where **x** must be between −1 and 1. The function returns an angle in radians between $\pi/2$ and $-\pi/2$.	**asin(−1)** **ans =** **−1.5708**
sinh(x)	Finds the hyperbolic sine of **x** when **x** is expressed in radians.	**sinh(pi)** **ans =** **11.5487**
asinh(x)	Finds the inverse hyperbolic sin of **x**.	**asinh(1)** **ans =** **0.8814**
sind(x)	Finds the sin of **x** when **x** is expressed in degrees.	**sind(90)** **ans =** **1**
asind(x)	Finds the inverse sin of **x** and reports the result in degrees.	**asind(1)** **ans =** **90**

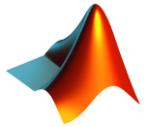
tip.
deg2rad: degree → radian
rad2deg: radian → degree



자주 쓰는 내장함수

함수명	동작	예시	참고
i, j	허수 단위	>> 1+2j ans = 1.0000 + 2.0000i	가급적 1i 또는 1j라고 쓸 것
conj	켤레복소수	>> conj(1+2j) ans = 1.0000 - 2.0000i	
real	복소수의 실수부	>> real(1+2j) ans = 1	
imag	복소수의 허수부	>> imag(1+2j) ans = 2	
angle	복소수의 위상각	>> angle(1+1j) ans = 0.7854	radian [-pi, pi] 범위

함수명	동작	예시	참고
pi	원주율	>> cos(pi) ans = -1	
inf	무한대	>> a = 1/0 a = Inf	isinf
nan	not a number	>> a = 0/0 a = NaN	isnan
ans	변수를 지정하지 않은 연산의 결과를 임시저장	>> 1+2+3 ans = 6	언제든 덮어쓸 수 있으므로 사용 금지



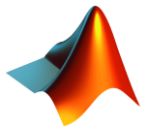
자주 쓰는 내장함수 – 정수를 다루는 함수들

factor(x)	Finds the prime factors of **x**.	**factor(12)** **ans =** **2 2 3**
gcd(x,y)	Finds the greatest common denominator of **x** and **y**.	**gcd(10,15)** **ans =** **5**
lcm(x,y)	Finds the least common multiple of **x** and **y**.	**lcm(2,5)** **ans =** **10** **lcm(2,10)** **ans =** **10**
rats(x)	Represents **x** as a fraction.	**rats(1.5)** **ans =** **3/2**
factorial(x)	Finds the value of **x** factorial (**x**!). A factorial is the product of all the integers less than **x**. For example, $6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 720$	**factorial(6)** **ans =** **720**
nchoosek(n,k)	Finds the number of possible combinations of *k* items from a group of *n* items. For example, use this function to determine the number of possible subgroups of 3 chosen from a group of 10.	**nchoosek(10,3)** **ans =** **120**
primes(x)	Finds all the prime numbers less than **x**.	**primes(10)** **ans =** **2 3 5 7**
isprime(x)	Checks to see if **x** is a prime number. If it is, the function returns 1; if not, it returns 0.	**isprime(7)** **ans =** **1** **isprime(10)** **ans =** **0**
prod(x)	모든 원소의 곱	

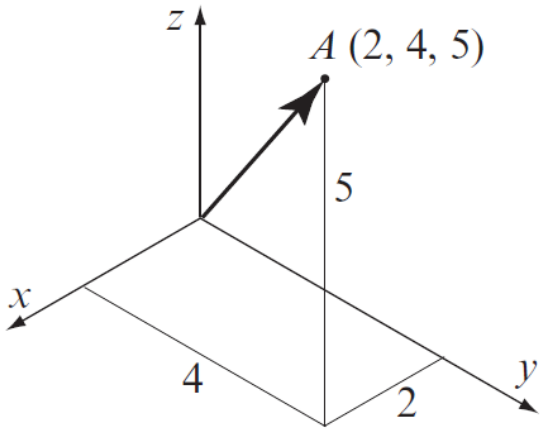
※ 변수명에 내장함수를 쓰지 말 것

```
>> sin = 100;
>> sin(pi)
배열 인덱스는 양의 정수이거나 논리값이어야 합니다.
```

행 렬



배열이란? 왜 필요할까?



- 3차원 공간 상의 한 점의 좌표 / 두 벡터의 합
- 도시 A의 년도별 인구

Year	1984	1986	1988	1990	1992	1994	1996
Population (millions)	127	130	136	145	158	178	211

- 책장의 각 칸에 들어있는 책 권수

- 이미지 = 2차원 배열

```
>> img = imread('cameraman.tif');
>> imshow(img)
```

img

256x256 uint8

	1	2	3	4
1	156	159	158	155
2	160	154	157	158
3	156	159	158	155
4	160	154	157	158
5	156	153	155	159
6	155	155	155	157
7	156	153	157	156

행렬이란?

※ 연산 방법에 주의할 것

- 1차 연립 방정식을 좀 더 간단하게 써보자.

$$\begin{cases} x + 2y - 4z = 5 \\ 2x + y - 6z = 8 \\ 4x - y - 12z = 13 \end{cases}$$

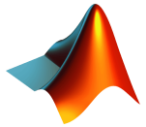
$$\begin{bmatrix} 1 & 2 & -4 \\ 2 & 1 & -6 \\ 4 & -1 & -12 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 13 \end{bmatrix}$$

3×3 행렬 | 3×1 행렬 (벡터) | 3×1 행렬 (벡터)

$$\begin{aligned} 2x + y + z &= 1 \\ 5x - y + 7z &= 0 \end{aligned}$$

$$\begin{bmatrix} 2 & 1 & 1 \\ 5 & -1 & 7 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

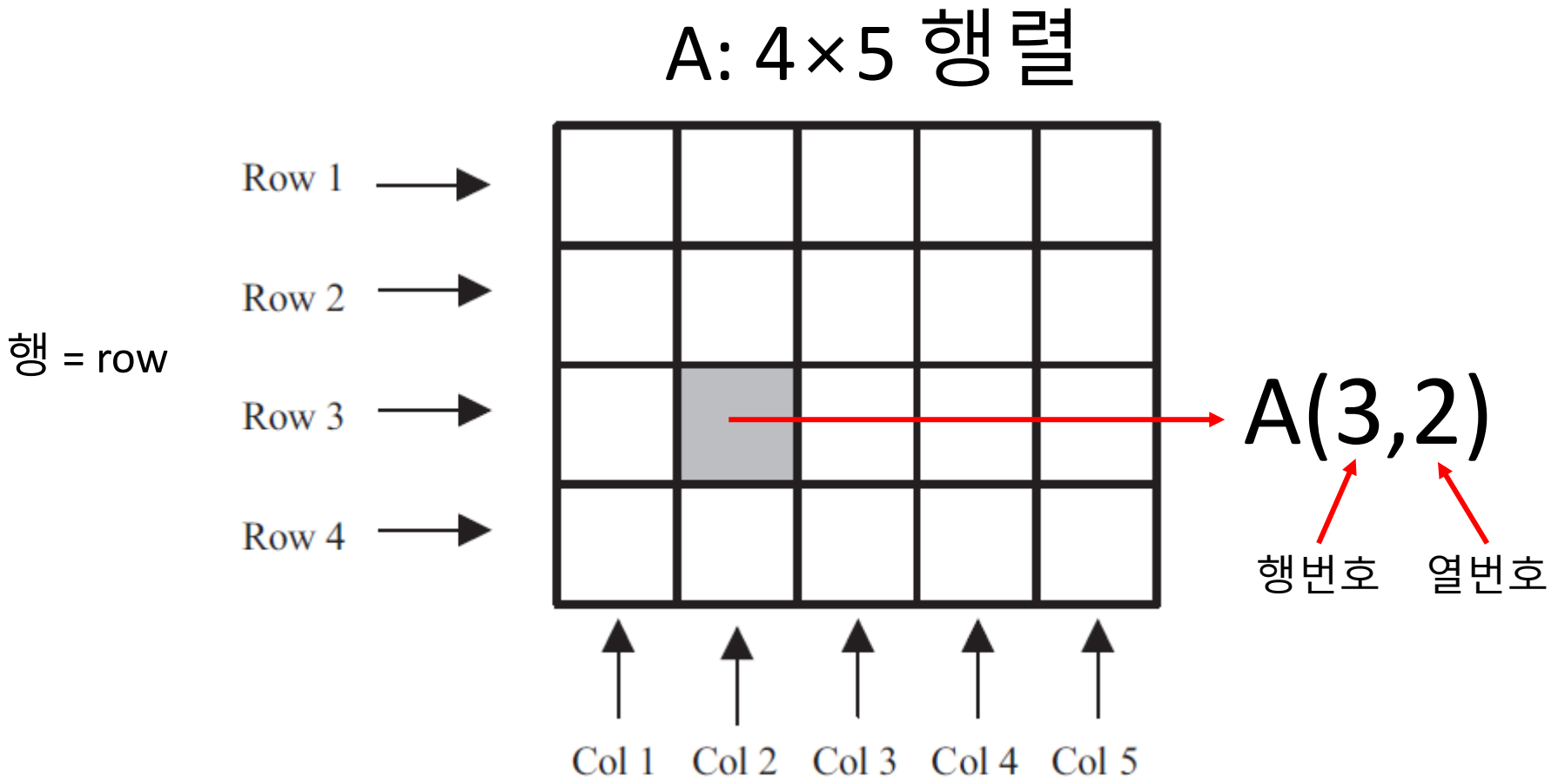
2×3 행렬 | 3×1 행렬 (벡터) | 2×1 행렬 (벡터)



행렬이란?

엑셀?

- 1개 이상의 수, 식, 기호 등을 사각형의 배열로 나열한 것



행렬: 수학 vs 매트랩

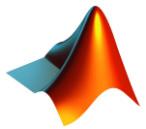
- 매트랩에서는 모든 것이 행렬이다.
  - 순수 스칼라는 없다.
  - 모든 자료형에 size와 length를 쓸 수 있다.

tip1. 매트랩에서…
- array와 matrix는 같은 말
- scalar는 vector의 부분집합
- vector는 matrix의 부분집합
- scalar ⊂ vector ⊂ matrix

tip2. M×M: square matrix

tip3. 3차원? 4차원? N차원?

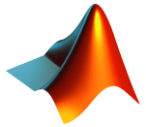
값	용어	수학에서	매트랩에서
100	스칼라	값 1개	1×1 행렬
$[3 \quad 9 \quad 17]$, $\begin{bmatrix} 5 \\ 8 \\ 13 \end{bmatrix}$	벡터	1×N 벡터 (행벡터) N×1 벡터 (열벡터)	1×N 행렬 (행벡터) N×1 행렬 (열벡터)
$\begin{bmatrix} 1 & 2 & -4 \\ 2 & 1 & -6 \\ 4 & -1 & -12 \end{bmatrix}$	행렬	M×N 행렬	M×N 행렬



행렬을 만드는 방법 1 – 모든 원소를 직접 입력

```
a = [1,2,3,4];

b = [5 6 7 8];

c = [1,2,3; 4,5,6; 7,8,9];

d = [1 10 100
     2 20 200
     3 30 300];

e = [];
```

- 변수명 = [값1, 값2, 값3 ; 값4, 값5, 값6];
  - 값의 구분은 빈칸, 컴마 모두 가능 (가독성?)
  - 빈 행렬: [ ]
- 새로운 행 시작: 엔터 또는 세미콜론
  - 주의: 행이 너무 길어질 땐? → …를 이용
    - 이때 새로운 행이 시작되는 것이 아님
- 명령창에서 입력하다가 실수했을 때?
  - ctrl+c
  - 대부분의 “취소”에 해당하는 단축키



행렬을 만드는 방법 2 – 콜론 연산자

a1 = 1:10; ➔ a1 = [1 2 3 4 5 6 7 8 9 10]

a2 = 1:2:9; ➔ a2 = [1 3 5 7 9]

b1 = 10:-1:5; ➔ b1 = [10 9 8 7 6 5]

b2 = 10:5; ➔ b2 = [ ]

c1 = 5:5; ➔ c1 = 5

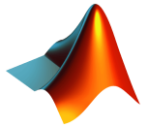
c2 = 5:10:5; ➔ c2 = 5

d = 1:10:100; ➔ d = [1 11 21 … 91]

- 변수명 = 시작값:간격:끝값
  - 간격은 생략하면 1
  - 음수 간격도 가능
  - 시작값>끝값이고 간격>0이면 → 빈 행렬
  - 시작값<끝값이고 간격<0이면 → 빈 행렬
  - 끝값에 도달할 수 없으면 → 끝값 전 마지막 값까지

year = 1984:2:1996

➔ year = [1984 1986 1988 1990 1992 1994 1996]



행렬을 만드는 방법 3 – linspace

a1 = linspace(1,10,10); ➔ a1 = [1 2 3 4 5 6 7 8 9 10]

a2 = linspace(1,9,5); ➔ a2 = [1 3 5 7 9]

a3 = linspace(1,100); ➔ a3 = [1 2 3 … 99 100]

b1 = linspace(9,1,5); ➔ b1 = [9 7 5 3 1]

b2 = linspace(100,10,1); ➔ b2 = 10

b3 = linspace(100,10,2); ➔ b3 = [100 10]

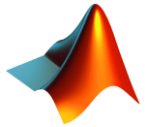
c1 = linspace(10,10); ➔ c1 = [10 10 10 … 10]

x = linspace(0, pi, 1000);

➔ x = [0 0.0031 0.0063 … 3.1416];

- linspace(시작값,끝값,개수)
  - 간격은 알아서 계산 (=(끝값-시작값)/(개수-1))
  - 참고: 콜론 연산자는 개수를 알아서 계산
  - 개수를 생략하면 100개
  - 끝점은 항상 포함
- 콜론 연산자 vs linspace
  - 간격을 알 때 → 콜론 연산자
    - 1920년에서 5년 간격으로 2020년까지
  - 개수를 알 때 → linspace
    - 0~pi를 1000등분

※ 열벡터를 만드는 방법
1) 세미콜론
2) 행벡터 + transpose(')



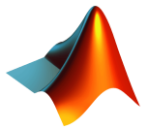
행렬을 만드는 방법 4 – 이미 있는 행렬을 합침

```
>> a1 = [1,2,3];
>> a2 = [4,5,6];
>> ar = [a1 a2]
ar =
     1     2     3     4     5     6
>> ac = [a1; a2]
ac =
     1     2     3
     4     5     6
>> at = [a1' a2']
at =
     1     4
     2     5
     3     6
```

```
>> a3 = [1:3, 4:7]
a3 =
     1     2     3     4     5     6     7
```

```
>> a4 = [1:5; 6:10]
a4 =
     1     2     3     4     5
     6     7     8     9    10
```

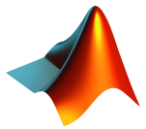
```
>> al = [linspace(10,20,6); linspace(20,30,6)]
al =
    10    12    14    16    18    20
    20    22    24    26    28    30
```



행렬을 만드는 방법 4 – 이미 있는 행렬을 합침

```
>> a = [1 2; 3 4];
>> b = [a a ; a a]
b =
     1     2     1     2
     3     4     3     4
     1     2     1     2
     3     4     3     4

>> a = [1 2; 3 4];
>> b = [5 6]';
>> c = [7 8];
>> d = 9;
>> e = [a b ;c d]
e =
     1     2     5
     3     4     6
     7     8     9
```

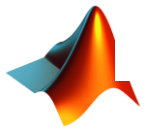
```
>> a = [1 2 ; 3 4];
>> b = [5 6];
>> a = [a;b]
a =
     1     2
     3     4
     5     6
>> a = [a (1:3)']
a =
     1     2     1
     3     4     2
     5     6     3

>> a = [1 2 3; 4 5 6];
>> b = [4 5 6]';
>> c = [a b]
다음 사용 중 오류가 발생함: horzcat
결합하려는 배열의 차원이 일치하지 않습니다.
```



행렬을 만드는 방법 5 – 특별한 함수들

```
>> ones(3,4)
ans =
     1     1     1     1
     1     1     1     1
     1     1     1     1
>> ones(5)
ans =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

```
>> zeros(2,4)
ans =
     0     0     0     0
     0     0     0     0
>> zeros(3)
ans =
     0     0     0
     0     0     0
     0     0     0
```



```
>> a = 1:5;
>> b = repmat(a,3,1)
b =
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
```

```
>> a = 1:30;
>> b = reshape(a,5,6)
b =
     1     6    11    16    21    26
     2     7    12    17    22    27
     3     8    13    18    23    28
     4     9    14    19    24    29
     5    10    15    20    25    30
```

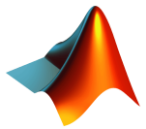
행렬의 연산 – 더하기, 빼기, *스칼라, /스칼라

1) 행렬 + 행렬
   - 크기가 같은 행렬만 가능(이지만 꼭 그런건 아님)
   - 각 위치 원소별로 더하기 or 빼기

2) 행렬 + 스칼라 (1×1 행렬)
   - 행렬의 모든 원소에 일괄적으로 스칼라를 더함

3) *스칼라, /스칼라
   - 모든 원소를 동일하게 곱하거나 나눔

```
>> a = (1:5);
>> b = (11:15);
>> c = a+b

c =

    12    14    16    18    20

>> c = c+10

c =

    22    24    26    28    30

>> c = c*2

c =

    44    48    52    56    60

>> c = c/4

c =

    11    12    13    14    15
```



행렬의 연산 – 곱하기 (행렬곱)

- 행렬곱이 정의되려면
  → (앞 행렬의 열 개수) = (뒤 행렬의 행 개수)
- (M×N 행렬) × (P×Q 행렬)
  - N=P이어야 행렬곱이 정의됨
  - 곱의 결과 = M×Q 행렬

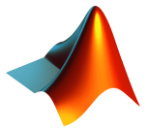
$$\begin{cases} x + 2y - 4z = 5 \\ 2x + y - 6z = 8 \\ 4x - y - 12z = 13 \end{cases}$$

$$\begin{bmatrix} 1 & 2 & -4 \\ 2 & 1 & -6 \\ 4 & -1 & -12 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 13 \end{bmatrix}$$

$$\begin{aligned} 2x + y + z &= 1 \\ 5x - y + 7z &= 0 \end{aligned}$$

$$\begin{bmatrix} 2 & 1 & 1 \\ 5 & -1 & 7 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 & 3 \\ 2 & 6 & 1 \\ 5 & 2 & 8 \end{bmatrix} \begin{bmatrix} 5 & 4 \\ 1 & 3 \\ 2 & 6 \end{bmatrix} = \begin{bmatrix} (1\cdot5+4\cdot1+3\cdot2) & (1\cdot4+4\cdot3+3\cdot6) \\ (2\cdot5+6\cdot1+1\cdot2) & (2\cdot4+6\cdot3+1\cdot6) \\ (5\cdot5+2\cdot1+8\cdot2) & (5\cdot4+2\cdot3+8\cdot6) \end{bmatrix} = \begin{bmatrix} 15 & 34 \\ 18 & 32 \\ 43 & 74 \end{bmatrix}$$



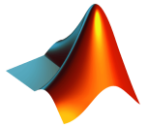
행렬의 연산 – 곱하기 (행렬곱)

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

- 행렬곱이 정의되려면
  → (앞 행렬의 열 개수) = (뒤 행렬의 행 개수)
- (M×N 행렬) × (P×Q 행렬)
  - N=P이어야 행렬곱이 정의됨
  - 곱의 결과 = M×Q 행렬

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \end{bmatrix}$$

$$A \times B = \begin{bmatrix} (A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31}) & (A_{11}B_{12} + A_{12}B_{22} + A_{13}B_{32}) \\ (A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31}) & (A_{21}B_{12} + A_{22}B_{22} + A_{23}B_{32}) \\ (A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31}) & (A_{31}B_{12} + A_{32}B_{22} + A_{33}B_{32}) \\ (A_{41}B_{11} + A_{42}B_{21} + A_{43}B_{31}) & (A_{41}B_{12} + A_{42}B_{22} + A_{43}B_{32}) \end{bmatrix}$$



https://medium.com/ai%C2%B3-theory-practice-business/fastai-partii-lesson08-notes-fddcdb6526bb

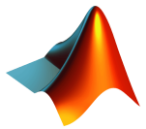
행렬의 연산 – 곱하기 (행렬곱)

```
>> a
a =
     1     4     7    10
     2     5     8    11
     3     6     9    12
>> b
b =
    16     2     3
     5    11    10
     9     7     6
     4    14    15
>> c = a*b
c =
   139   235   235
   173   269   269
   207   303   303
```

```
>> a = [1 2 3 4 5];
>> b = [1 2 3 4 5]';
>> c = a*b
c =
    55
```

```
>> s
s =
     8     1     6
     3     5     7
     4     9     2
>> s^2
ans =
    91    67    67
    67    91    67
    67    67    91
```

- 행렬곱이 정의되려면
  → (앞 행렬의 열 개수) = (뒤 행렬의 행 개수)
- (M×N 행렬) × (P×Q 행렬)
  - N=P이어야 행렬곱이 정의됨
  - 곱의 결과 = M×Q 행렬
- 행렬의 거듭제곱
  - square matrix에 대해서만 정의됨
- 행렬곱의 특징
  - 결합법칙 O A(BC) = (AB)C
  - 분배법칙 O A(B+C) = AB + AC
  - 교환법칙 X AB ≠ BA



행렬의 연산 – 곱하기 (원소별 곱셈)

$$a = [1 \quad 2 \quad 3]$$
$$\times \quad \times \quad \times$$
$$b = [4 \quad 3 \quad 1]$$
$$\| \quad \| \quad \|$$
$$c = [4 \quad 6 \quad 3]$$

```
>> a = [1 2 3];
>> b = [4 3 1];
>> a*b
다음 사용 중 오류가 발생함: *
행렬 곱셈의 차원이 잘못되었습니다. 첫 번째 행렬의 열 개수가 두 번째 행렬의 행 개수와 일치하는지 확인하십시오. 요소별 곱셈 연산을 수행하려면 '.*'를 사용하십시오.

>> a.*b

ans =

     4     6     3
```

```
>> s = magic(3)
s =
     8     1     6
     3     5     7
     4     9     2
>> s^2
ans =
    91    67    67
    67    91    67
    67    67    91
>> s.^2
ans =
    64     1    36
     9    25    49
    16    81     4
```

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \text{ and } B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

- *는 행렬곱
- .*는 원소별 곱셈
- 두 행렬의 크기는 같아야 함

$$A .* B = \begin{bmatrix} A_{11}B_{11} & A_{12}B_{12} & A_{13}B_{13} \\ A_{21}B_{21} & A_{22}B_{22} & A_{23}B_{23} \\ A_{31}B_{31} & A_{32}B_{32} & A_{33}B_{33} \end{bmatrix} \quad \text{A .^ } n = \begin{bmatrix} (A_{11})^n & (A_{12})^n & (A_{13})^n \\ (A_{21})^n & (A_{22})^n & (A_{23})^n \\ (A_{31})^n & (A_{32})^n & (A_{33})^n \end{bmatrix}$$

https://www.youtube.com/watch?v=mrfvYq0xMmw

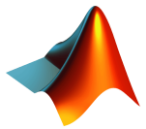
행렬의 연산 – 곱하기 (원소별 곱셈)

- 행렬의 원소별 곱셈 → 연산자 앞에 점(.)을 찍음
  - 두 행렬을 원소별 곱셈 → A.*B
  - 한 행렬을 원소별 거듭제곱 → A.^n
  - 스칼라.^행렬 → 스칼라를 행렬 각 원소별로 거듭제곱

```
>> s1
s1 =
     1     1     1     1
     2     2     2     2
     3     3     3     3
>> s2
s2 =
     1     2     3     4
     1     2     3     4
     1     2     3     4
```

```
>> s1*s2
다음 사용 중 오류가 발생함: *
행렬 곱셈의 차원이 잘못되었습니다. 첫 번째 행렬의 열 개수가 두 번째 행렬의 행 개수
와 일치하는지 확인하십시오. 요소별 곱셈 연산을 수행하려면 '.*'를 사용하십시오.
>> s1.*s2
ans =
     1     2     3     4
     2     4     6     8
     3     6     9    12
>>
>> s1^2
다음 사용 중 오류가 발생함: ^ (line 51)
차원이 정확하지 않아 행렬을 거듭제곱할 수 없습니다. 행렬이 정사각 행렬이고 지수 값이 스칼라인지
확인하십시오. 요소별 행렬 거듭제곱 연산을 수행하려면 '.^'을 사용하십시오.
>> s1.^2
ans =
     1     1     1     1
     4     4     4     4
     9     9     9     9
```

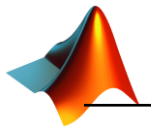
```
>> 2.^[1 2 ; 3 4]
ans =
     2     4
     8    16
```



https://www.youtube.com/watch?v=mrfvYq0xMmw

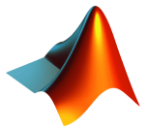
행렬의 연산 – 나누기 (역행렬)

	덧셈 항등원	곱셈 항등원	곱셈 역원
정의	더해도 값이 그대로인 수 $a + e = e + a = a$	곱해도 값이 그대로인 수 $a \times e = e \times a = a$	곱해서 곱셈 항등원이 나오는 수 $a \times x = x \times a = e$
실수	$e = 0$	$e = 1$	$x = \frac{1}{a} \ (a \neq 0)$
행렬	O = 영행렬 $O = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ O = zeros(N);	E = 단위행렬 $E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ E = eye(N); ※ E는 square matrix	x = 역행렬 $x = a^{-1}$ x = inv(a) ※ square matrix에 대해서만 존재 ※ a가 invertible할 때만 존재



행렬의 연산 – 나누기 (역행렬)

```
>> a = magic(3);
>> b = inv(a)

b =

    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028

>> a*b

ans =

    1.0000         0   -0.0000
   -0.0000    1.0000         0
    0.0000         0    1.0000

>> b*a

ans =

    1.0000         0   -0.0000
         0    1.0000         0
         0    0.0000    1.0000
```



```
a =

     1     4     7    10
     2     5     8    11
     3     6     9    12

>> inv(a)
다음 사용 중 오류가 발생함: inv
행렬은 정사각 행렬이어야 합니다.
```

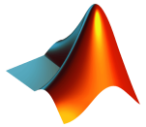
```
a =

     1     2     3
     2     4     6
     7     8     9

>> inv(a)
경고: 행렬이 설정된 작업 정밀도에서 특이 행렬입니다.

ans =

   Inf   Inf   Inf
   Inf   Inf   Inf
   Inf   Inf   Inf
```

행렬의 연산 – 나누기 (역행렬)

$$\begin{cases} x + 2y - 4z = 5 \\ 2x + y - 6z = 8 \\ 4x - y - 12z = 13 \end{cases} \qquad \begin{bmatrix} 1 & 2 & -4 \\ 2 & 1 & -6 \\ 4 & -1 & -12 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 13 \end{bmatrix}$$

$$A \cdot x = b$$
$$A^{-1} \cdot A \cdot x = A^{-1} \cdot b$$
$$E \cdot x = A^{-1} \cdot b$$
$$x = A^{-1} \cdot b$$

```
>> a
a =
     1     2    -4
     2     1    -6
     4    -1   -12
>> b
b =
     5
     8
    13
>> x = inv(a)*b
x =
    5.0000
    1.0000
    0.5000
>> a*x
ans =
     5
     8
    13
```



행렬의 연산 – 나누기 (원소별 나눗셈)

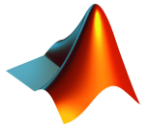
$$a = [1 \quad 2 \quad 3]$$
$$\quad\quad / \quad / \quad /$$
$$b = [4 \quad 3 \quad 1]$$
$$\quad\quad \| \quad \| \quad \|$$
$$c = \left[\frac{1}{4} \quad \frac{2}{3} \quad 3\right]$$

```
>> a = [1 2 3];
>> b = [4 3 1];
>> a/b
ans =
    0.5000  ??
>> a./b
ans =
    0.2500    0.6667    3.0000
```

OK

- 원소별 곱셈과 동일
- / 대신 ./ 사용
- 두 행렬의 크기는 같아야 함

```
>> s1
s1 =
     1     1     1     1
     2     2     2     2
     3     3     3     3
>> s2
s2 =
     1     2     3     4
     1     2     3     4
     1     2     3     4
>> s1/s2
경고: 랭크 부족, rank = 1, tol =  4.864754e-15.

ans =
    0.3333         0         0
    0.6667         0         0
    1.0000         0         0
>> s1./s2
ans =
    1.0000    0.5000    0.3333    0.2500
    2.0000    1.0000    0.6667    0.5000
    3.0000    1.5000    1.0000    0.7500
```



행렬의 연산 – 나누기 (원소별 나눗셈)

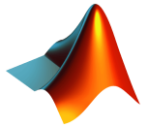
- 행렬의 원소별 나눗셈 → 연산자 앞 점(.)을 찍음
  - 두 행렬을 원소별 나눗셈 → A./B
  - 행렬의 각 원소의 역수 → 1./A

※ 실습: “복잡한 수식”을 원소별 연산으로

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \text{ and } B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

$$A ./ B = \begin{bmatrix} A_{11}/B_{11} & A_{12}/B_{12} & A_{13}/B_{13} \\ A_{21}/B_{21} & A_{22}/B_{22} & A_{23}/B_{23} \\ A_{31}/B_{31} & A_{32}/B_{32} & A_{33}/B_{33} \end{bmatrix}$$

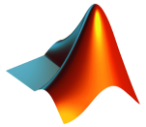
```
>> s = magic(3)
s =
     8     1     6
     3     5     7
     4     9     2

>> 1./s
ans =
    0.1250    1.0000    0.1667
    0.3333    0.2000    0.1429
    0.2500    0.1111    0.5000
>> 10./s
ans =
    1.2500   10.0000    1.6667
    3.3333    2.0000    1.4286
    2.5000    1.1111    5.0000
```



https://www.youtube.com/watch?v=mrfvYq0xMmw

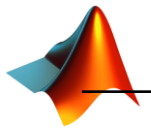
행렬을 다루는 내장함수들

함수명	동작	예시	참고
min [i, v] = min(A)	행렬의 최소값 [i, v] → 최소값의 위치도 반환	>> min([5 3 1 2 4]) ans = 1	2차원일 경우 각 열의 최소값을 1행으로 반환
max [i, v] = max(A)	행렬의 최대값 [i, v] → 최대값의 위치도 반환	>> max([5 3 1 2 4]) ans = 5	2차원일 경우 각 열의 최대값을 1행으로 반환
size	행렬의 크기	>> size(magic(5)) ans = 5 5	N차원 행렬일 경우 길이가 N인 벡터 반환
length	행렬의 길이	>> length(magic(5)) ans = 5	size의 결과 중 최대값과 같음
numel	모든 원소의 개수	>> numel(magic(5)) ans = 25	size의 결과 원소를 모두 곱한 것과 같음



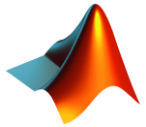
행렬을 다루는 내장함수들

함수명	동작	예시	참고
' transpose	전치행렬	>> (1:2)' ans = 1 2	정확히는 conjugate transpose (ctranspose) non-conjugate는 .'
flipud	행렬을 위아래로 뒤집음	>> flipud([1 2; 3 4]) ans = 3 4 1 2	
fliplr	행렬을 좌우로 뒤집음	>> fliplr([1 2; 3 4]) ans = 2 1 4 3	
reshape	행렬의 원소를 재배열	>> reshape(1:6,2,3) ans = 1 3 5 2 4 6	reshape(a,m,n) a의 원소 개수가 m×n과 같아야 함
repmat	행렬을 복제하여 이어붙임	>> repmat(1:3,2,2) ans = 1 2 3 1 2 3 1 2 3 1 2 3	



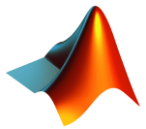
행렬을 다루는 내장함수들

함수명	동작	예시	참고
ones	모든 원소가 1인 행렬 ones(m, n) → m×n 크기의 행렬 ones(n) → n×n 크기의 행렬	>> ones(1,5) ans = 1 1 1 1 1	모든 원소가 100인 행렬 100*ones(m,n)
zeros	모든 원소가 0인 행렬 zeros(m, n) → m×n 크기의 행렬 zeros(n) → n×n 크기의 행렬	>> zeros(1,4) ans = 0 0 0 0	
norm	벡터 노름(norm) $\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{N} \|x_i\|^2\right)^{1/2} = \sqrt{x_1^2 + x_2^2 + \ldots + x_N^2}$	>> norm([3,4]) ans = 5	
find [i, j] = find(a)	행렬의 0이 아닌 원소의 위치 반환 [i, j] = find(a) → 0이 아닌 원소의 위치를 2차원 인덱스로 반환	>> find([1 2 0 0 2]) ans = 1 2 5	
sort [i, v] = sort(a)	행렬을 오름차순으로 정렬 2차원 행렬 → 각 열을 정렬 [i, v] = sort(A) → 정렬 후 인덱스 변화도 반환	>> sort([1 3 5 4 2]) ans = 1 2 3 4 5	'descend' 옵션 사용 시 내림차순 정렬



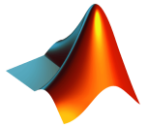
행렬을 다루는 내장함수들

함수명	동작	예시	참고
mean mean2	mean(A) 또는 mean(A,1) → 행방향 평균 mean(A,2) → 열방향 평균 mean2(A) → 2차원 평균	>> mean([1 3 5 ; 3 5 7]) ans = 2 4 6	
std std2	std(A) 또는 std(A,1) → 행방향 표준편차 std(A,2) → 열방향 표준편차 std2(A) → 2차원 표준편차	>> std(magic(4)) ans = 5.4467 5.1962 5.1962 5.4467	
sum	sum(A) 또는 sum(A,1) → 행방향 합 sum(A,2) → 열방향 합	>> sum([1 3 5 ; 3 5 7]) ans = 4 8 12	
cumsum	벡터 → 누적합 행렬 → 각 열의 누적합	>> cumsum([1 2 3 4 5]) ans = 1 3 6 10 15	
median	중간값 짝수개일 경우 두 중간값의 평균	>> median([1 3 5 7]) ans = 4	



행렬을 다루는 내장함수들

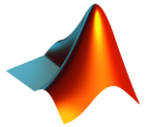
함수명	동작	예시	참고
rand	0~1 범위의 균등분포 난수 rand(m, n) → m×n 크기의 행렬 rand(n) → n×n 크기의 행렬	>> rand(2,4) ans = 0.8147 0.1270 0.6324 0.2785 0.9058 0.9134 0.0975 0.5469	hist로 분포 확인 가능
randn	평균 0, 표준편차 1인 정규분포 난수 randn(m, n) → m×n 크기의 행렬 randn(n) → n×n 크기의 행렬	>> randn(2,3) ans = 0.3480 -0.6357 -0.7615 -0.4551 -0.9799 -1.2835	hist로 분포 확인 가능
randi	균등 분포의 정수 난수 값 범위, 크기 설정 가능	>> randi([1,50],[2,3]) ans = 21 42 5 8 37 42	도움말 확인



행렬을 입력으로 받는 내장함수

- 여러 내장함수들 (sqrt, exp, log, sin, round, …)
  - M×N 행렬을 넣으면
    - 알아서 각 원소를 연산하여 M×N 행렬을 반환

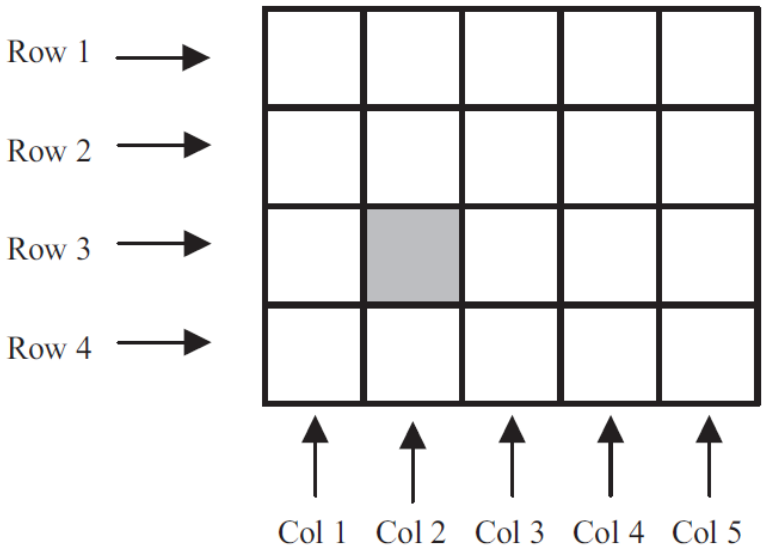
```
>> sqrt([1 4 9 16])
ans =
     1     2     3     4
>> sind(90*(0:4))
ans =
     0     1     0    -1     0
>> round(pi*(1:5))
ans =
     3     6     9    13    16
```

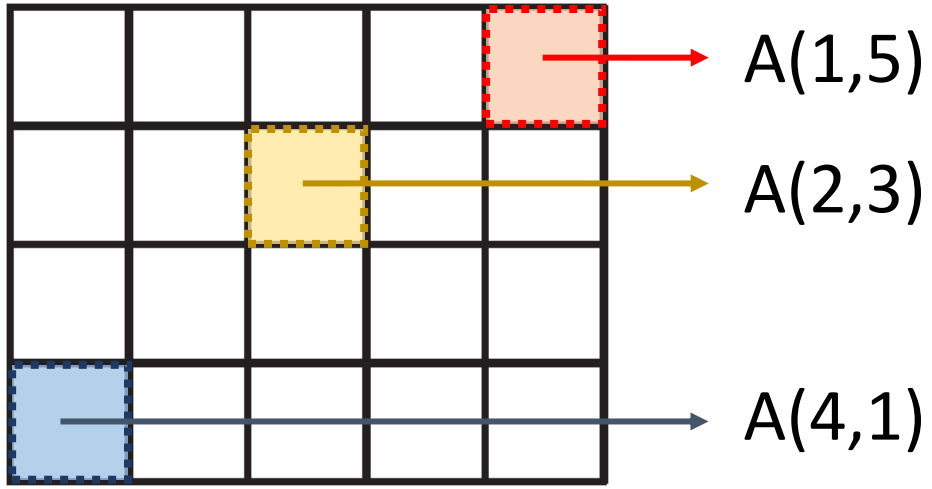


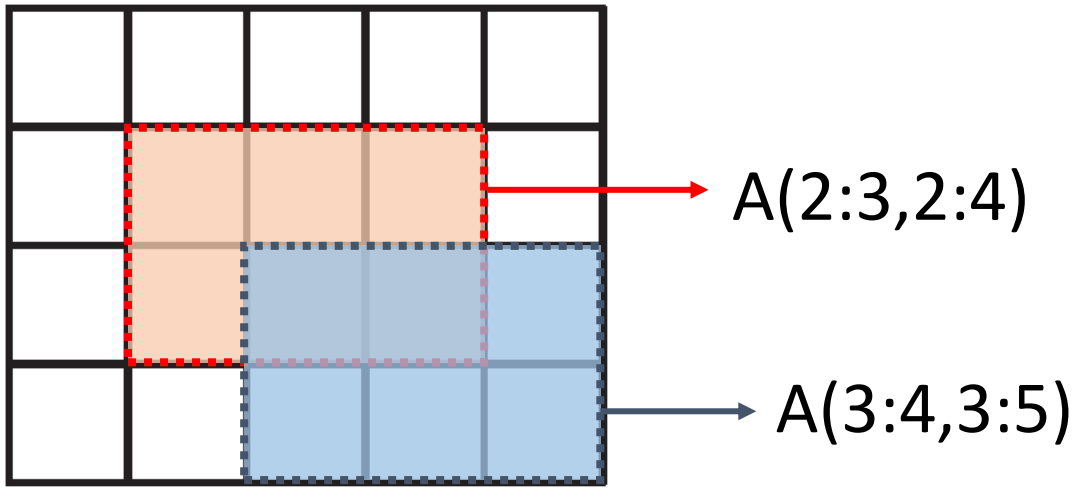
행렬의 인덱싱 (indexing)

변수명(행범위, 열범위)

A: 4×5 행렬



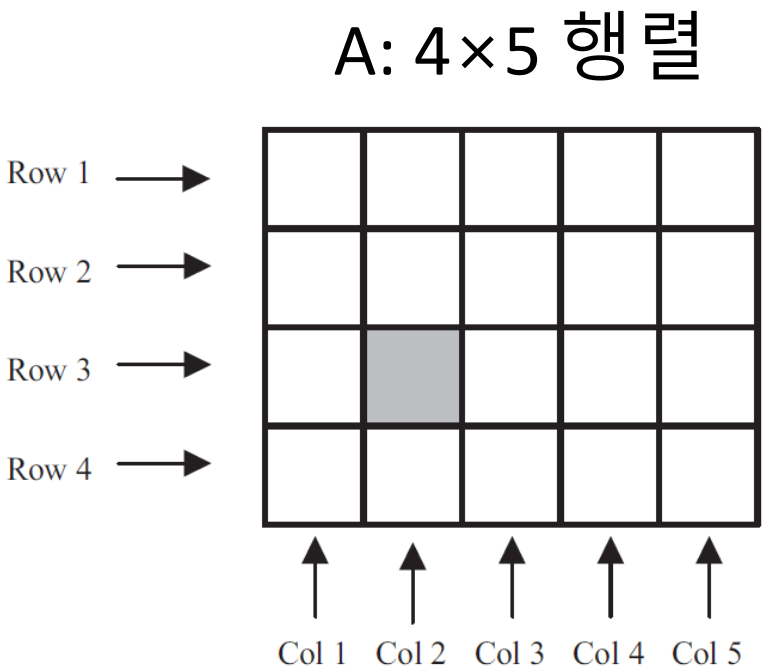


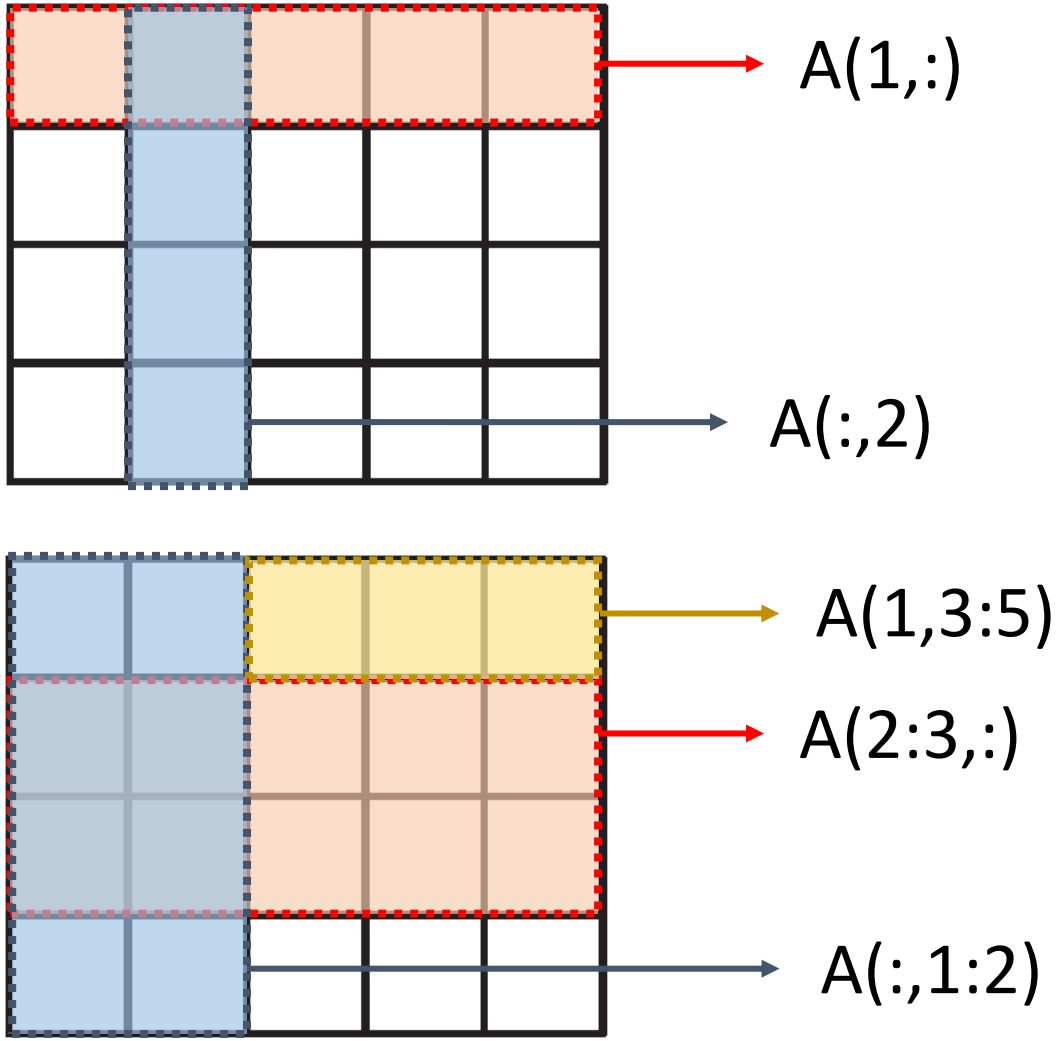


https://lazymatlab.tistory.com/28

행렬의 인덱싱 (indexing)

변수명(행범위, 열범위)



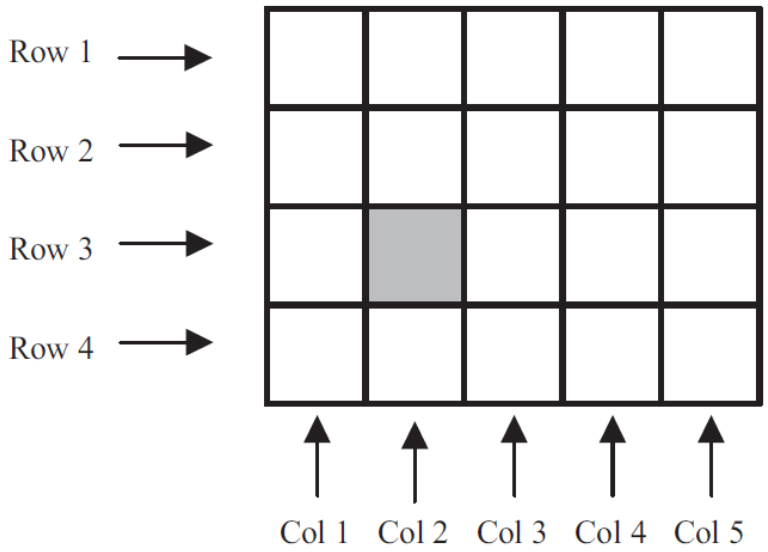


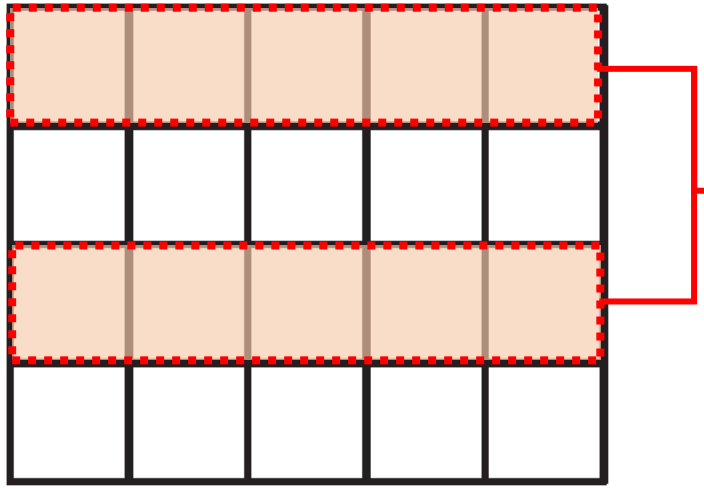
https://lazymatlab.tistory.com/28

행렬의 인덱싱 (indexing)

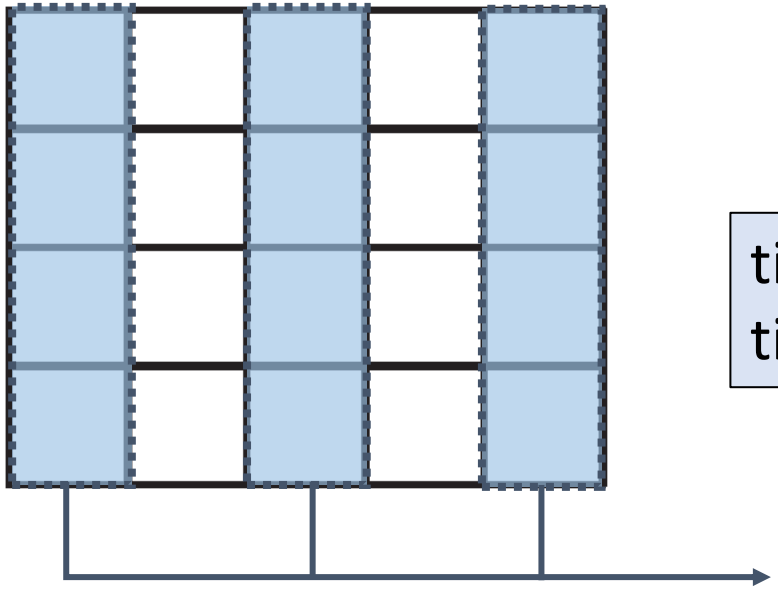
변수명(행범위, 열범위)

A: 4×5 행렬





A([1,3],:)



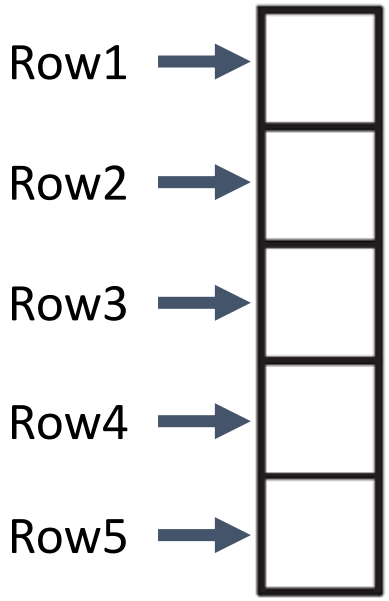
A(:,[1,3,5])

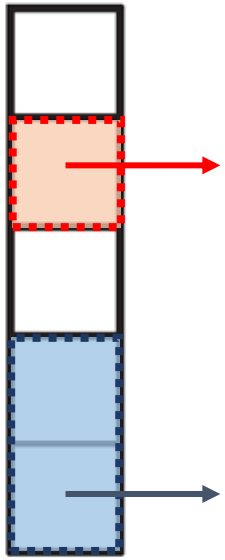
tip1. A(:,1:2:5)도 가능
tip2. A(:,[5 3 1])=?

https://lazymatlab.tistory.com/28

행렬의 인덱싱 (indexing) – 벡터의 경우

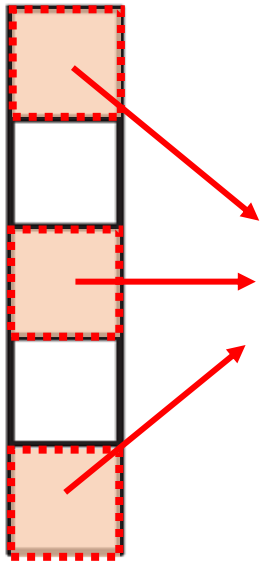
B: 5×1 행렬





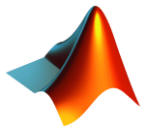
B(2,1) 또는 B(2,:) 또는 B(2)

B(4:5,1) 또는 B(4:5,:) 또는 B(4:5)
또는 B([4,5])



B([1,3,5],1) 또는 B(1:2:5,:)
또는 B([1,3,5])

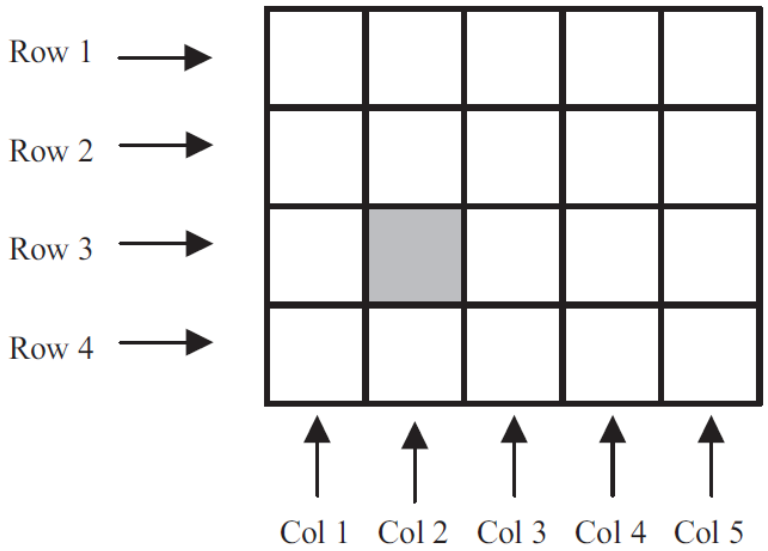
tip. B([5 3 1])=? (행렬의 reorder)

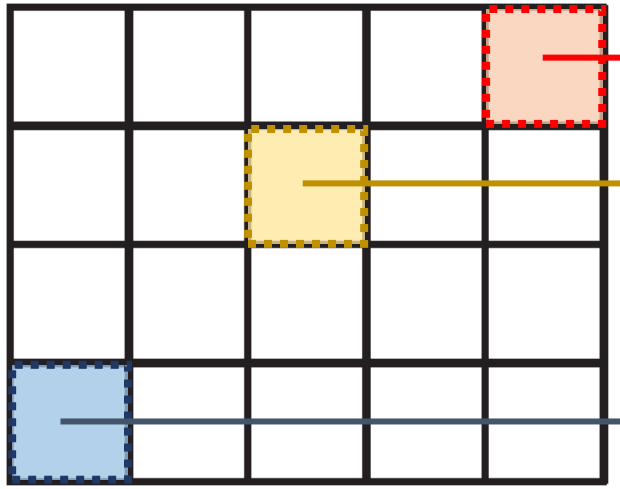


https://lazymatlab.tistory.com/28

행렬의 인덱싱 (indexing) – end의 활용

A: 4×5 행렬

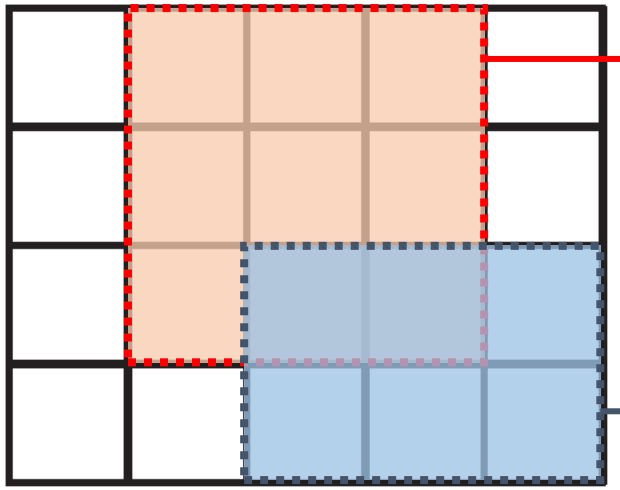




A(1,5) 또는 A(1,end)

A(2,3) 또는 A(2, end-2)

A(4,1) 또는 A(end,1)



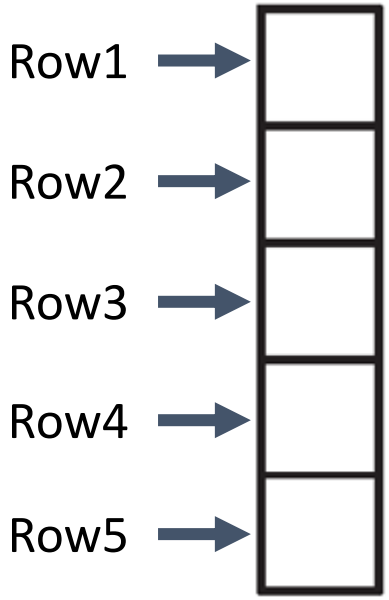
A(1:3,2:4)
또는 A(1:3, 2:end-1)

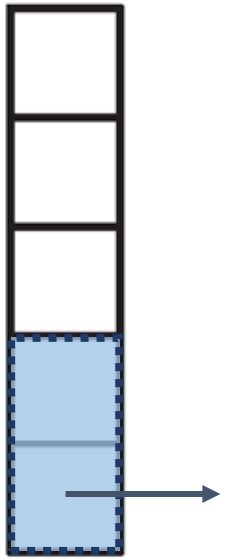
A(3:4,3:5)
또는 A(3:end, 3:end)

https://lazymatlab.tistory.com/28

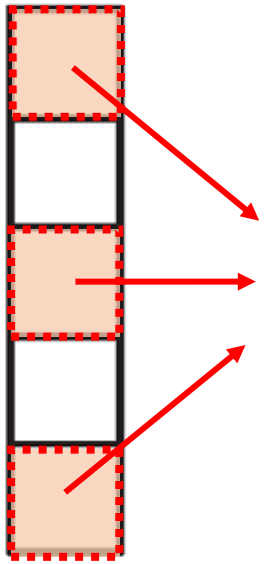
행렬의 인덱싱 (indexing) – end의 활용

B: 5×1 행렬





B(4:5) 또는 B(4:end)



B(1:2:5) 또는 B([1,3,end])

https://lazymatlab.tistory.com/28

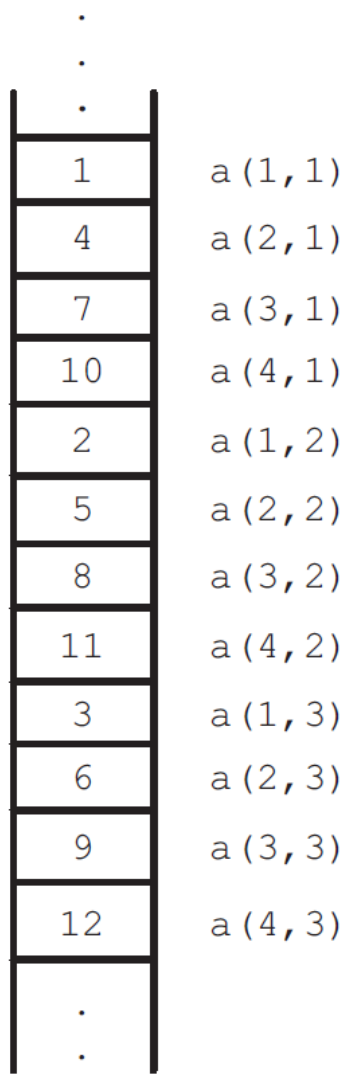
행렬의 인덱싱 (indexing) – (:)의 활용

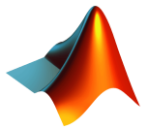
```
>> a
a =
     1     5     9
     2     6    10
     3     7    11
     4     8    12
>> a(:)
ans =
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
    11
    12
```

1	2	3
4	5	6
7	8	9
10	11	12

a

Arrangement in Computer Memory





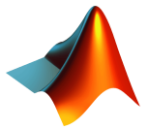
https://lazymatlab.tistory.com/28

행렬을 수정하는 방법

```
>> a = reshape(1:20,4,5)
a =
     1     5     9    13    17
     2     6    10    14    18
     3     7    11    15    19
     4     8    12    16    20
```

```
>> a(2,3) = 100
a =
     1     5     9    13    17
     2     6   100    14    18
     3     7    11    15    19
     4     8    12    16    20
```

```
>> a(:,3) = [100 101 102 103]'
a =
     1     5   100    13    17
     2     6   101    14    18
     3     7   102    15    19
     4     8   103    16    20
```



```
>> a(:,3:4) = -1
a =
     1     5    -1    -1    17
     2     6    -1    -1    18
     3     7    -1    -1    19
     4     8    -1    -1    20
```

```
>> a(3:end,:) = 42
a =
     1     5    -1    -1    17
     2     6    -1    -1    18
    42    42    42    42    42
    42    42    42    42    42
```

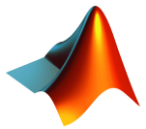
```
>> a(3:4,1:2) = [10 11 ; 12 13]
a =
     1     5    -1    -1    17
     2     6    -1    -1    18
    10    11    42    42    42
    12    13    42    42    42
```

행렬을 수정하는 방법

```
>> a = [a ones(4,1)]
a =
     1     5    -1    -1    17     1
     2     6    -1    -1    18     1
    10    11    42    42    42     1
    12    13    42    42    42     1
```

```
>> a = [a 2*ones(size(a,1),1)]
a =
     1     5    -1    -1    17     1     2
     2     6    -1    -1    18     1     2
    10    11    42    42    42     1     2
    12    13    42    42    42     1     2
```

```
>> a = [a; 5*(1:size(a,2))]
a =
     1     5    -1    -1    17     1     2
     2     6    -1    -1    18     1     2
    10    11    42    42    42     1     2
    12    13    42    42    42     1     2
     5    10    15    20    25    30    35
```



```
>> a = a(1:end-1, 1:end-2)
a =
     1     5    -1    -1    17
     2     6    -1    -1    18
    10    11    42    42    42
    12    13    42    42    42
```

```
>> a(end,:) = []
a =
     1     5    -1    -1    17
     2     6    -1    -1    18
    10    11    42    42    42
```

```
>> a(:,4:end) = []
a =
     1     5    -1
     2     6    -1
    10    11    42
```

```
>> a(5,5) = -100
a =
     1     5    -1     0     0
     2     6    -1     0     0
    10    11    42     0     0
     0     0     0     0     0
     0     0     0     0  -100
```

예제 - boxBlur

$$image = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 7 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\left\lfloor \frac{1+1+1+1+7+1+1+1+1}{9} \right\rfloor = 1$$

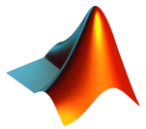
$$blurred = [1]$$

$$image = \begin{bmatrix} 36 & 0 & 18 & 9 \\ 27 & 54 & 9 & 0 \\ 81 & 63 & 72 & 45 \end{bmatrix}$$

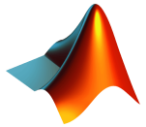
$$\left\lfloor \frac{36+0+18+27+54+9+81+63+72}{9} \right\rfloor = 40$$

$$blurred = [40 \quad 30]$$

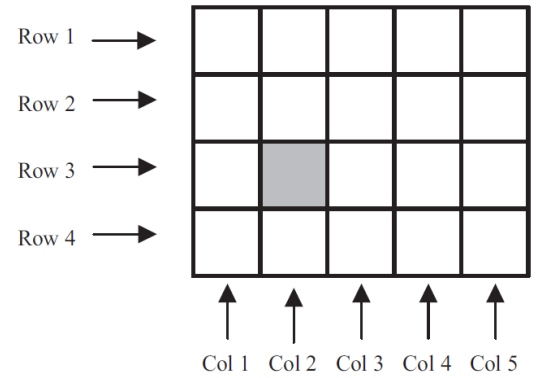
$$\left\lfloor \frac{0+18+9+54+9+0+63+72+45}{9} \right\rfloor = 30$$



문자열도 행렬이다.



문자열도 행렬이다.



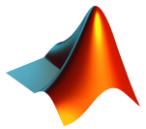

```
>> s = 'hongik';
>> size(s)
ans =
     1     6
>> s(2:4)
ans =
    'ong'
```

```
>> s = ['hongik'; 'matlab'; 'autumn'; 'yr2020']
s =
  4x6 char 배열
    'hongik'
    'matlab'
    'autumn'
    'yr2020'
>> s(2:4,3:end)
ans =
  3x4 char 배열
    'tlab'
    'tumn'
    '2020'
```

```
>> s1 = 'hongik';
>> s2 = ' ';
>> s3 = 'matlab';
>> s = [s1 s2 s3]
s =
    'hongik matlab'
```

- 각 행은 길이가 같은 문자열이어야 함
- 행렬이므로, 숫자행렬(numeric array)처럼 이어 붙일 수 있음
- 따옴표('')를 쓰지 않으면 변수명으로 인식
- 1과 '1'은 다름
- 작은 따옴표와 큰 따옴표는 다름
  - 'hongik' → char array
  - "hongik" → string

사용자 입력, 출력문

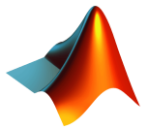


사용자 입력

- 사용자로부터 숫자 또는 문자 입력을 받을 수 있음
  - `age = input('Type your age: '); % 숫자로 받음`
  - `name = input('Type your name: ','s'); % 문자로 받음`
  - num2str: 숫자를 문자로 변환
  - str2double: 문자를 숫자로 변환

- 예제

```
game1=input('Enter the points scored in the first game ');
game2=input('Enter the points scored in the second game ');
game3=input('Enter the points scored in the third game ');
ave_points=(game1+game2+game3)/3
```



disp

```
>> a
a =
     1     5     9    13    17
     2     6    10    14    18
     3     7    11    15    19
     4     8    12    16    20
>> disp(a)
     1     5     9    13    17
     2     6    10    14    18
     3     7    11    15    19
     4     8    12    16    20
>> s
s =
  4x6 char 배열
    'hongik'
    'matlab'
    'autumn'
    'yr2020'
>> disp(s)
hongik
matlab
autumn
yr2020
```

```
>> help disp
disp - Display value of variable

    This MATLAB function displays the value of variable X without printing the
    variable name.
```

```
game1=input('Enter the points scored in the first game    ');
game2=input('Enter the points scored in the second game   ');
game3=input('Enter the points scored in the third game    ');
ave_points=(game1+game2+game3)/3;
disp(' ')
disp('The average of points scored in a game is:')
disp(' ')
disp(ave_points)
```

Display empty line.

Display text.

Display empty line.

Display the value of the variable ave_points.

fprintf – formatted print

\ : escape character

※ 주석

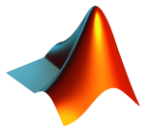
```
clear

% simple text print
fprintf('I am Groot.');

% using \t and \n
fprintf('I am Groot.');
fprintf('\tI am Quill.');
fprintf('I need a new line.\n');
fprintf('\t\tThere you go!\n')

% using %d and %s
name = 'Quill';
age = 38;
fprintf('I am %s. %d years old.\n', name, age);

% using %f and %g and %%
weight_old = 136.078;
weight_now = 104.326;
fprintf('I weighed %f kg once.\n',weight_old);
fprintf('But I am %g kg now.\n',weight_now);
fprintf('So I lost %g%% of my weight.\n', ...
    100*(1-weight_now/weight_old));
```

단순 문자열 출력

\t : 탭 (4칸 또는 8칸)
\n : 줄넘김

%s : 문자열 출력
%d : 정수 출력
%가 여러 개? → 순서대로 1:1 대응

%f : 실수 출력
%g : 실수 출력, 뒤이어 붙는 0이 없음
%% : % 문자 자체를 출력



fprintf – formatted print

```
% using field width and precision
fprintf('pi is about %5.2f.\n',pi)
fprintf('pi is about %10.2f.\n',pi)
fprintf('pi is about %10.5f.\n',pi)

% add heading zeros
fprintf('pi is about %010.3f.\n',pi)
fprintf('pi^2 is about %04d.\n', round(pi^2))

% align left
fprintf('pi is about %-10.3f.\n',pi)
```

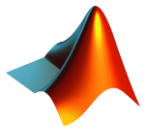
%5.2f : 전체 5자리, 소수점 이하 2자리
%10.2f : 전체 10자리, 소수점 이하 2자리
%10.5f : 전체 10자리, 소수점 이하 5자리

%010.3f : 전체 10자리, 소수점 이하 3자리,
10자리가 안되면 앞을 0으로 채움
%04d : 정수, 4자리가 안되면 앞을 0으로 채움

%-10.3f : 전체 10자리, 소수점 이하 3자리,
좌측정렬

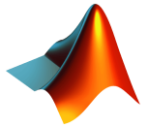
※ num2str도 동일한 포맷팅 가능

```
>> num2str(pi,'%010.5f')
ans =
    '0003.14159'
```



강의 요약

- 수식작성
  - 사칙연산, 거듭제곱(^), 각종 내장함수 (sqrt, exp, log, sin, …)
- 행렬
  - 매트랩에서는 모든 것이 행렬이다.
  - 행렬을 만드는 방법 (원소 직접 입력, 콜론, linspace, 이미 있는 행렬을 합침)
  - 행렬의 연산 (행렬곱, 원소별 곱셈, 나누기, 원소별 나눗셈)
  - 행렬을 다루는 내장함수들
  - 행렬의 인덱싱
  - 행렬을 수정하는 방법
- 문자열도 행렬이다.
- 사용자 입력 (input)
- 출력문 (fprintf, disp)



Q&A