

## 2장

# 배열의 생성

###



\$\$\$\$\$



1. 1차원/2차원 배열의 생성
2. 전치 연산자
3. 배열 원소의 주소 지정
4. 콜론 연산자 사용
5. 기존 변수에 원소 추가/제거
6. 배열 조작을 위한 내장 함수
7. 응용예제
8. 문자열과 문자열 변수



## 2.1 1차원 배열의 생성

MATLAB 에서의 1차원 배열  
생성을 알아보자.

# 1차원 배열의 생성

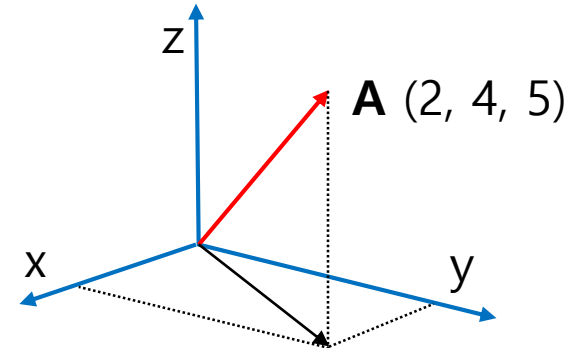
## 배열과 1차원 배열의 이용예

### ❖ 배열

- ▶ MATLAB이 데이터를 저장하고 다루기 위해 사용하는 기본적인 형태
- ▶ 행(row)이나 열(column), 또는 행과 열로 정렬된 수들의 나열

### ❖ 1차원 배열(벡터)의 예

- ▶ 3차원 공간상의 점 **A (2,4,5)**
  - MATLAB: [2 4 5], 또는 [2,4,5]
- ▶ 인구 증가 데이터
  - 두개의 집합 또는 벡터로 표시
  - year = [1984 1986 1988 1990 1992 1994 1996]
  - pop = [127 130 136 145 158 178 211]



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

# 1차원 배열의 생성

## 배열 생성

### ❖ 배열의 원소

- ▶ 값/ 함수를 포함하는 수식

### ❖ 배열의 생성 방법

- ▶ 꺾은 괄호 [ ] 안에 배열의 원소들을 직접 입력
- ▶ 외부 데이터 파일로부터 행렬을 읽어 들임
- ▶ 명령어나 자신이 만든 m 파일을 이용하여 생성

### ❖ 벡터는 대괄호 [ ] 사이에 원소 나열

Variable\_name = [ 벡터 원소들의 나열 ]

- ▶ 행(row)벡터: 콤마 ',' 나 " "공백으로 원소를 구분
- ▶ 열(column)벡터: 세미콜론 ';' 또는 Enter키로 원소를 구분
  - 행벡터 입력 후 전치(transpose)기호를 이용하는 방법도 있음.

# 1차원 배열의 생성

알고있는 집합으로 벡터 생성

## ▶ 행(row)벡터:

```
>> a = [5, 7, 2, 4 10 29]    % 공백/콤마로 원소 구분
>> b = [0, 0.5*pi, 1.5*pi]    % 수식표현 가능
>> c = [0 0.5 1.5]*pi        % c와 결과는 같다
```

## ▶ 열(column)벡터:

```
>> v = [3; 4; 5]
v = 3
    4
    5
```

```
>> v = [3
4
5]
v = 3
    4
    5
```

```
>> v = [3 4 5]'
v = 3
    4
    5
```

# 1차원 배열의 생성

## 간격이 일정한 벡터 생성

### ❖ 간격이 일정한 벡터 생성

Vector\_name = [ m : q : n ] 또는 Vector\_name = m : q : n

- ▶ m: 첫 번째 원소, q: 간격, n: 마지막 원소
  - 최종 원소는 n을 초과할 수 없다.
  - q 가 생략되면 1씩 증가
  - q 가 음수이면  $m > n$  이어야 함.

```
>> x = [1: 2: 13]
```

```
x =
```

```
1 3 5 7 9 11 13
```

```
>> x = 15: -3: 8
```

```
x =
```

```
15 12 9
```

```
>> x = [0: 0.5: 1]*pi
```

```
x =
```

```
0 1.5708 3.1416
```

```
>> x = -1: 0.5: 1.0
```

```
x =
```

```
-1.00 -0.50 0 0.50 1.50
```

# 1차원 배열의 생성

원소의 개수를 지정하여 간격이 일정한 벡터 생성

❖ 원소의 개수를 지정하여 간격이 일정한 벡터 생성

```
Variable_name = linspace(xi, xf, n)
```

- ▶ xi: 첫 번째 원소, xf: 최종원소, n: 갯수
  - 최종 원소는 항상 xf 임.
  - 원소는  $(xf - xi)/(n-1)$ 만큼씩 증가
  - n 을 생략하면 기본으로 100개 생성.

```
>> x = linspace(2, 14, 6) % 2부터 14까지 6개의 원소를 생성
x = 2.0000  4.4000  6.8000  9.2000  11.6000  14.0000
>> y = linspace( 0.1, 10) % 개수를 지정 안하면 100개 생성
y = 0.1000  0.2000  0.3000 ... 9.8000  9.9000  10.0000
```

- ▶ 참고: logspace(a,b,n):  $10^a$ 와  $10^b$ 사이에 n개의 로그적으로 같은 간격





## 2.2 2차원 배열의 생성

행렬(matrix)로도 불리는 2차원 배열의 생성 방법을 알아본다.

# 2차원 배열의 생성

## 2차원 배열(행렬)의 생성

### ❖ 2차원 배열

- ▶ 행렬(matrix)라고도 하며  $n$ 개의 행과  $m$ 개의 열을 가짐
- ▶ 이를  $m \times n$  행렬이라 함.
- ▶ 선형대수에서 매우 중요.
- ▶ 정방행렬(square matrix): 행과 열의 수가 동일

### ❖ 입력방법

- ▶ 행의 원소의 입력은 1차원 배열과 같음.
- ▶ 새로운 행을 만들 때에는 세미콜론 ';' 또는 **Enter** 사용.

```
Variable_name = [ 첫 번째 행의 원소들 ; 두 번째 행의 원소들 ;  
                  ...                ; 마지막 행의 원소들 ]
```

# 2차원 배열의 생성

## 2차원 배열(행렬)의 생성

### ❖ 입력 예

- ▶ 행의 원소의 입력은 1차원 배열과 같음.
- ▶ 새로운 행을 만들 때에는 세미콜론 ';' 또는 **Enter** 사용.

$$A = \begin{bmatrix} 2 & 4 & 10 \\ 16 & 3 & 7 \\ 8 & 12 & 35 \end{bmatrix}$$

```
>> A=[2 4 10; 16 3 7; 8 12 35]
```

A =

2	4	10
16	3	7
8	12	35

Semi  
colon

```
>> A=[2 4 10
```

```
16 3 7
```

```
8 12 35]
```

A =

2	4	10
16	3	7
8	12	35

Enter

# 2차원 배열의 생성

## 다양한 생성 방법

❖ 수식, 변수, linspace, 콜론 ':' 을 사용한 생성 예

```
>> x=4; y=2; z=8; % 세 변수의 정의
```

```
>> A=[x y z; sin(x/z) x^2 x+y]
```

```
A =
```

```
4.0000 2.0000 8.0000
```

```
0.4794 16.0000 6.0000
```

```
>> A=[1:2:11; 0:5:25;
```

```
linspace(10, 60, 6); 67 2 43 68 4 13]
```

```
A=
```

```
1 3 5 7 9 11
```

```
0 5 10 15 20 25
```

```
10 20 30 40 50 60
```

```
67 2 43 68 4 13
```

```
>> A=[2 4 10;linspace(8, -2, 3)]
```

```
A =
```

```
2 4 10
```

```
8 3 -2
```

```
>> A=[2 4 10; 8: -5: -2 ]
```

```
A =
```

```
2 4 10
```

```
8 3 -2
```

```
>> A=[A; 10 20 30]
```

```
A =
```

```
2 4 10
```

```
8 3 -2
```

```
10 20 30
```

# 2차원 배열의 생성

## 다양한 생성 방법

```
>> r1=[2, 4, 10]; r2=[16, 3, 7];  
>> A=[r1; r2]  
A =  
     2     4    10  
    16     3     7  
  
>> A=[[2, 4, 10]; [16, 3, 7]] % A=[2, 4, 10; 16, 3, 7].  
A =  
     2     4    10  
    16     3     7  
  
>> v=[r1 r2]  
v = 2     4    10    16     3     7  
  
>> cd=6; h=4;  
>> Mat=[3, cd*h, cos(pi/3) ; h^2, sqrt(h*h/cd), 14]  
Mat=  
    3.0000    24.0000    0.5000  
   16.0000    1.6330   14.0000
```

# 2차원 배열의 생성

## zeros, ones, eye 명령

### ❖ 특수한 원소를 갖는 행렬생성

- ▶ **zeros(m, n)** : 원소가 모두 0인  $m \times n$  크기의 행렬
- ▶ **ones(m, n)** : 원소가 모두 1인  $m \times n$  크기의 행렬
- ▶ **eye(n)** : 대각선 원소만 1이고 나머지 원소들은 모두 0인  $n \times n$ 의 행렬

```
>> Z = zeros(2, 3)
```

```
Z =
```

```
0 0 0
0 0 0
```

```
>> O = ones(2, 2)
```

```
O =
```

```
1 1
1 1
```

size(Z): 배열 Z의 행과 열의 크기 반환

```
>> I = eye(3)
```

```
I =
```

```
1 0 0
0 1 0
0 0 1
```

```
>> B = ones(size(Z))
```

```
B =
```

```
1 1 1
1 1 1
```

# MATLAB 변수에 대한 유의사항

## 배열관련 고려사항

- ❖ MATLAB의 모든 변수들은 배열
  - ▶ 스칼라도 원소가 하나인 배열
  - ▶ 벡터도 행이나 열이 하나인 배열
- ❖ 원소를 할당하기 전에 배열의 크기를 정의 필요
  - ▶ 변수(스칼라, 벡터, 행렬)는 변수가 할당될 때의 입력에 의해 정의됨.
- ❖ 행렬 변수의 크기나 유형은 마음대로 변경 가능
  - ▶ 스칼라 → 벡터, 또는 행렬
  - ▶ 벡터 → 스칼라, 다른 크기의 벡터, 또는 행렬
  - ▶ 행렬 → 다른 크기의 행렬, 벡터, 스칼라.

# 전치(transpose) 연산자

## transpose 연산자

### ❖ 전치연산자

- ▶ 행과 열을 서로 바꾸는 연산
- ▶ 벡터나 행렬 뒤에 따옴표(')를 붙임

```
>> v=[10; 20; 30];  
>> vt = v'  
vt =  
    10    20    30  
>> x=[10 20 30]; xt = x'  
xt =  
     10  
     20  
     30
```

```
>> A=[1 2 3; 10 20 30]  
A =  
     1     2     3  
    10    20    30  
>> B=A'  
B =  
     1    10  
     2    20  
     3    30
```



# 배열 원소의 주소 지정

## 벡터의 경우

- ❖ 배열이나 벡터의 한 개 또는 여러 개의 원소에 직접 접근방법
  - ▶ 일부 또는 하나의 원소값을 참조/변경 시 유용
- ❖ 벡터의 원소 주소
  - ▶ 벡터이름 뒤 괄호 () 와 원소의 순서

**Vector\_name(원소의 순서)**

▶ 예

```
>> v=[10 20 30 40];  
>> v(1)      % 내용 참조  
ans =  
      10  
>> v(3) =50; % 내용 변경
```

```
>> v(4)=v(1)+v(2); % 수식  
>> v  
v =  
      10  20  50  30
```

# 배열 원소의 주소 지정

## 벡터의 경우

```
>> VCT=[35 46 78 23 5 14 81 3 55]
VCT =
    35    46    78    23     5    14    81     3    55
>> VCT(4)           % 2 번째 원소 참조
ans =
    23
>> VCT(6)=273       % 6 번째 원소에 변경
VCT=
    35    46    78    23     5   273    81     3    55
>> VCT(2)+VCT(8)     % 2 번째 8 번째 원소 참조
ans =
    49
>> VCT(5)^VCT(8)+sqrt(VCT(7)) % 수학적 구성
ans =
   134
```

# 배열 원소의 주소 지정

## 행렬의 경우

- ❖ 배열의 주소는 원소가 있는 열(row)과 행(column)의 위치
- ❖ 배열의 원소주소

**Matrix\_name(r, c)**

▶ r: 행, c: 열

▶ 예: M(1,1)=5, M(2,2)=1, M(2,3)=7

$$M = \begin{bmatrix} 5 & 10 & 9 \\ 18 & 1 & 7 \end{bmatrix}$$

```
>> A = [3 11 6 5; 4 7 10 2;  
        13 9 0 8];
```

```
>> A(3,2) = 55
```

```
A = 3 11 6 5  
    4 7 10 2  
    13 55 0 8
```

```
>> A(2,1) = A(2,1)-A(2,4)
```

```
A = 3 11 6 5  
    2 7 10 2  
    13 55 0 8
```



## 2.6 콜론(:)을 사용하여 배열 원소의 주소 지정

벡터나 행렬에서 콜론(:)을 이용하여 어떤 범위의 원소들의 주소를 지정 방법에 대하여 알아본다.

# 콜론(:)을 사용한 주소 지정

## 콜론(:)을 사용하여 배열 원소의 주소 지정

❖ 콜론(:)으로 배열의 원소 주소를 지정

❖ 벡터

- ▶  $\mathbf{v}(:)$  : 행 또는 열 벡터  $\mathbf{v}$ 의 모든 원소이며 **열 벡터**로 반환
- ▶  $\mathbf{v}(m:n)$  : 행 또는 열 벡터  $\mathbf{v}$ 의  $m$  부터  $n$ 까지의 원소들을 의미
  - 참고:  $n$ 을 잘 모르면 `end`를 사용. ( `end` 는 `length(v)`를 의미).

```
>> v=[10 20 30 40 50];  
>> v(3)  
ans = 30  
>> w=v(2:4)  
w =  
    20    30    40  
>> length(v)  
ans = 5
```

```
>> v=[10 20 30 40 50];  
>> x=v(2:end)  
x = 20    30    40    50  
>> y=v(2: length(v))  
y = 20    30    40    50  
>> v(3:end)=0  
v =  
    10    20     0     0     0
```

# 콜론(:)을 사용한 주소 지정

콜론(:)을 사용하여 배열 원소의 주소 지정

## ❖ 예제

```
>> v=[10 20 30 40 50];  
>> w = v(3 :-1 : 1)      % v(3), v(2), v(1) 원소로 구성된 벡터 생성  
w =  
    30    20    10  
>> y=v(2:3:5)           % v(2), v(5) 원소로 구성된 벡터 생성  
y =  
    20    50  
>> z =v( [1 3 4] )      % v(1), v(3), v(4) 원소로 구성된 벡터 생성  
z =  
    10    30    40  
>> vt=z( :)             % 콜론은 열 벡터를 반환하여 전치와 동일 효과  
vt =  
    10  
    30  
    40
```

# 콜론(:)을 사용한 주소 지정

## 콜론(:)을 사용하여 배열 원소의 주소 지정

### ❖ 행렬

- ▶ 콜론(:)으로 행렬 A의 일부 행이나 열을 지정
- ▶  $A(:, n)$  : 행렬 A의 n번째 열(column)의 모든 원소들
- ▶  $A(n, :)$  : 행렬 A의 n번째 행(row)의 모든 원소들
- ▶  $A(:, m:n)$  : 행렬 A의 m~n번째 열(column)의 모든 원소들
- ▶  $A(m:n, :)$  : 행렬 A의 m~n번째 행(row)의 모든 원소들
- ▶  $A(m:n, p:q)$ : m~n번째 행과 p~q번째 열에 해당하는 원소들

$$A = \begin{bmatrix} 2 & 4 & 13 & 9 \\ 16 & 3 & 7 & 11 \\ 8 & 21 & 6 & 5 \end{bmatrix}$$

D

C

```
>> C = A(2:3, 1:3)
```

```
C = 16    3    7
```

```
     8   21    6
```

```
>> D=A(:, 2)
```

```
D = 4
```

```
     3
```

```
    21
```

# 콜론(:)을 사용한 주소 지정

콜론(:)을 사용하여 배열 원소의 주소 지정

```
>> B=[ 9:-2:1; zeros(1, 3) 1 2; ones(2, 5)*2 ]
```

B =

9	7	5	3	1
0	0	0	1	2
2	2	2	2	2
2	2	2	2	2

```
>> B(:, end) = 1
```

B =

9	7	5	3	1
0	0	0	1	1
2	2	2	2	1
2	2	2	2	1

```
>> C= B( [1 3], [1, 3:4] )
```

C =

9	5	3
2	2	2



# 콜론(:)을 사용한 주소 지정

콜론(:)을 사용하여 배열 원소의 주소 지정

## ❖ 행렬의 벡터로의 변환 예제

```
>> D=[11 12 13;21 22 23]
```

```
D =
```

```
    11    12    13  
    21    22    23
```

```
>> v = D(:)
```

```
v =
```

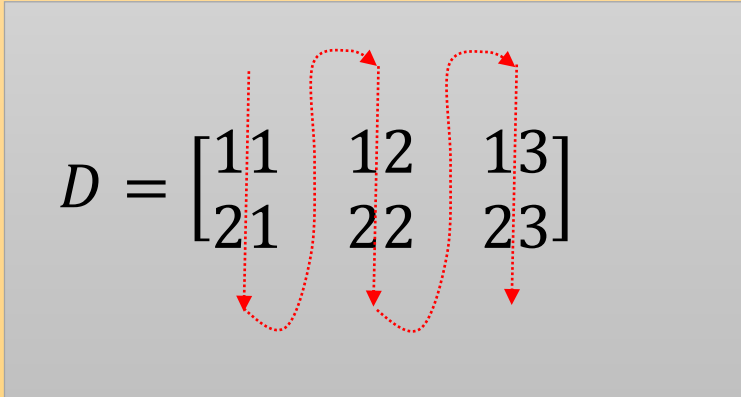
```
    11  
    21  
    12  
    22  
    13  
    23
```

```
>> v=v'
```

```
v =
```

```
    11    21    12    22    13    23
```

D(:) 은 행렬을 열벡터로


$$D = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{bmatrix}$$

# 콜론(:)을 사용한 주소 지정

콜론(:)을 사용하여 배열 원소의 주소 지정

```
>> A=[2 4 10 13; 16 3 7 1;8 4 9 21]
```

```
A =
```

```
    2    4   10   13  
   16    3    7    1  
    8    4    9   21
```

```
>> indx = [1 2];
```

```
>> B = A(indx, indx)           % A([1 2], [1 2])
```

```
B =
```

```
    2    4  
   16    3
```

```
>> v=[ A(3), A(4), A(7) ]      % 벡터로 변형 했을 때 순서대로
```

```
v =
```

```
    8    4   10
```

```
>> size(A), length(A)         % length(A)는 max(size(A))와 동일
```

```
ans = 3    4
```

```
ans = 4
```



## 2.7/8 기존 변수에 원소 추가/제거

벡터나 행렬로 존재하는 변수에 원소를 추가하여 확장하거나 제거하여 축소 시키는 방법에 대하여 알아본다.

# 벡터의 확장

## 벡터의 원소추가

❖ MATLAB에서는 배열을 미리 선언 불요

▶ 언제든지 배열의 크기 변경이 가능

```
>> DF=1:4
```

% 벡터 DF의 정의

```
DF =
```

```
1 2 3 4
```

```
>> DF(5:10)=10:5:35
```

% 5 번째부터 6개의 원소 추가

```
DF =
```

```
1 2 3 4 10 15 20 25 30 35
```

```
>> AD=[5 7 2]
```

% 벡터 AD 정의

```
AD =
```

```
5 7 2
```

```
>> AD(8)=4
```

% 4~7번째 원소가 0으로 자동 할당

```
AD =
```

```
5 7 2 0 0 0 0 4
```

```
>> AR(5)=24
```

% 4번째 원소까지 0이 자동 할당

```
AR = 0 0 0 0 24
```

# 벡터의 확장

## 벡터의 확장

```
>> RE=[3 8 1 24];
>> GT=4:3:16;
>> KNH = [RE GT]      % RE와 GT를 붙여 새로운 벡터 KNH를 정의.
KNH =
    3    8    1   24    4    7   10   13   16
>> v=10                % v를 스칼라 변수로 지정
v = 10
>> v(end+1) = 20        % 스칼라 변 → 두 원소 벡터로 확장
v =
    10    20
>> v(end+1 : length(v)+3)=[3 4 5] % 5원소 벡터로 확장
v =
    10    20    3    4    5
>> n = length(v)        % 벡터의 길이(원소의 개수)를 확인
n = 5
>> v(n)=50              % 벡터의 마지막 원소를 50으로 대체한다.
ans = 10    20    3    4    50
```

# 행렬의 확장

## 행렬에 원소 추가

### ❖ 행렬의 확장

- ▶ 행과 열을 추가 할 수 있지만 기존 크기와 크기가 일치 해야 함.

```
>> E=[1 2 3 4; 5 6 7 8]
E = 1 2 3 4
     5 6 7 8
>> E(3, :)= [10:4:22] % 3행에 [10 14 18 22] 추가
E = 1 2 3 4
     5 6 7 8
    10 14 18 22
>> K=eye(3);
K = 1 0 0
     0 1 0
     0 0 1
>> G=[E K] % 행렬 E에 행렬 K를 추가함.
G =
     1     2     3     4     1     0     0
     5     6     7     8     0     1     0
    10    14    18    22     0     0     1
```

```
>> E=[1 2 3 4; 5 6 7 8];
>> E(3, 4)=1
E = 1 2 3 4
     5 6 7 8
     0 0 0 1
>> H(3,4)=8
H =
     0     0     0     0
     0     0     0     0
     0     0     0     8
```

# 행렬의 확장

## 행렬에 원소 추가

```
>> A=[11 12 13 14;21 22 23 24]
24]
```

A =

11	12	13	14
21	22	23	24

```
>> A(3, :) = 31:34
```

A =

11	12	13	14
21	22	23	24
31	32	33	34

```
>> B=ones(3, 1);
```

```
>> A=[A B]
```

A =

11	12	13	14	1
21	22	23	24	1
31	32	33	34	1

```
>> A(:, 3:end)=[ ] % 원소 삭제
```

A =

11	12
21	22
31	32

```
>> A(2, 4) = 50 % 크기를 벗어난 원소 삽입
```

A =

11	12	0	0
21	22	0	50
31	32	0	0

```
>> clear; % 모든 변수 삭제
```

```
>> B(2,3) = 5 % 크기를 벗어난 원소 삽입
```

B =

0	0	0
0	0	5

# 벡터의 축소

## 공벡터 [] 이용 원소제거

❖ 공(empty, null)벡터: 원소가 없으며 []로 표시

▶ 원소에 공벡터를 할당하면 삭제 효과로 축소 가능

```
>> v=[]           % v에 공벡터 할당
v = []
>> length(v)      % 벡터 v의 원소 갯수
ans = 0
>> v=[1  3  100  5  7  200  300  400];
>> v(3)=[]        % v(3)에 공벡터 할당
v =
     1     3     5     7    200    300    400
>> v( 5 : end)=[] % 5~7에 공벡터 할당
v =
     1     3     5     7
>> A=[5  11  4  9; 4 0 26 10; 56 1  5  89]
```

```
A =
     5    11     4     9
     4     0    26    10
    56     1     5    89
>> A(:, 2:3) = []
A =
     5     9
     4    10
    56    89
```





## 2.9 배열 조작을 위한 내장함수

배열을 조작하거나 관리하는  
내장함수에 대하여 알아본다.

# 배열조작을 위한 내장함수

## 요약

함수	설명	예
<b>length(A)</b>	벡터 A의 원소 개수를 보여준다.	<pre>&gt;&gt;A=[5 9 2 4]; &gt;&gt;length(A) ans = 4</pre>
<b>size(A)</b>	크기가 $m \times n$ 인 배열 A의 크기를 행 벡터 [m,n]으로 돌려준다.	<pre>&gt;&gt;A=[6 1 4 0 12; 5 19 6 8 2]; &gt;&gt;size(A) ans = 2 5</pre>
<b>reshape(A,m,n)</b>	크기가 $r \times s$ 인 행렬 A를 $m \times n$ 의 행렬로 재정렬시킨다. r과 s의 곱은 m과 n의 곱과 반드시 같아야 한다.	<pre>&gt;&gt;A=[5 1 6; 8 0 2] A = 5 1 6     8 0 2 &gt;&gt;B = reshape(A,3,2) B = 5 0     8 6     1 2</pre>
<b>diag(v)</b>	v가 벡터일 때, v의 원소를 대각선 원소로 갖는 정방행렬을 생성한다.	<pre>&gt;&gt;v=[7 4 2]; &gt;&gt;A = diag(v) A = 7 0 0     0 4 0     0 0 2</pre>
<b>diag(A)</b>	A가 행렬일 때, A의 대각선 원소를 원소로 갖는 벡터를 생성한다.	<pre>&gt;&gt;A=[1 2 3; 4 5 6; 7 8 9]; &gt;&gt;vec = diag(A) vec = 1       5       9</pre>

# 배열조작을 위한 내장함수

## length

❖ length(v):

- ▶ v가 벡터: 원소의 갯수 반환
  - 벡터의 크기를 구하면  $\sqrt{v'v}$ 로 구함.
- ▶ v가 행렬: 행과 열중 큰 값을 반환 cf. size(A)

```
>> v=[2, -5, 4, -3]; n=length(v)
ans = 4
>> A=[ 11 12 13 14; 21 22 23 24]; length(A)
ans = 4
>> mag = sqrt(v*v')
ans = 7.3485
```

size(Z): 배열 Z의 행과 열의 크기 반환

# 배열조작을 위한 내장함수

## 기타 유용한 함수

### ❖ sum(v):

- ▶ v가 벡터: 원소의 총합 반환
- ▶ v가 행렬: v의 각 열들의 합을 행벡터로 반환

```
>> v=[5 3 9 2]; sum(v)  
ans = 19  
>> A=[1:2:5; 11:2:15; 21:2:25]; S=sum(A)  
S = 33 39 45
```

### ❖ max(v), min(v):

- ▶ v가 벡터: 원소 중 가장 큰 값/작은 값을 반환
- ▶ v가 행렬: v의 각 열 중 가장 큰 값/작은 값을 행벡터로 반환

```
>> v=[5 3 9 2]; max(v)  
ans = 9  
>> A=[6 2; -4 8; 3 10]; max(A)  
ans = 6 10
```

# 배열조작을 위한 내장함수

## 기타 유용한 함수

### ❖ sort(v):

- ▶ v가 벡터: 원소를 오름차순으로 정렬하여 반환
- ▶ v가 행렬: v의 각 열별로 오름차순으로 정렬하여 반환

```
>> v=[5 3 9 2]; sort(v)
ans = 2 3 5 9
>> A=[3 9 1 5; 8 4 6 7; 5 7 3 4]; sort(A)
ans = 3 4 1 4
      5 7 3 5
      8 9 6 7
```

- ▶ 내림차순 : sort(A, 'descend')

### ❖ find(v):

- ▶ v가 벡터: 원소 중 0이 아닌 원소들의 주소를 반환
- ▶ v가 행렬: v의 원소중 0이 아닌 원소의 행, 열, 그리고 값을 반환
  - [r, c, v]=find(A)
- ▶ find(v>100) 도 가능



## 2.8 응용예제

다양한 공학 응용 예제를 통하여 이해의 깊이를 더한다.

# 예제 2.1

## 4x5의 행렬 생성

❖ ones, zeros 명령어를 이용하여 첫 두 행은 0이고 다음 두 행은 1인 4x5 행렬을 생성하라.

❖ 방법 1

```
>> A(1:2,:)=zeros(2,5)
```

A =

0	0	0	0	0
0	0	0	0	0

```
>> A(3:4,:)=ones(2,5)
```

A =

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1

방법2

```
>> A=[zeros(2,5); ones(2,5)]
```

A =

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1

# 예제 2.2

## 6x6의 행렬 생성

- ❖ 6×6 행렬에서 가운데 두 행과 가운데 두 열의 원소가 1이고 나머지 원소는 모두 0인 행렬을 생성하라.

```
>> AR=zeros(6,6)
```

```
AR =
```

```
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

```
>> AR(3:4,:)=ones(2,6)
```

```
AR =
```

```
0 0 0 0 0 0
0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0
```

```
>> AR(:,3:4)=ones(6,2)
```

```
AR =
```

```
0 0 1 1 0 0
0 0 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 1 0 0
0 0 1 1 0 0
```



# 예제 2.3

## 행렬의 조작

- ❖ 명령어 창에서 다음 세 배열 **A**, **B**, **v**를 만들고, 명령어 한 개로 **A**의 1번째 행과 3번째 행의 마지막 4개의 열을 **B**의 첫 두 행의 첫 네 열로 대체하고 **A**의 4번째 행의 마지막 네개의 열을 **v**의 5~8번째 원소로 대체하고 마지막으로 **A**의 다섯 번째 행의 마지막 4개의 열을 **B**의 3행에서 2~5번째 열의 원소들로 바꾼다..

Initial matrices and replacement mapping:

$$A = \begin{bmatrix} 2 & 5 & 8 & 11 & 14 & 17 \\ 3 & 6 & 9 & 12 & 15 & 18 \\ 4 & 7 & 10 & 13 & 16 & 19 \\ 5 & 8 & 11 & 14 & 17 & 20 \\ 6 & 9 & 12 & 15 & 18 & 21 \end{bmatrix}$$
$$B = \begin{bmatrix} 5 & 10 & 15 & 20 & 25 & 30 \\ 30 & 35 & 40 & 45 & 50 & 55 \\ 55 & 60 & 65 & 70 & 75 & 80 \\ 99 & 98 & 97 & 96 & 95 & 94 & 93 & 92 & 91 \end{bmatrix}$$
$$v = [99 \ 98 \ 97 \ 96 \ 95 \ 94 \ 93 \ 92 \ 91]$$

Replacement mapping (indicated by arrows and boxes):

- Row 1 of A (last 4 columns) ← Row 1 of B (first 4 columns)
- Row 3 of A (last 4 columns) ← Row 2 of B (first 4 columns)
- Row 4 of A (last 4 columns) ← v (elements 5 to 8)
- Row 5 of A (last 4 columns) ← Row 3 of B (columns 2 to 5)

```
>> A=[ 2:3:17; 3:3:18; 4:3:19; 5:3:20; 6:3:21 ]
```

A =

2	5	8	11	14	17
3	6	9	12	15	18
4	7	10	13	16	19
5	8	11	14	17	20
6	9	12	15	18	21

# 예제 2.3

## 행렬의 조작

```
>> B=[5:5:30; 30:5:55; 55:5:80 ]
```

```
B =
```

5	10	15	20	25	30
30	35	40	45	50	55
55	60	65	70	75	80

```
>> v=[99:-1:91]
```

```
v =
```

99	98	97	96	95	94	93	92	91
----	----	----	----	----	----	----	----	----

```
>> A([1 3 4 5], 3:6)=[ B([1 2], 1:4); v(5:8); B(3,2:5)]
```

```
A =
```

2	5	5	10	15	20
3	6	9	12	15	18
4	7	30	35	40	45
5	8	95	94	93	92
6	9	60	65	70	75

A =	2	5	8	11	14	17
	3	6	9	12	15	18
	4	7	10	13	16	19
	5	8	11	14	17	20
	6	9	12	15	18	21



## 2.10 문자열과 문자열 변수

문자열의 생성, 특징, 사용  
방법에 대하여 알아본다.

# 문자열과 문자열 변수

## 문자열(String)

### ❖ 문자열:

- ▶ 작은 따옴표로 묶은 문자들의 배열로 문자, 숫자, 기호와 공백 등을 포함 가능
- ▶ 예) 'class', 'test', 'math%1', {'mech:05!'}

### ❖ 문자열 내에 작은 따옴표 ' 를 사용하는 경우

- ▶ 작은 따옴표를 두 번 연달아 표시. 예) 'I'm a student'

### ❖ 자동 인식:

- ▶ 문자열을 입력을 위해 작은 따옴표를 처음 표시하면 글자 색이 **적갈색**으로 변하며, 입력 완료하면 글자 색이 **자주색**으로 변함.

### ❖ 변수에 저장

- ▶ 각 글자에 해당하는 2바이트 코드가 숫자 배열처럼 저장
- ▶ 한 줄로 된 문자열은 글자(공백도 포함) 수와 같은 원소의 행벡터로 간주.

# 문자열과 문자열 변수

## 문자열의 용도

❖ MATLAB에서 문자열의 용도 예 :

- ▶ 출력 명령어(4장)에서 텍스트 메시지를 표시할 때
- ▶ plot 명령어에서 출력할 그래프의 형식을 지정할 때(5장),
- ▶ 함수의 입력 인자(6장).
- ▶ 그래프의 형식을 지정(축의 라벨, 제목, 설명문)할 때
- ▶ 이때는 문자열 내의 문자들이 특정 폰트와 크기, 위치, 색을 갖도록 형식 지정 가능

# 문자열과 문자열 변수

## 문자열 변수 예

### ❖ 행벡터로 간주

- ▶ 글자 개수와 같은 개수의 원소를 갖는 행벡터
- ▶ 벡터의 주소 지정을 이용하여 각 원소에 접근 가능.

```
>> str='MATLAB is a high-performance language.'
str = MATLAB is a high-performance language.
>> size(str)                % str의 글자수 (벡터의 원소수)
ans = 1    38
>> name = str(1:6)          % 벡터처럼 참조
name = MATLAB
>> str(1:6)= ' C++'         % 크기 불일치
??? In an assignment A(:) = B, the number of elements in A and B
must be the same.
>> str(4:6)=[ ]; str(1:3)='C++' % 공벡터를 이용하여 소거
str =
    C++ is a high-performance language.
```

# 문자열과 문자열 변수

## 사용 예

```
>> str
```

```
str =
```

```
C++ is a high-performance language.
```

```
>> str(1:3)='MAT'
```

```
str =
```

```
MAT is a high-performance language.
```

```
>> str=[str(1:3), 'LAB ', str(4:end)]
```

```
str =
```

```
    MATLAB  is a high-performance language.
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
str	1x38	76	char	

```
>> str(end:-1:30)
```

```
ans = .egaugnal
```

```
% language.의 역순
```

# 문자열과 문자열 변수

## 행렬로 만들기

❖ 문자열도 행렬처럼 여러 행을 가질 수 있음.

- ▶ 단, 각 행의 길이는 모두 같아야 하기 때문에 글자 수를 다른 경우 공백으로 맞춤.
- ▶ char(str1, str2, str3) 함수를 사용하면 자동으로 길이를 맞춤.

```
>> S = ['My'; 'name'; 'is Jina.']  
??? Error using ==> vertcat  
All rows in the bracketed expression must have the same number  
of columns.  
>> length(S(3, :))  
ans = 8  
>> S = ['My      '; 'name    '; 'is Jina.'] % 공백으로 행의 길이를 맞춤  
S =  
My  
name  
is Jina.  
>> S=char('My', 'name', 'is Jina.') % char가 자동으로 행의 길이를 맞춤  
S =  
My  
name  
is Jina.
```





Thank you!

2 장 끝