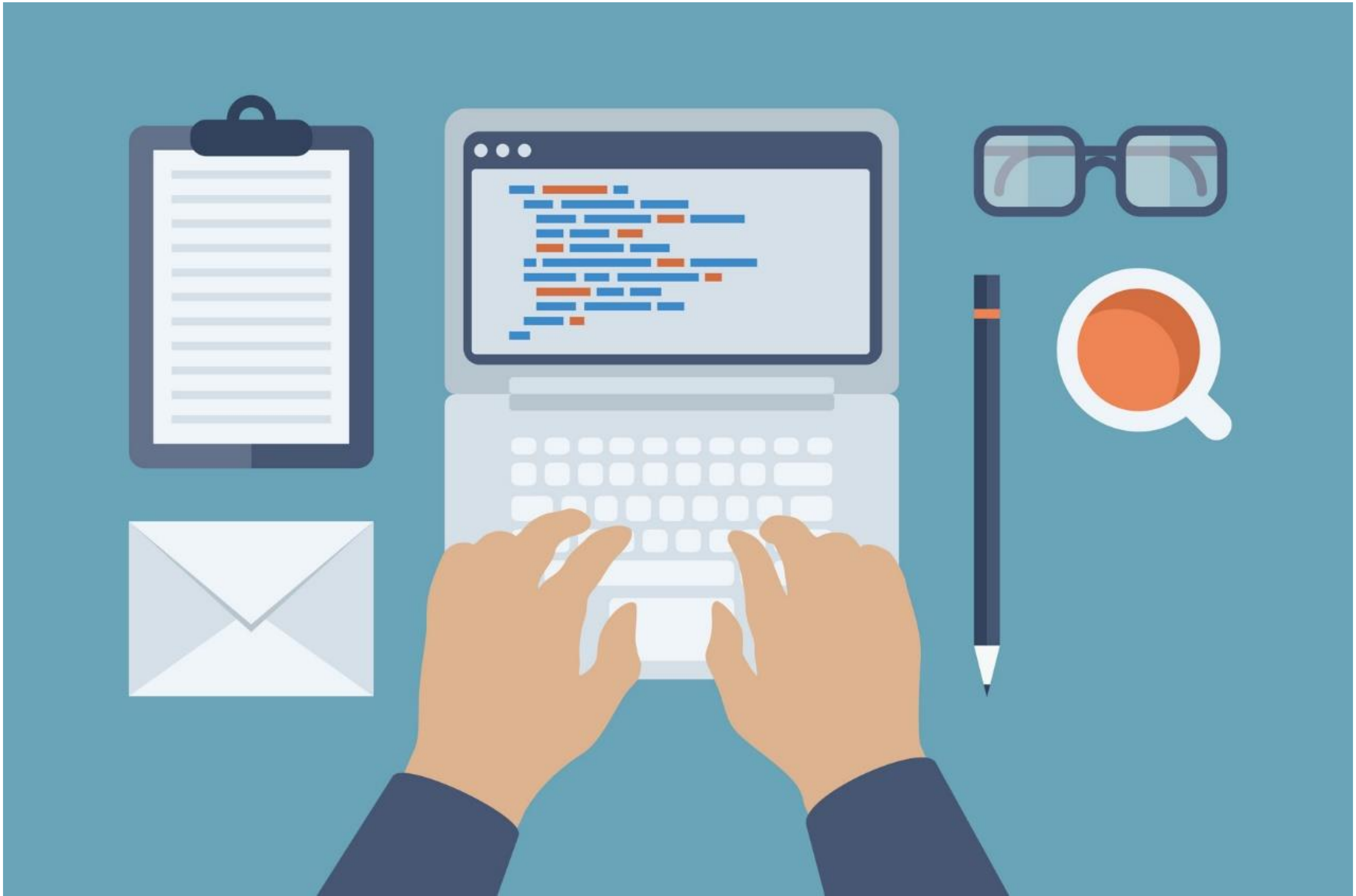
Computer Programming



CONTENTS

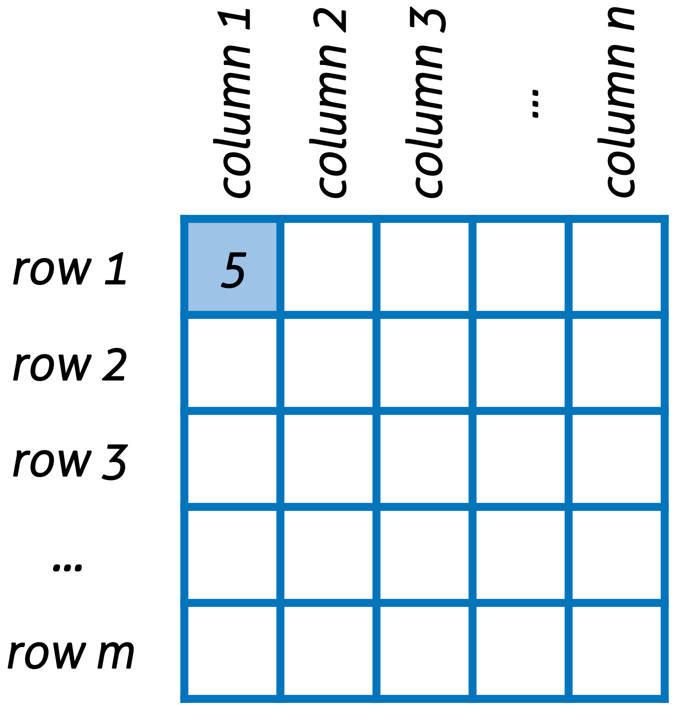
chapter 02: CREATING ARRAYS

SECTION	DETAILLE
SECTION 01	CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)
SECTION 02	CREATING A TWO-DIMENSIONAL ARRAY (MATRIX)
SECTION 03	NOTES ABOUT VARIABLES IN MATLAB
SECTION 04	THE TRANSPOSE OPERATOR
SECTION 05	ARRAY ADDRESSING
SECTION 06	USING A COLON : IN ADDRESSING ARRAYS
SECTION 07	ADDING ELEMENTS TO EXISTING VARIABLES
SECTION 08	DELETING ELEMENTS
SECTION 09	BUILT-IN FUNCTIONS FOR HANDLING ARRAYS
SECTION 10	STRINGS AND STRINGS AS VARIABLES
SECTION 11	PROBLEMS

MATLAB ARRAYS:

uses to store and manipulate data

MATLAB is an abbreviation for "MATRIX LABORATORY"

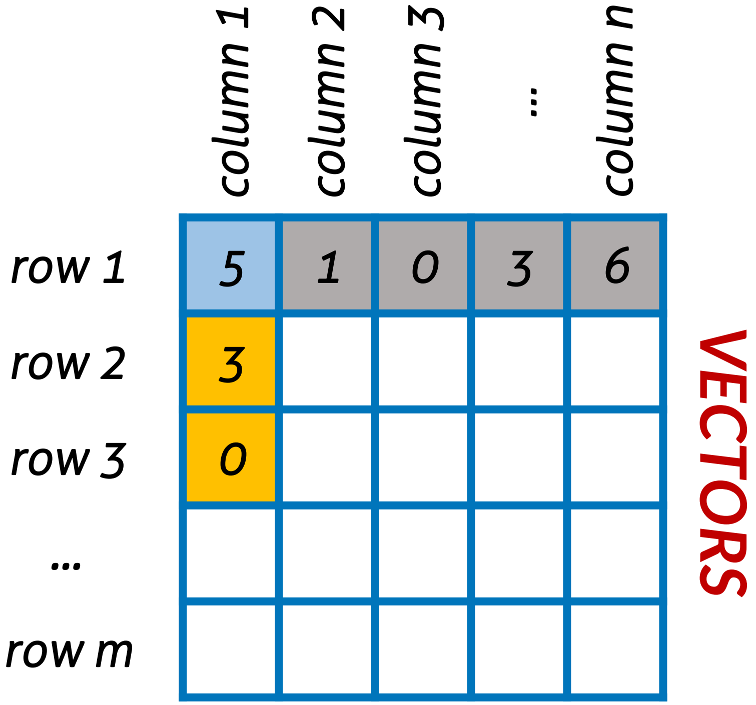


	column 1	column 2	column 3	...	column n
row 1	5				
row 2					
row 3					
...					
row m					

MATLAB ARRAYS:

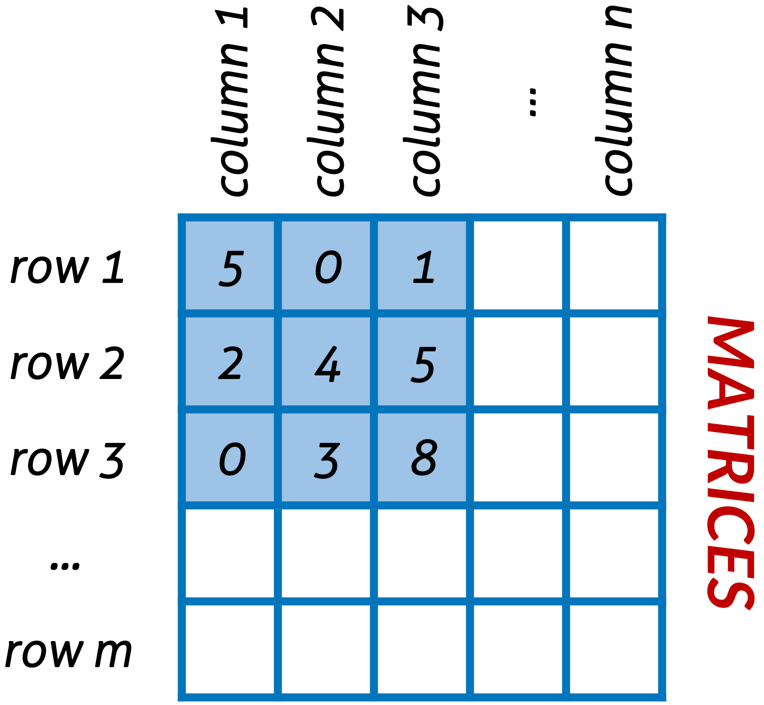
uses to store and manipulate data

MATLAB is an abbreviation for "MATRIX LABORATORY"



	column 1	column 2	column 3	…	column n
row 1	5	1	0	3	6
row 2	3				
row 3	0				
…					
row m					

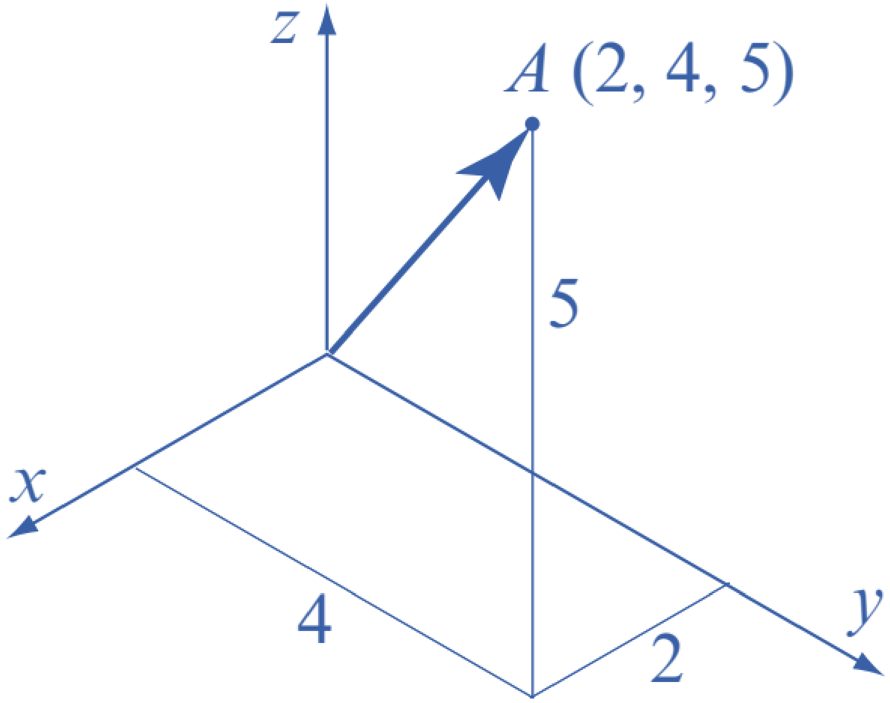
VECTORS



	column 1	column 2	column 3	…	column n
row 1	5	0	1		
row 2	2	4	5		
row 3	0	3	8		
…					
row m					

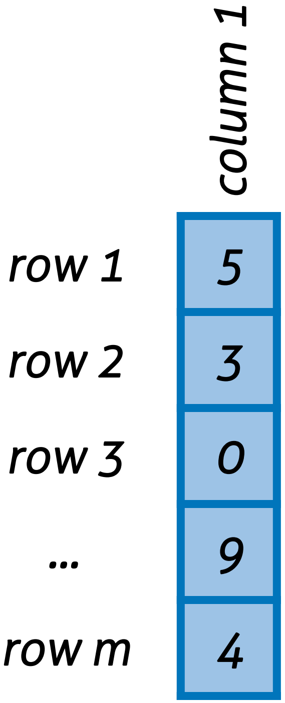
MATRICES

CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)

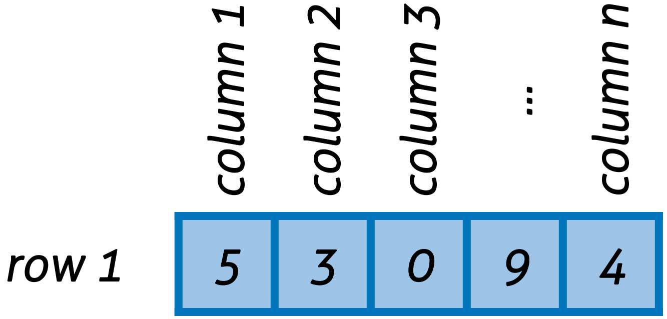


MONTH	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*
TEMPERATURE	*6*	*7*	*10*	*13*	*17*	*20*	*22*	*21*	*19*	*14*	*10*	*7*

CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)



	column 1
row 1	5
row 2	3
row 3	0
…	9
row m	4



	column 1	column 2	column 3	…	column n
row 1	5	3	0	9	4

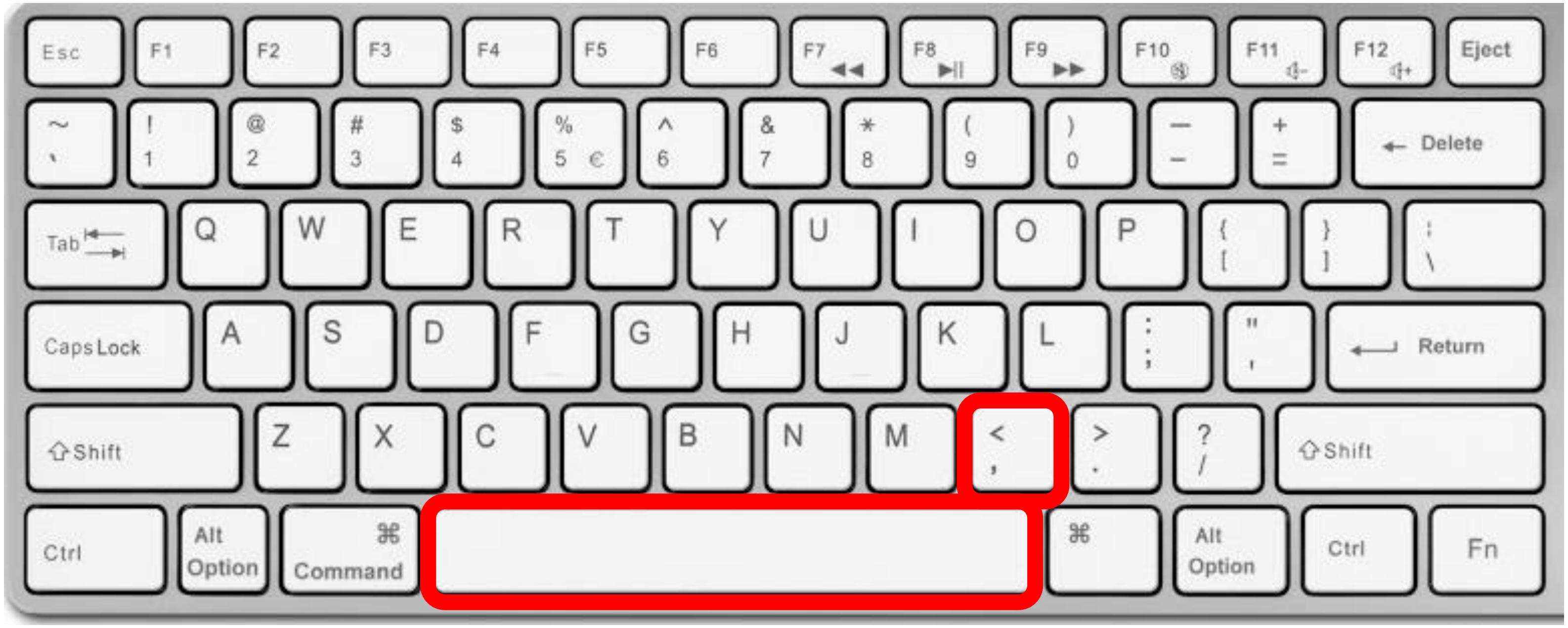
CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)

creating a vector from a known list of numbers:

square brackets [ ]

variable_name = [*type vector elements*]

ROW VECTOR



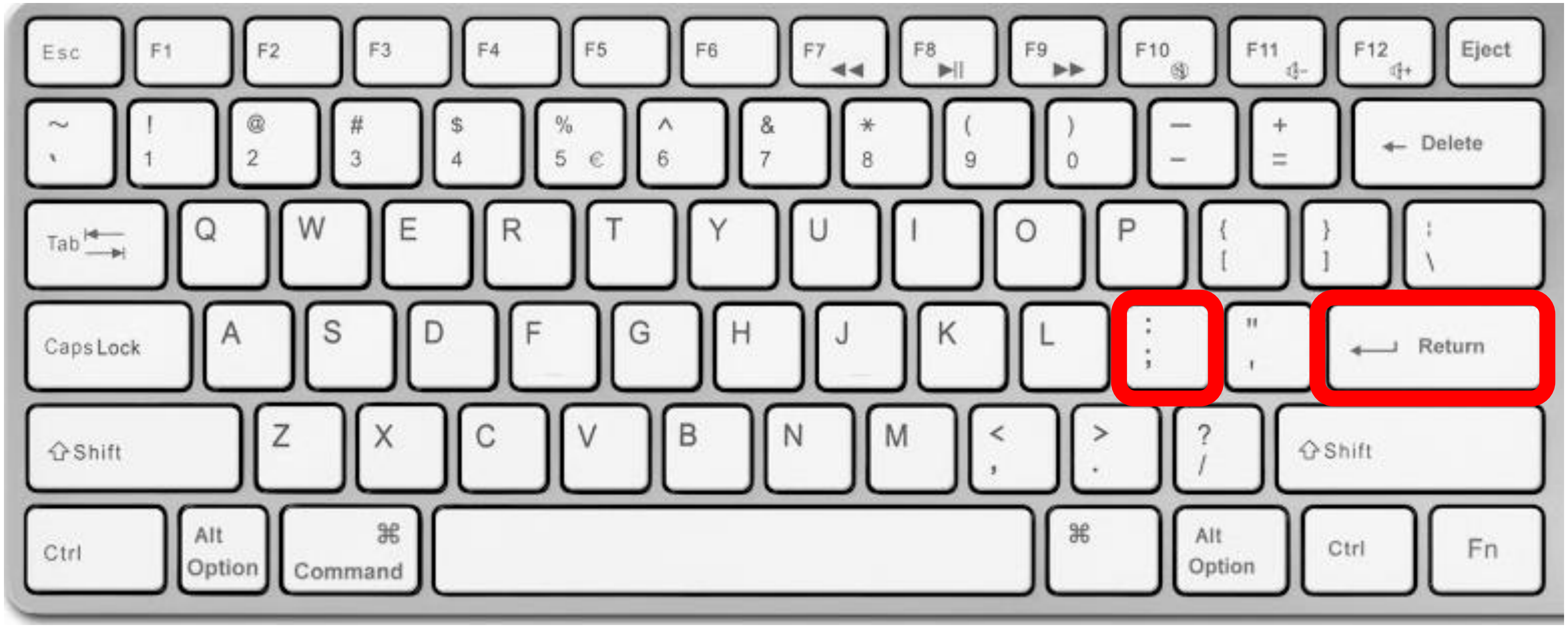
CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)

creating a vector from a known list of numbers:

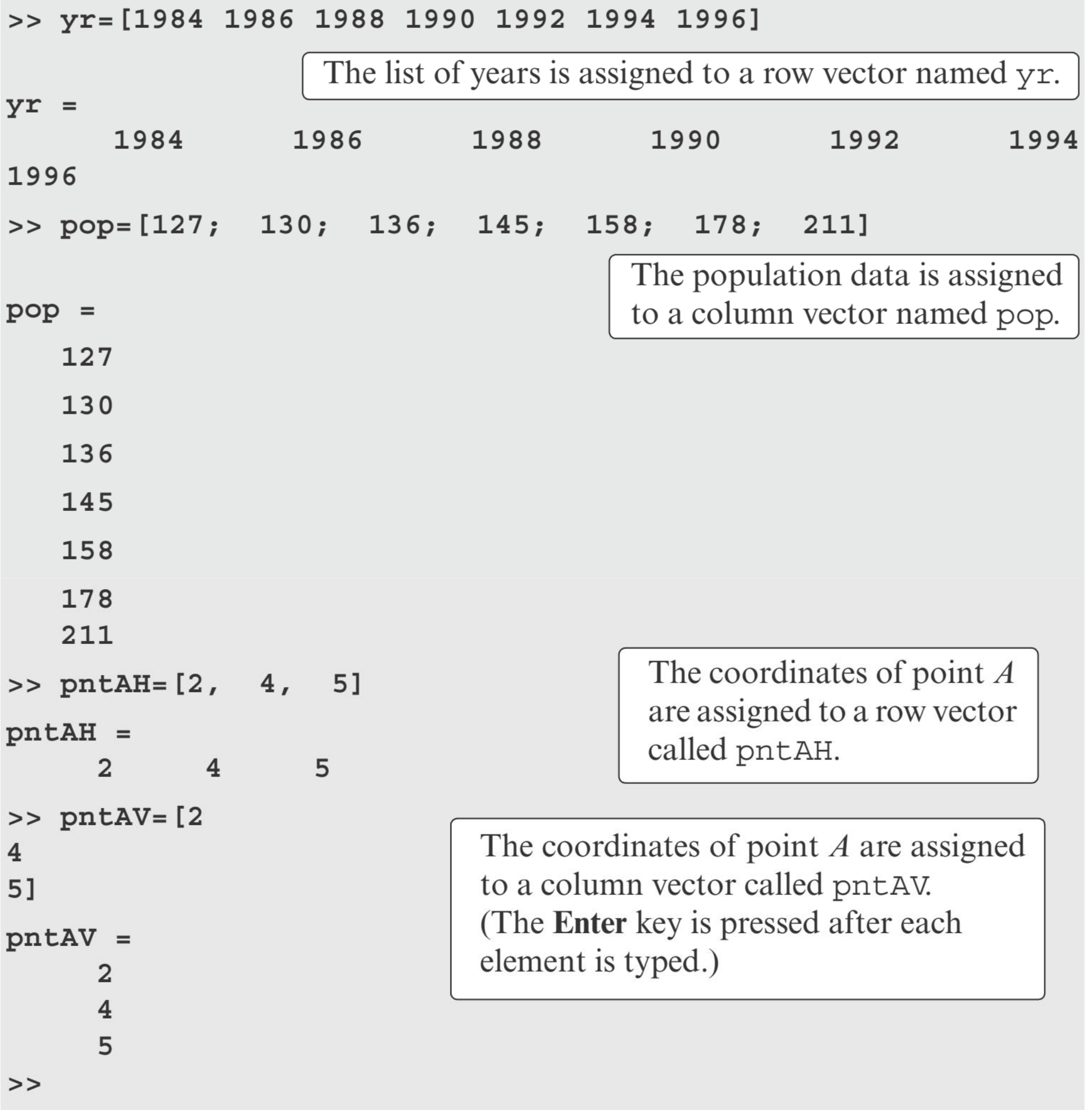
square brackets [ ]

variable_name = [type vector elements]

COLUMN VECTOR



CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)



```
>> yr=[1984 1986 1988 1990 1992 1994 1996]

yr =
      1984       1986       1988       1990       1992       1994
1996
>> pop=[127;  130;  136;  145;  158;  178;  211]

pop =
   127

   130

   136

   145

   158

   178
   211
>> pntAH=[2,  4,  5]
pntAH =
     2     4     5
>> pntAV=[2
4
5]
pntAV =
     2
     4
     5
>>
```

The list of years is assigned to a row vector named yr.

The population data is assigned to a column vector named pop.

The coordinates of point *A* are assigned to a row vector called pntAH.

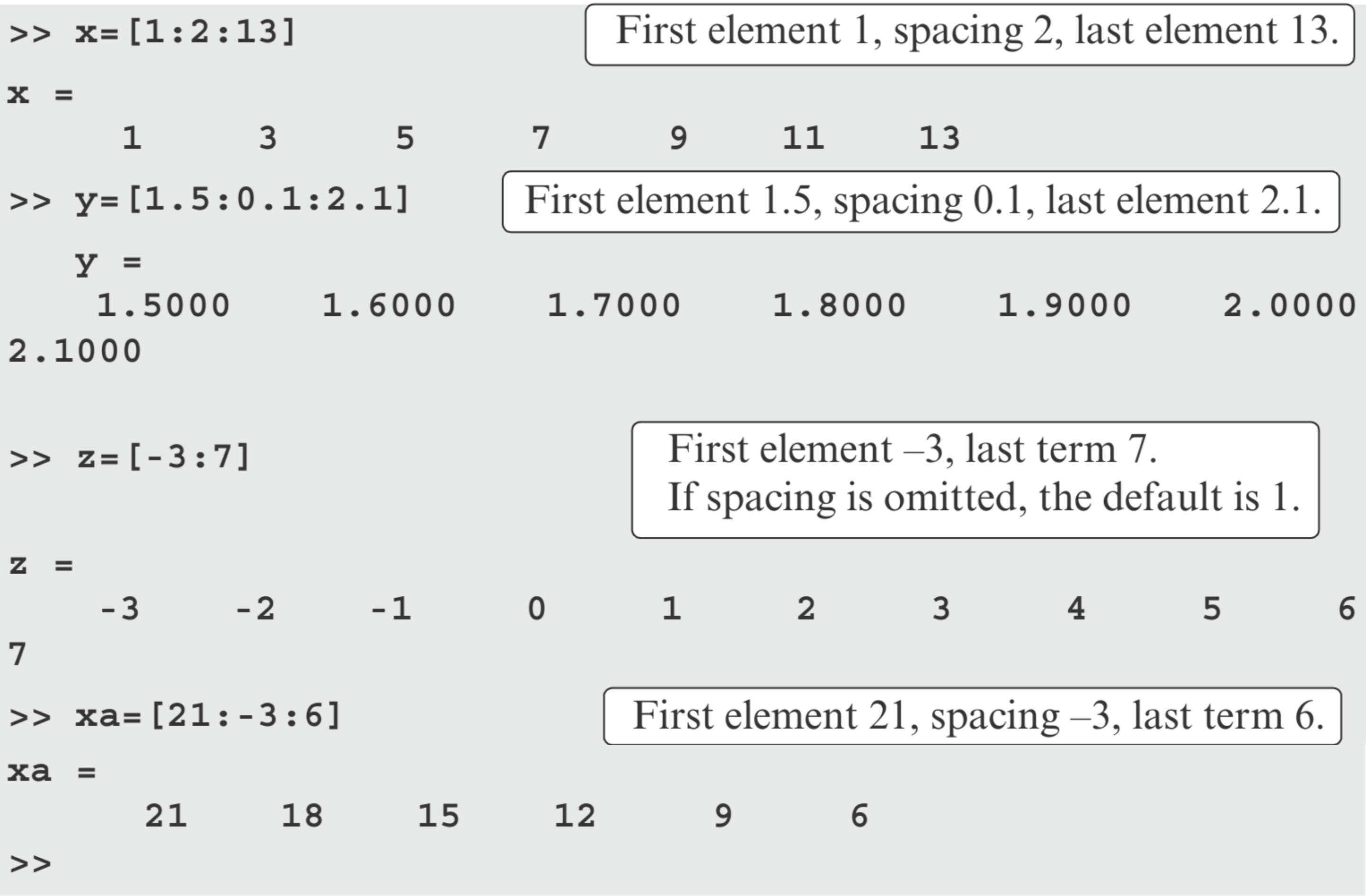
The coordinates of point *A* are assigned to a column vector called pntAV. (The **Enter** key is pressed after each element is typed.)

CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)

creating a vector with constant spacing by specifying the first term, the spacing, and the last term:

variable_name = [m:q:n]
or
variable_name = m:q:n

CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)



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

x =
     1     3     5     7     9    11    13
>> y=[1.5:0.1:2.1]

   y =
    1.5000    1.6000    1.7000    1.8000    1.9000    2.0000
2.1000


>> z=[-3:7]


z =
    -3    -2    -1     0     1     2     3     4     5     6
7
>> xa=[21:-3:6]
xa =
      21    18    15    12     9     6
>>
```

First element 1, spacing 2, last element 13.

First element 1.5, spacing 0.1, last element 2.1.

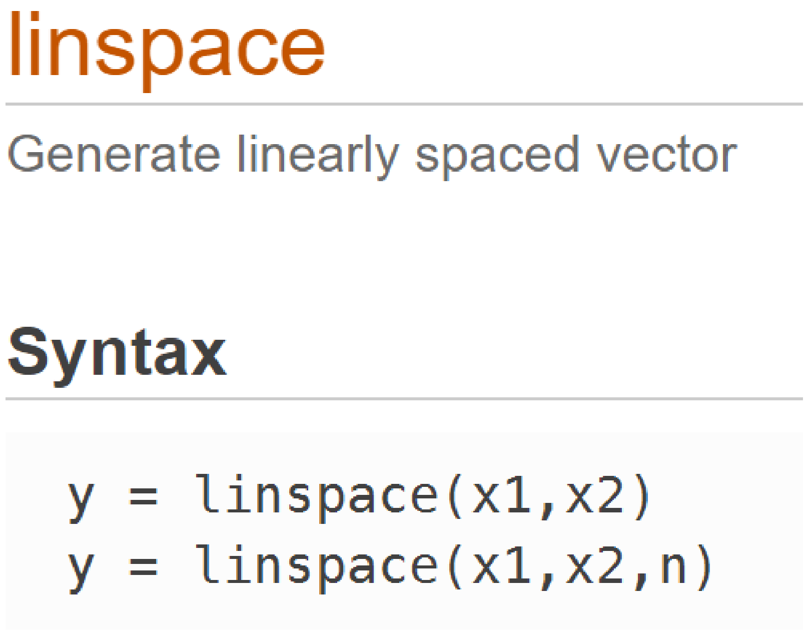
First element –3, last term 7.
If spacing is omitted, the default is 1.

First element 21, spacing –3, last term 6.

CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)

creating a vector with linear (equal) spacing by specifying the first and last terms, and the number of terms:

variable_name = linspace(xi, xf, n)



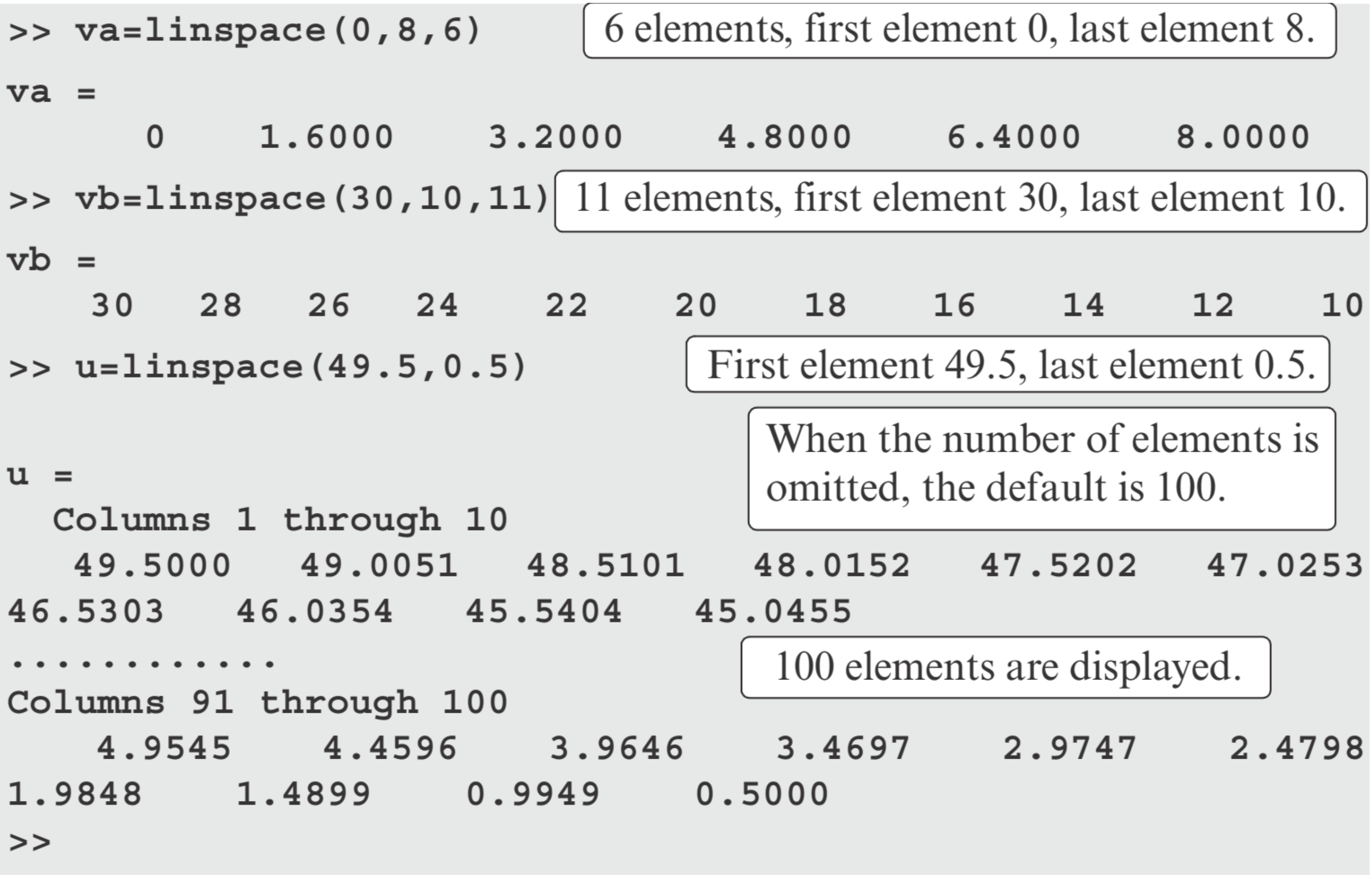
linspace

Generate linearly spaced vector

Syntax

```
y = linspace(x1,x2)
y = linspace(x1,x2,n)
```

CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)



```
>> va=linspace(0,8,6)
va =
      0    1.6000    3.2000    4.8000    6.4000    8.0000
```

6 elements, first element 0, last element 8.

```
>> vb=linspace(30,10,11)
vb =
    30    28    26    24    22    20    18    16    14    12    10
```

11 elements, first element 30, last element 10.

```
>> u=linspace(49.5,0.5)

u =
  Columns 1 through 10
   49.5000   49.0051   48.5101   48.0152   47.5202   47.0253
46.5303   46.0354   45.5404   45.0455
...........
Columns 91 through 100
    4.9545    4.4596    3.9646    3.4697    2.9747    2.4798
1.9848    1.4899    0.9949    0.5000
>>
```

First element 49.5, last element 0.5.

When the number of elements is omitted, the default is 100.

100 elements are displayed.

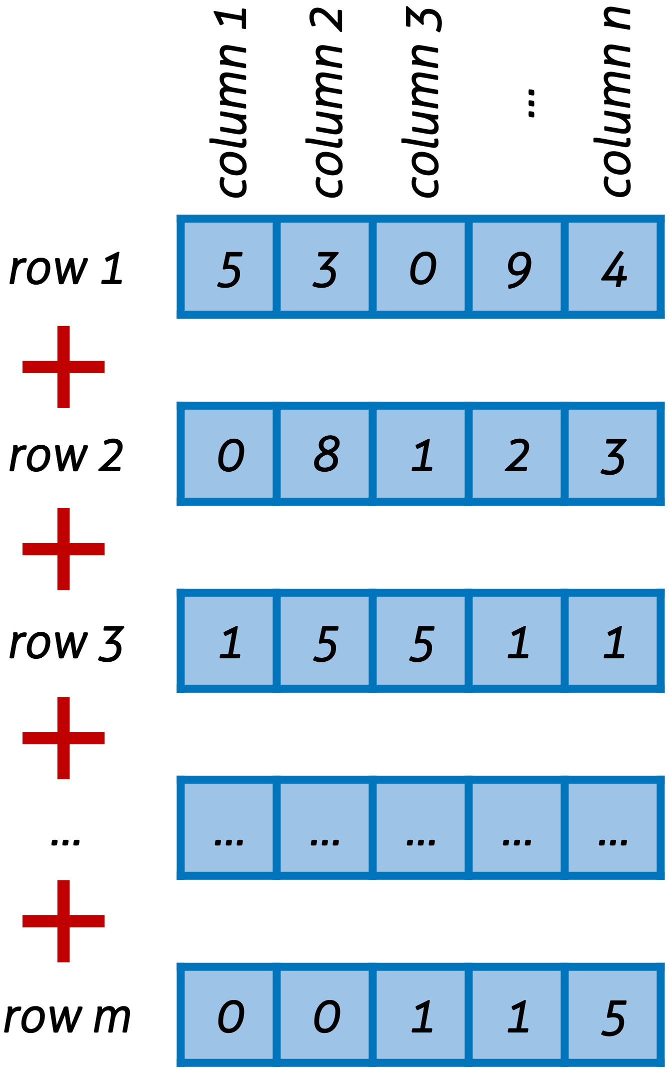
CREATING A TWO-DIMENSIONAL ARRAY (MATRIX)

$m \times n$ matrix

size of the matrix

$$
\begin{array}{c|ccccc}
 & \text{column 1} & \text{column 2} & \text{column 3} & \dots & \text{column } n \\
\hline
\text{row 1} & 5 & 3 & 0 & 9 & 4 \\
\text{row 2} & 0 & 8 & 1 & 2 & 3 \\
\text{row 3} & 1 & 5 & 5 & 1 & 1 \\
\dots & \dots & \dots & \dots & \dots & \dots \\
\text{row } m & 0 & 0 & 1 & 1 & 5
\end{array}
$$

CREATING A TWO-DIMENSIONAL ARRAY (MATRIX)

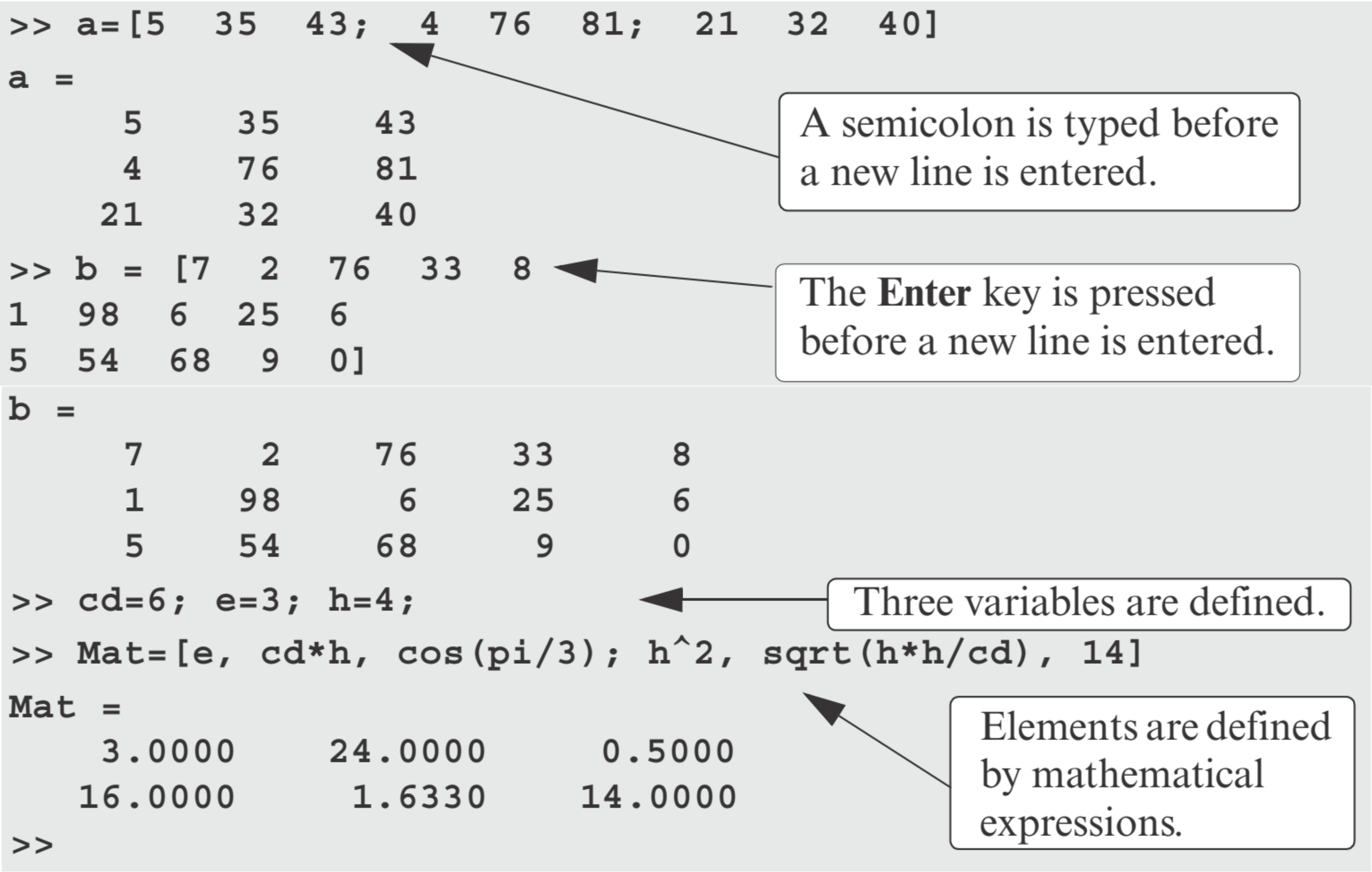


	column 1	column 2	column 3	…	column n
row 1	5	3	0	9	4
+					
row 2	0	8	1	2	3
+					
row 3	1	5	5	1	1
+					
…	…	…	…	…	…
+					
row m	0	0	1	1	5

```
variable_name = [1st row elements;
                 2nd row elements;
                 3rd row elements;
                 … ;
                 last row elements]
```

all the rows must have the same number of elements

CREATING A TWO-DIMENSIONAL ARRAY (MATRIX)



```
>> a=[5  35  43;  4  76  81;  21  32  40]
a =
     5     35     43
     4     76     81
    21     32     40
>> b = [7   2   76   33   8
1   98   6   25   6
5   54   68   9   0]
b =
     7      2     76     33      8
     1     98      6     25      6
     5     54     68      9      0
>> cd=6; e=3; h=4;
>> Mat=[e, 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
>>
```

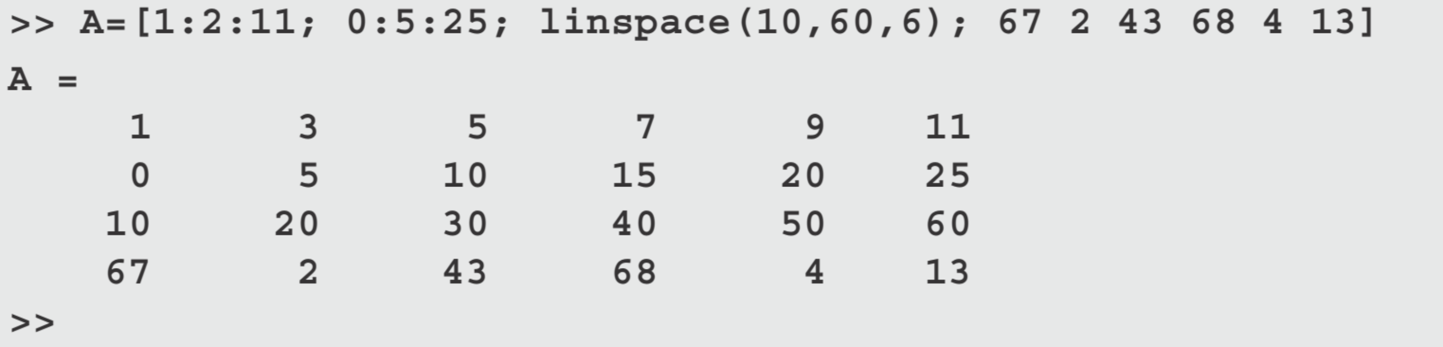
A semicolon is typed before a new line is entered.

The **Enter** key is pressed before a new line is entered.

Three variables are defined.

Elements are defined by mathematical expressions.

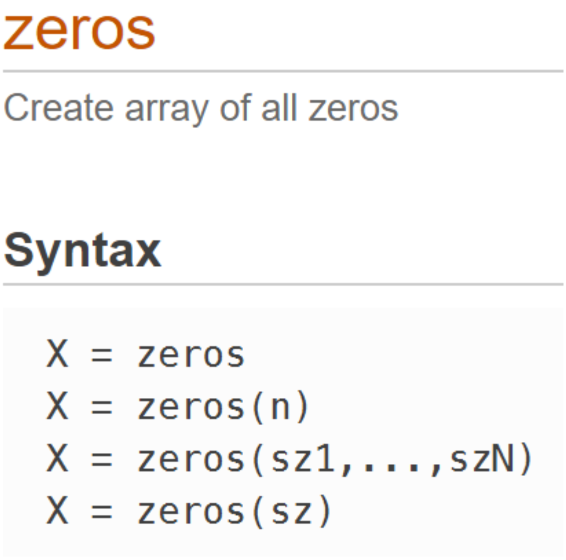
CREATING A TWO-DIMENSIONAL ARRAY (MATRIX)



```
>> 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
>>
```

CREATING A TWO-DIMENSIONAL ARRAY (MATRIX)

the zeros, ones and, eye commands:

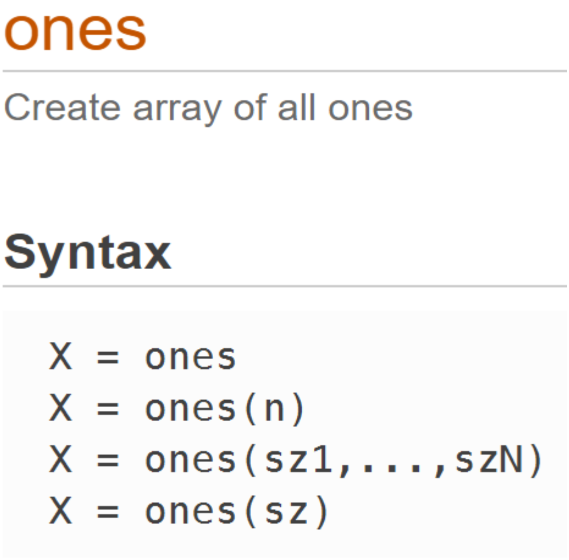


zeros

Create array of all zeros

Syntax

```
X = zeros
X = zeros(n)
X = zeros(sz1,...,szN)
X = zeros(sz)
```

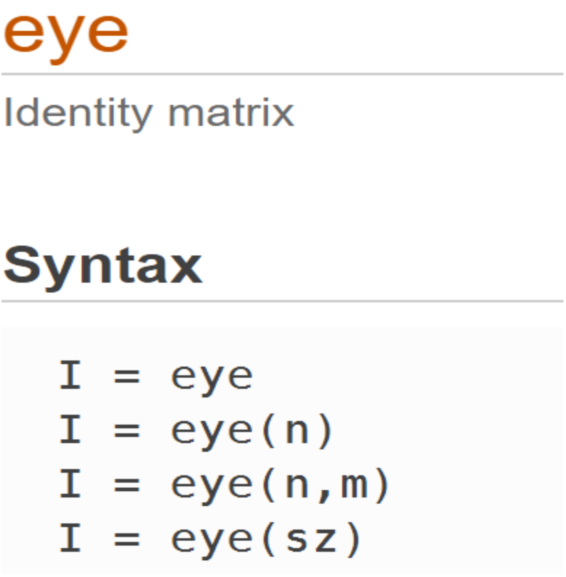


ones

Create array of all ones

Syntax

```
X = ones
X = ones(n)
X = ones(sz1,...,szN)
X = ones(sz)
```



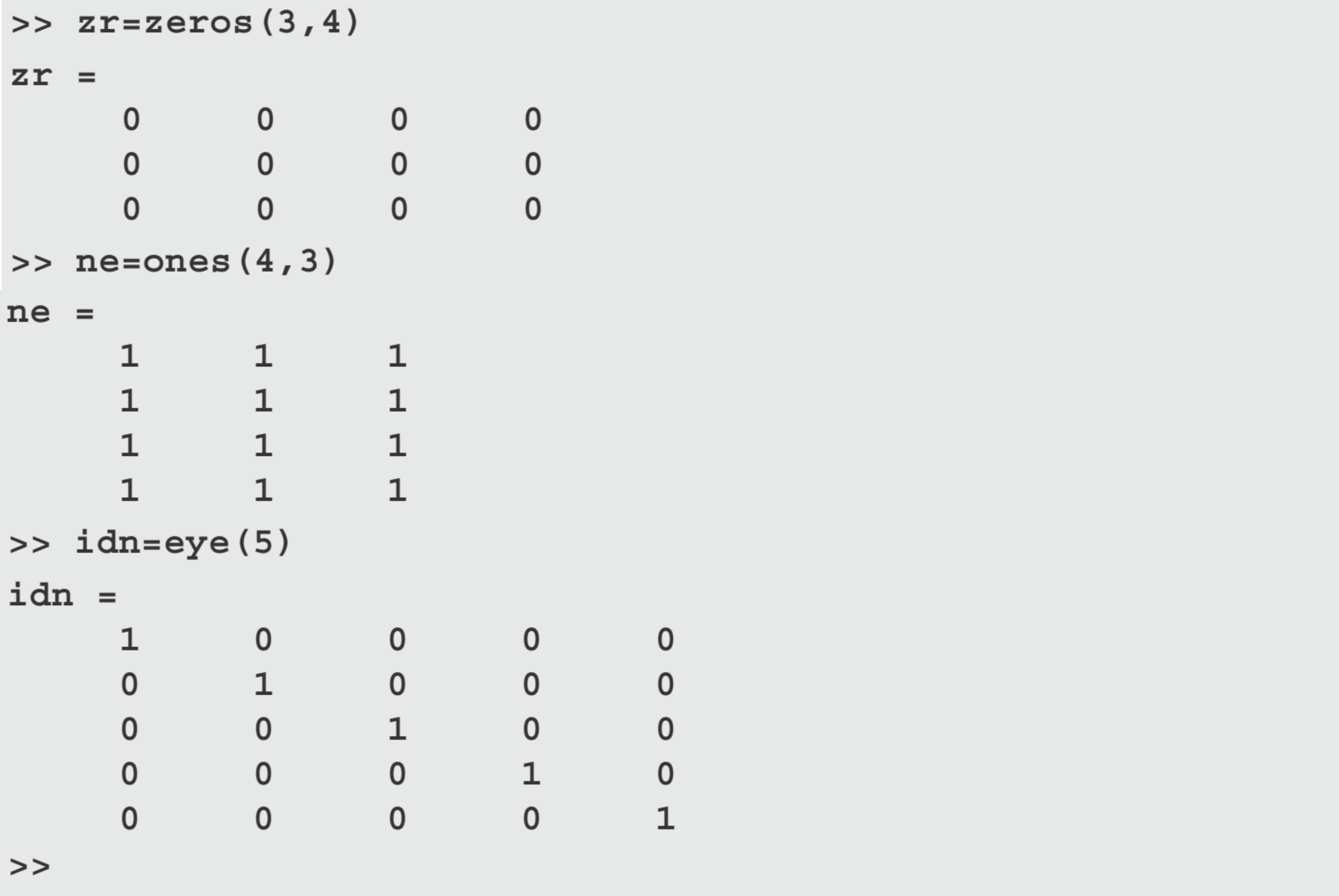
eye

Identity matrix

Syntax

```
I = eye
I = eye(n)
I = eye(n,m)
I = eye(sz)
```

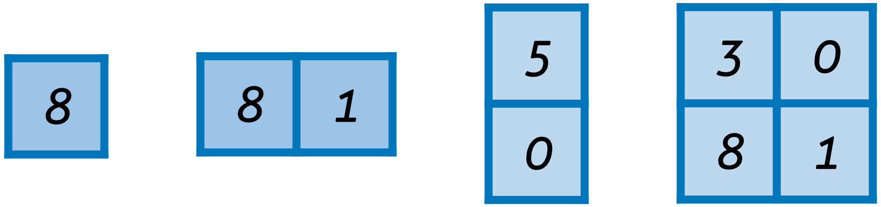
CREATING A TWO-DIMENSIONAL ARRAY (MATRIX)



```
>> zr=zeros(3,4)
zr =
     0     0     0     0
     0     0     0     0
     0     0     0     0
>> ne=ones(4,3)
ne =
     1     1     1
     1     1     1
     1     1     1
     1     1     1
>> idn=eye(5)
idn =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
>>
```

NOTES ABOUT VARIABLES IN MATLAB

- *all variables in MATLAB are arrays*



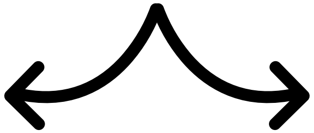
- *the variable (scalar, vector, or matrix) is defined by the input when the variable is assigned*

no need to define the size of the array

- *once a variable exists (as a scalar, vector, or matrix) it can be changed to any other size, or type, of variable*

THE TRANSPOSE OPERATOR

TRANSPOSE OPERATOR



VECTOR

switches a row (column) vector to a column (row) vector

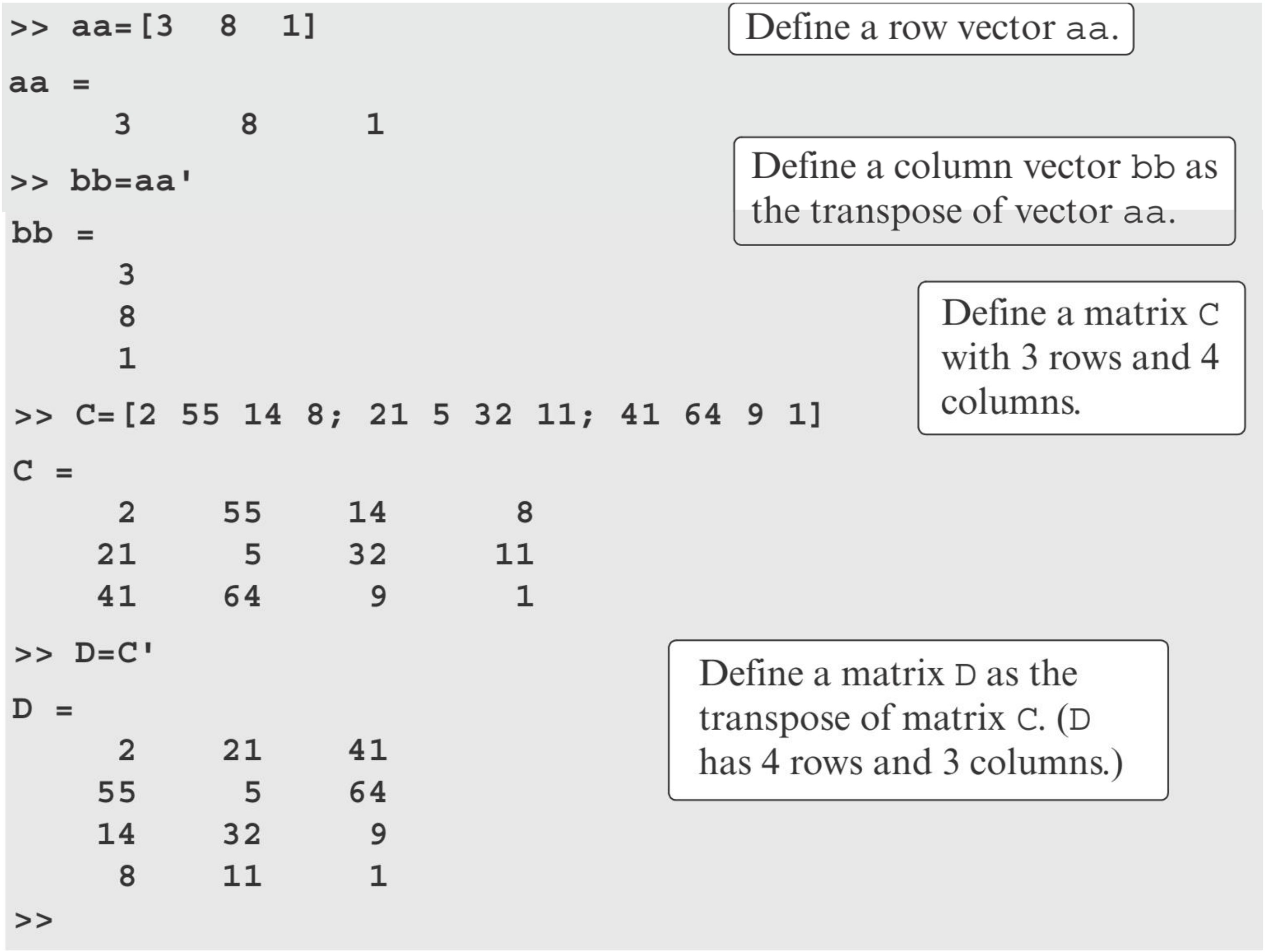
MATRIX

switches the rows (columns) to columns (rows)

single quote (')

variable_name'

THE TRANSPOSE OPERATOR



```
>> aa=[3  8  1]

aa =
     3     8     1

>> bb=aa'

bb =
     3
     8
     1

>> C=[2 55 14 8; 21 5 32 11; 41 64 9 1]

C =
     2    55    14     8
    21     5    32    11
    41    64     9     1

>> D=C'

D =
     2    21    41
    55     5    64
    14    32     9
     8    11     1

>>
```

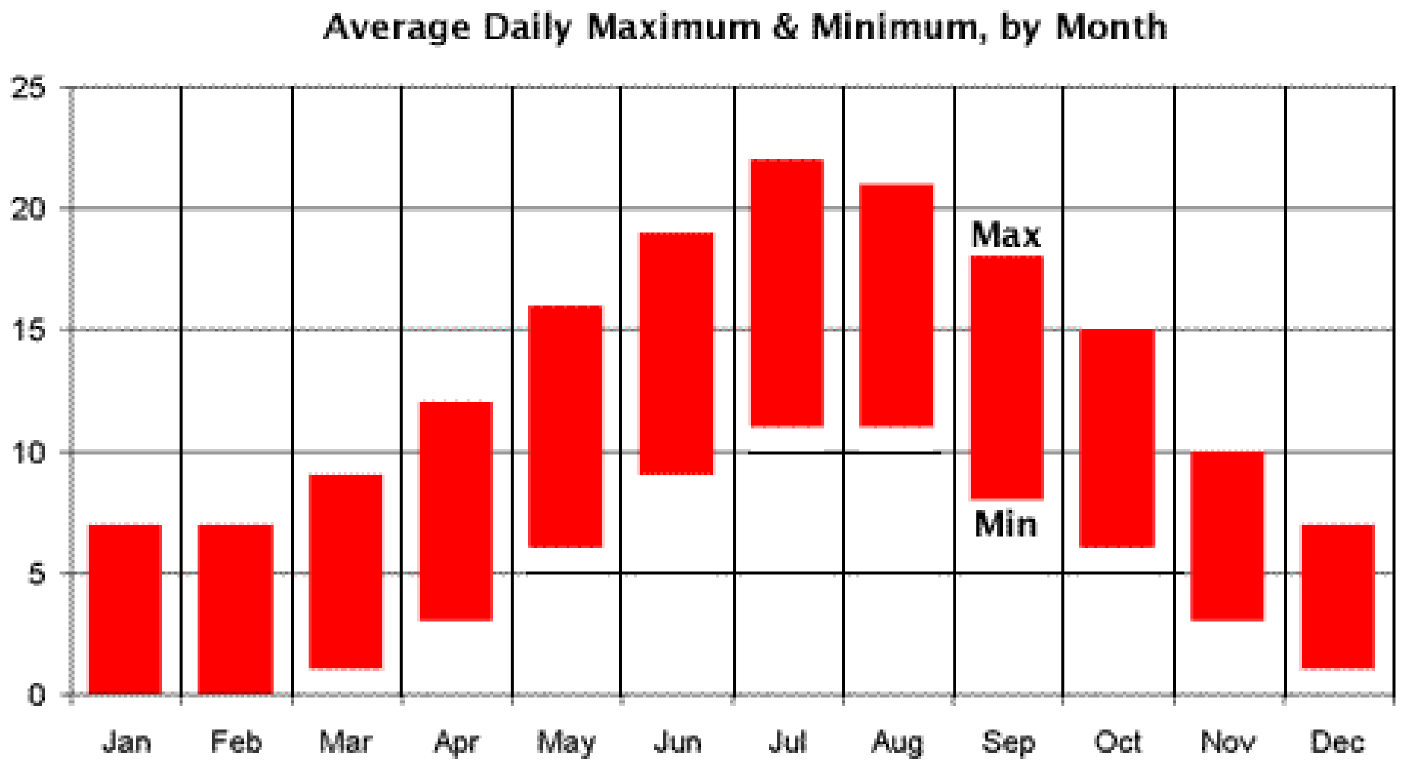
Define a row vector aa.

Define a column vector bb as the transpose of vector aa.

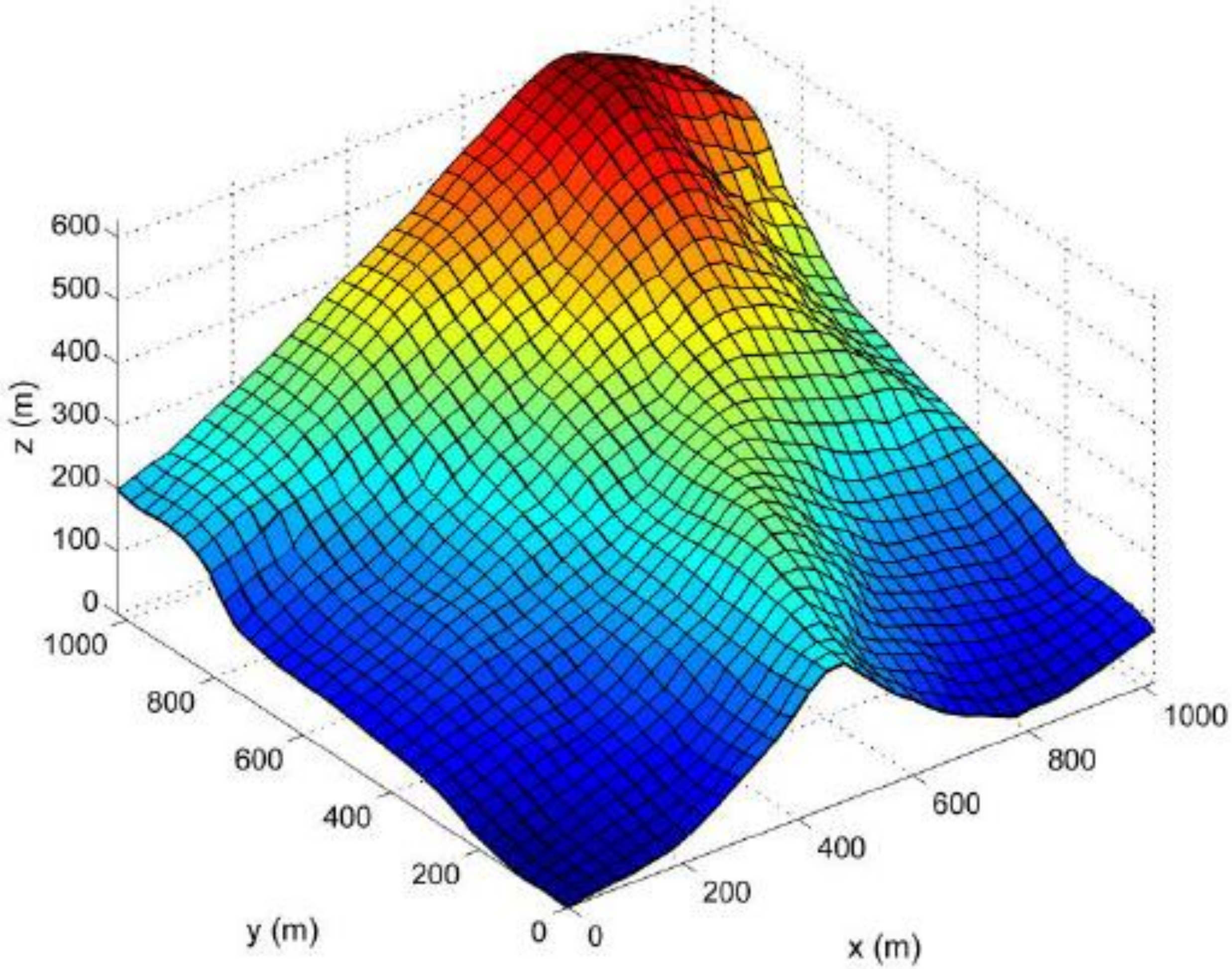
Define a matrix C with 3 rows and 4 columns.

Define a matrix D as the transpose of matrix C. (D has 4 rows and 3 columns.)

addressed individually or in subgroups



addressed individually or in subgroups



ARRAY ADDRESSING

VECTOR

variable_name = [type vector elements]

variable_name(k)

scalar

refers to the element in position k

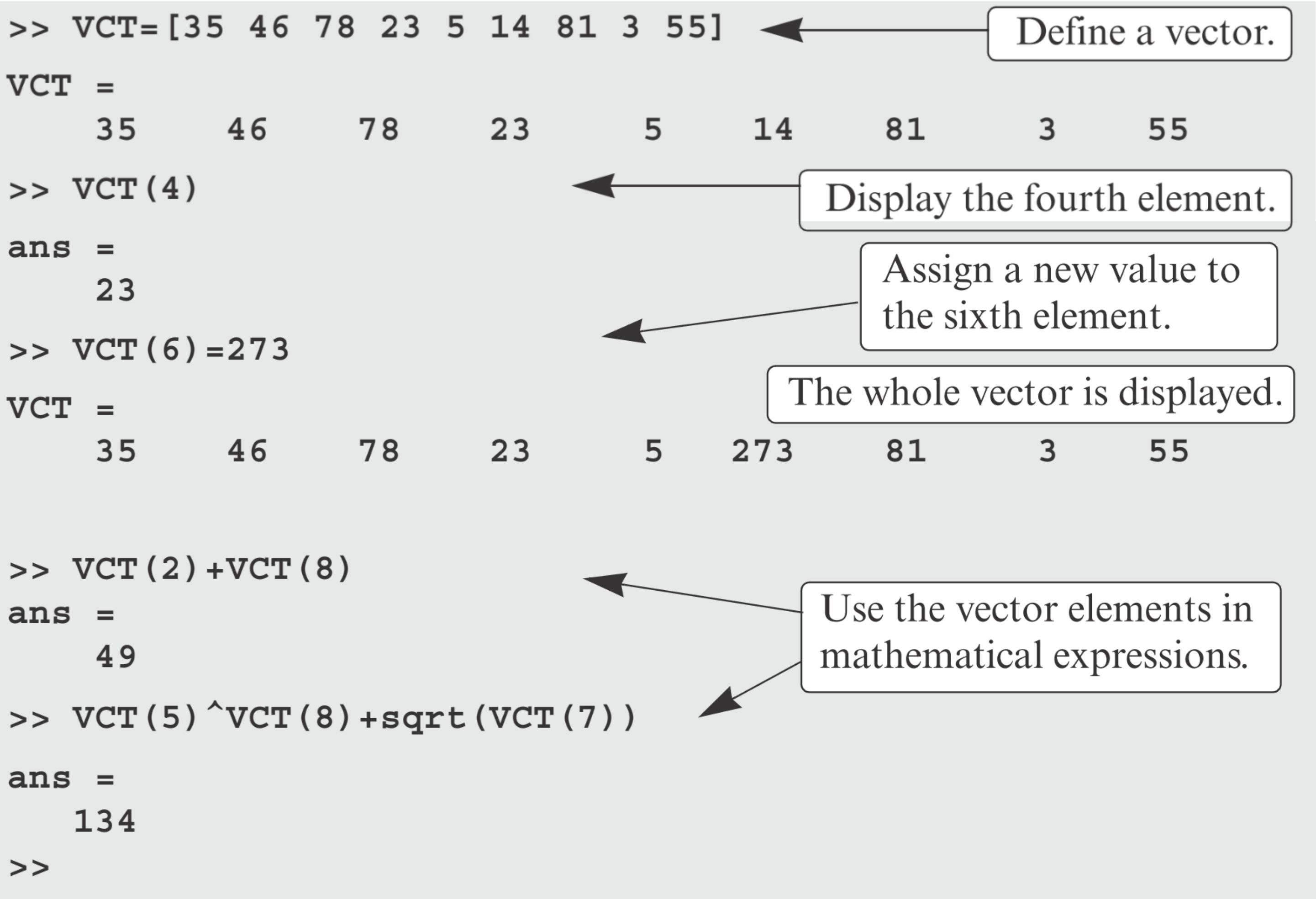
position	vector
1	5
2	0
3	1
…	…
m	4

position	1	2	3	…	n
vector	5	0	1	…	4

variable_name(k) = value

ARRAY ADDRESSING

VECTOR



```
>> VCT=[35 46 78 23 5 14 81 3 55]
VCT =
     35     46     78     23      5     14     81      3     55
>> VCT(4)
ans =
     23
>> VCT(6)=273
VCT =
     35     46     78     23      5    273     81      3     55


>> VCT(2)+VCT(8)
ans =
     49
>> VCT(5)^VCT(8)+sqrt(VCT(7))
ans =
    134
>>
```

Define a vector.

Display the fourth element.

Assign a new value to the sixth element.

The whole vector is displayed.

Use the vector elements in mathematical expressions.

ARRAY ADDRESSING

matrix

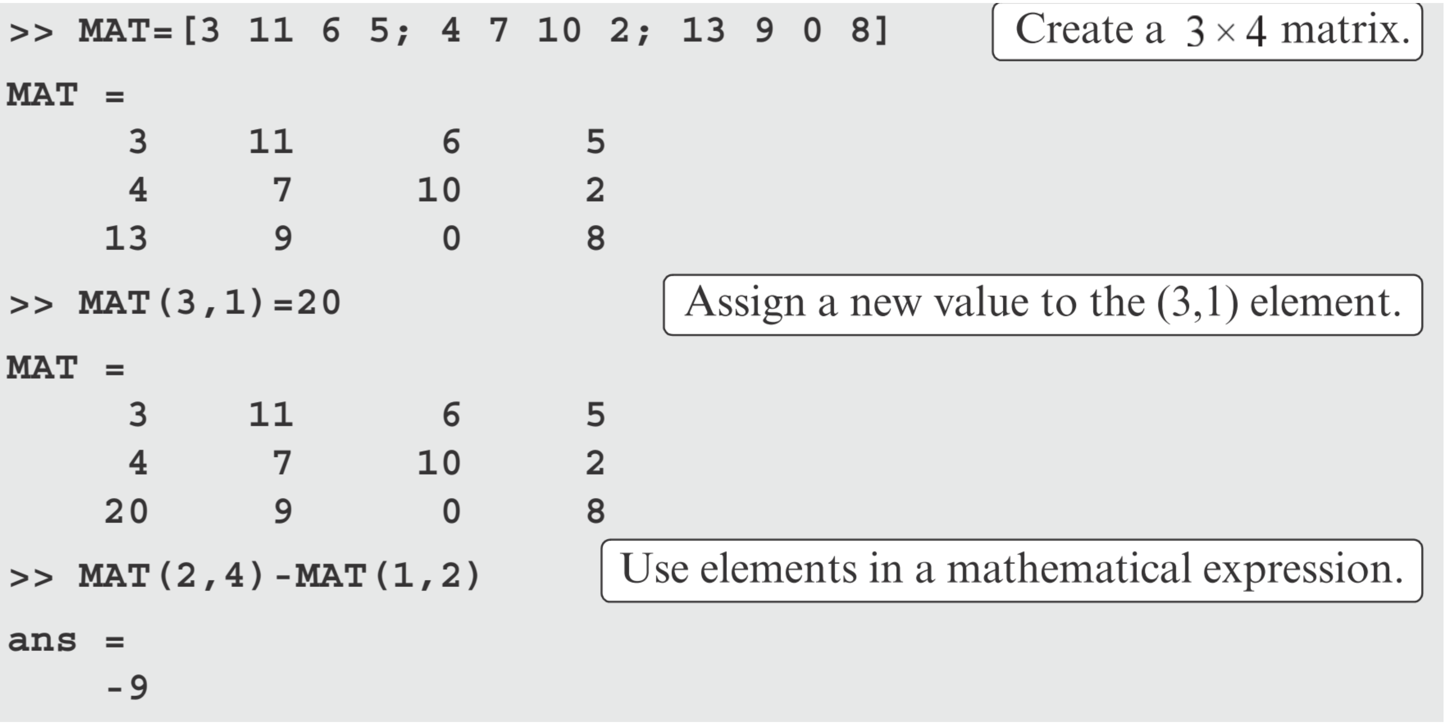
	1	2	3	…	n
1	5	3	0	9	4
2	0	8	1	2	3
3	1	5	5	1	1
…	…	…	…	…	…
m	0	0	1	1	5

```
variable_name = [1st row elements;
                 2nd row elements;
                 3rd row elements;
                 … ;
                 last row elements]

variable_name(m,n)
```

scalar

variable_name(m,n) = value

ARRAY ADDRESSING



```
>> MAT=[3 11 6 5; 4 7 10 2; 13 9 0 8]
MAT =
     3    11     6     5
     4     7    10     2
    13     9     0     8
>> MAT(3,1)=20
MAT =
     3    11     6     5
     4     7    10     2
    20     9     0     8
>> MAT(2,4)-MAT(1,2)
ans =
    -9
```

Create a 3×4 matrix.

Assign a new value to the (3,1) element.

Use elements in a mathematical expression.

USING A COLON : IN ADDRESSING ARRAYS

VECTOR

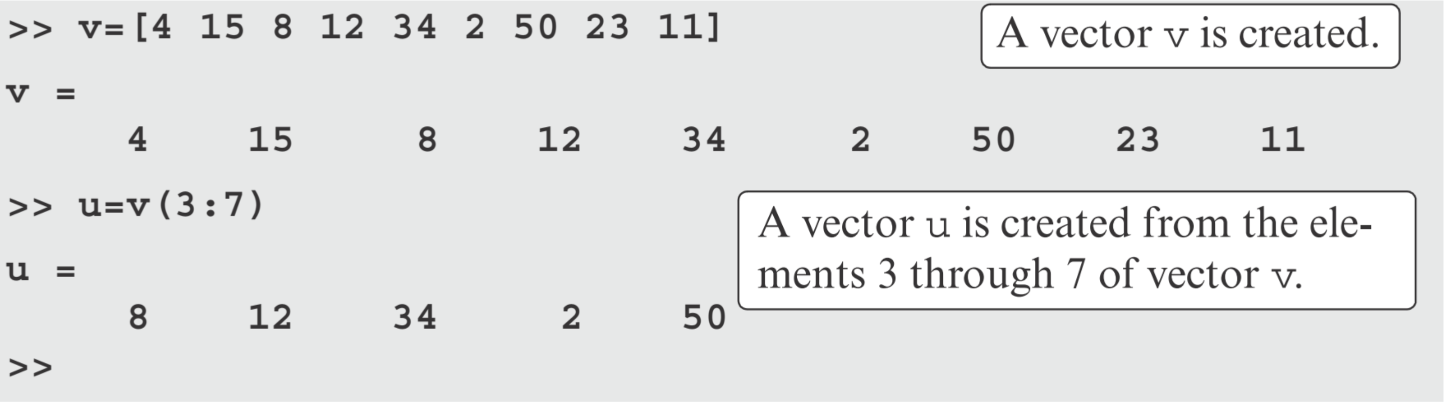
variable_name = [type vector elements]
variable_name(:)

variable_name = [type vector elements]
variable_name(m:n)

position	1	2	3	4	5	6	7	8	9
vector	5	0	1	1	4	2	8	6	8

USING A COLON : IN ADDRESSING ARRAYS

VECTOR



```
>> v=[4 15 8 12 34 2 50 23 11]
v =
      4     15      8     12     34      2     50     23     11
>> u=v(3:7)
u =
      8     12     34      2     50
>>
```

A vector v is created.

A vector u is created from the elements 3 through 7 of vector v.

USING A COLON : IN ADDRESSING ARRAYS

matrix

```
variable_name = [1st row elements;
                 2nd row elements;
                 3rd row elements;
                 ... ;
                 last row elements]
```

```
variable_name(:, n)
```

$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 5 & 3 & 0 & 9 & 4 \\
2 & 0 & 8 & 1 & 2 & 3 \\
3 & 1 & 5 & 5 & 1 & 1 \\
4 & 9 & 6 & 2 & 0 & 1 \\
5 & 0 & 0 & 1 & 1 & 5
\end{array}$$

```
variable_name(m, :)
```

$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 5 & 3 & 0 & 9 & 4 \\
2 & 0 & 8 & 1 & 2 & 3 \\
3 & 1 & 5 & 5 & 1 & 1 \\
4 & 9 & 6 & 2 & 0 & 1 \\
5 & 0 & 0 & 1 & 1 & 5
\end{array}$$

USING A COLON : IN ADDRESSING ARRAYS

matrix

```
variable_name = [1st row elements;
                 2nd row elements;
                 3rd row elements;
                 ... ;
                 last row elements]
```

`variable_name(:, m:n)`

	1	2	3	4	5
1	5	**3**	**0**	9	4
2	0	**8**	**1**	2	3
3	1	**5**	**5**	1	1
4	9	**6**	**2**	0	1
5	0	**0**	**1**	1	5

`variable_name(m:n, :)`

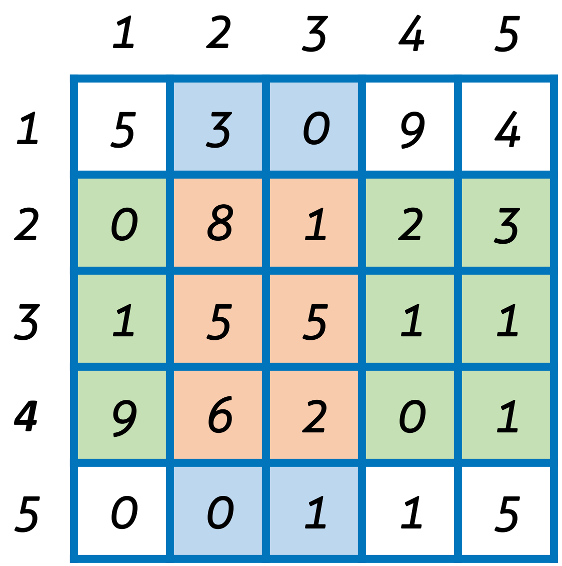
	1	2	3	4	5
1	5	3	0	9	4
2	0	8	1	2	3
3	1	5	5	1	1
4	**9**	**6**	**2**	**0**	**1**
5	**0**	**0**	**1**	**1**	**5**

USING A COLON : IN ADDRESSING ARRAYS

MATRIX

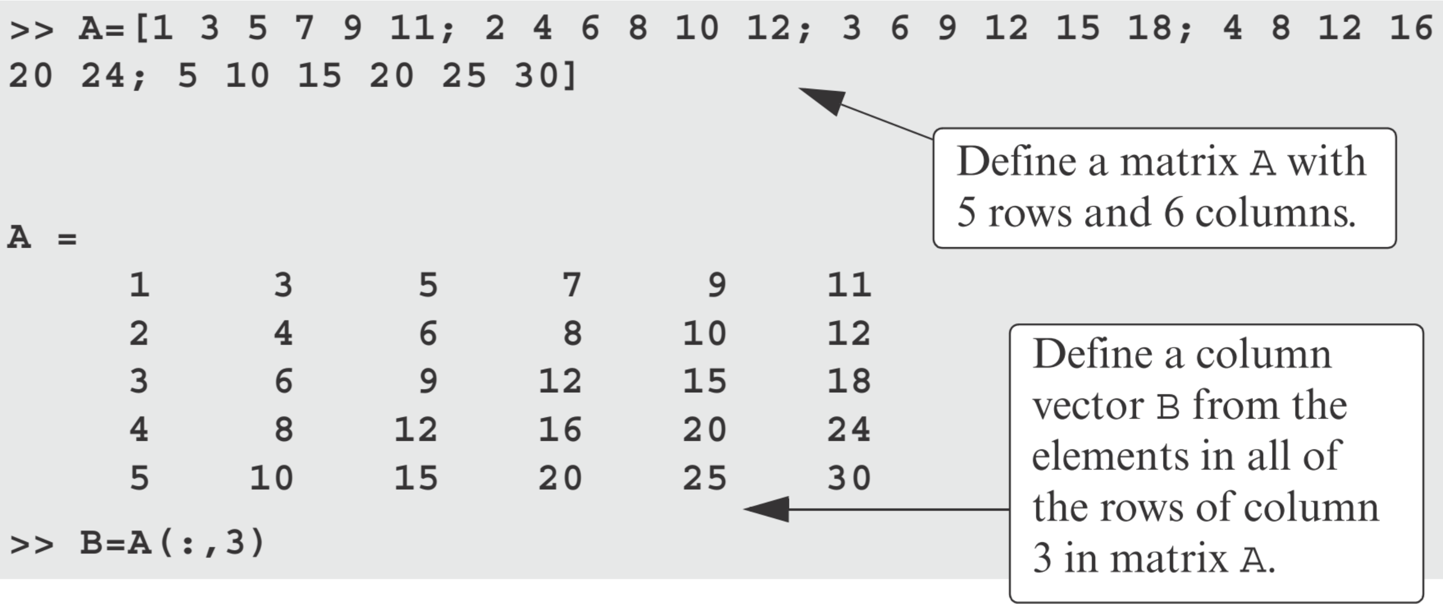
variable_name = [1st row elements;
2nd row elements;
3rd row elements;
... ;
last row elements]

variable_name(p:q, m:n)



	1	2	3	4	5
1	5	3	0	9	4
2	0	8	1	2	3
3	1	5	5	1	1
4	9	6	2	0	1
5	0	0	1	1	5

USING A COLON : IN ADDRESSING ARRAYS

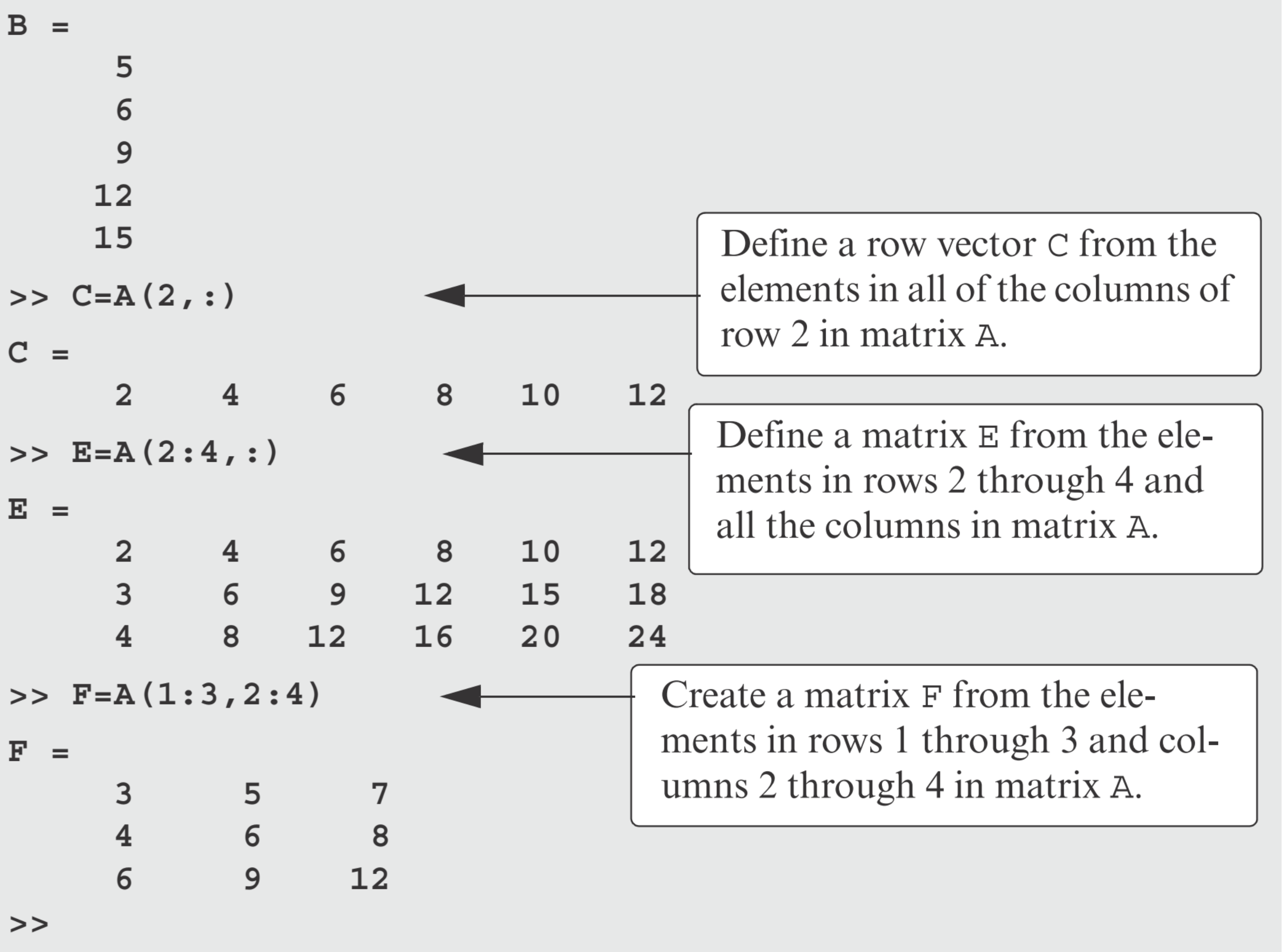


```
>> A=[1 3 5 7 9 11; 2 4 6 8 10 12; 3 6 9 12 15 18; 4 8 12 16
20 24; 5 10 15 20 25 30]



A =
     1     3     5     7     9    11
     2     4     6     8    10    12
     3     6     9    12    15    18
     4     8    12    16    20    24
     5    10    15    20    25    30

>> B=A(:,3)
```

Define a matrix A with 5 rows and 6 columns.

Define a column vector B from the elements in all of the rows of column 3 in matrix A.

USING A COLON : IN ADDRESSING ARRAYS



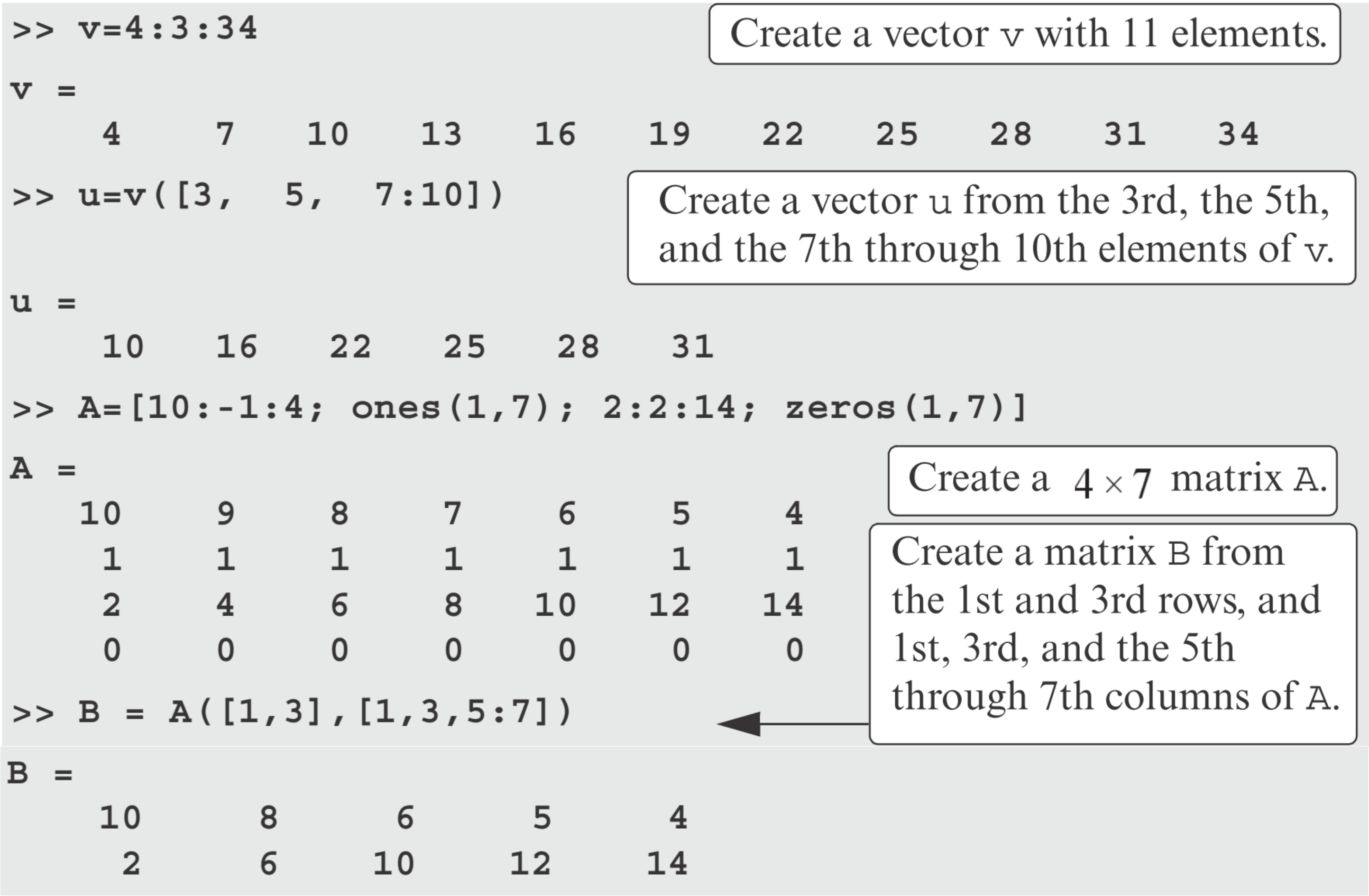
```
B =
     5
     6
     9
    12
    15

>> C=A(2,:)

C =
     2     4     6     8    10    12

>> E=A(2:4,:)

E =
     2     4     6     8    10    12
     3     6     9    12    15    18
     4     8    12    16    20    24

>> F=A(1:3,2:4)

F =
     3     5     7
     4     6     8
     6     9    12

>>
```

Define a row vector C from the elements in all of the columns of row 2 in matrix A.

Define a matrix E from the elements in rows 2 through 4 and all the columns in matrix A.

Create a matrix F from the elements in rows 1 through 3 and columns 2 through 4 in matrix A.

USING A COLON : IN ADDRESSING ARRAYS



```
>> v=4:3:34

v =
     4     7    10    13    16    19    22    25    28    31    34
>> u=v([3,  5,  7:10])


u =
    10    16    22    25    28    31
>> A=[10:-1:4; ones(1,7); 2:2:14; zeros(1,7)]

A =
    10     9     8     7     6     5     4
     1     1     1     1     1     1     1
     2     4     6     8    10    12    14
     0     0     0     0     0     0     0

>> B = A([1,3],[1,3,5:7])

B =
    10     8     6     5     4
     2     6    10    12    14
```

Create a vector v with 11 elements.

Create a vector u from the 3rd, the 5th, and the 7th through 10th elements of v.

Create a 4 × 7 matrix A.

Create a matrix B from the 1st and 3rd rows, and 1st, 3rd, and the 5th through 7th columns of A.

ADDING ELEMENTS TO EXISTING VARIABLES

vector

remember that a scalar is a vector with one element

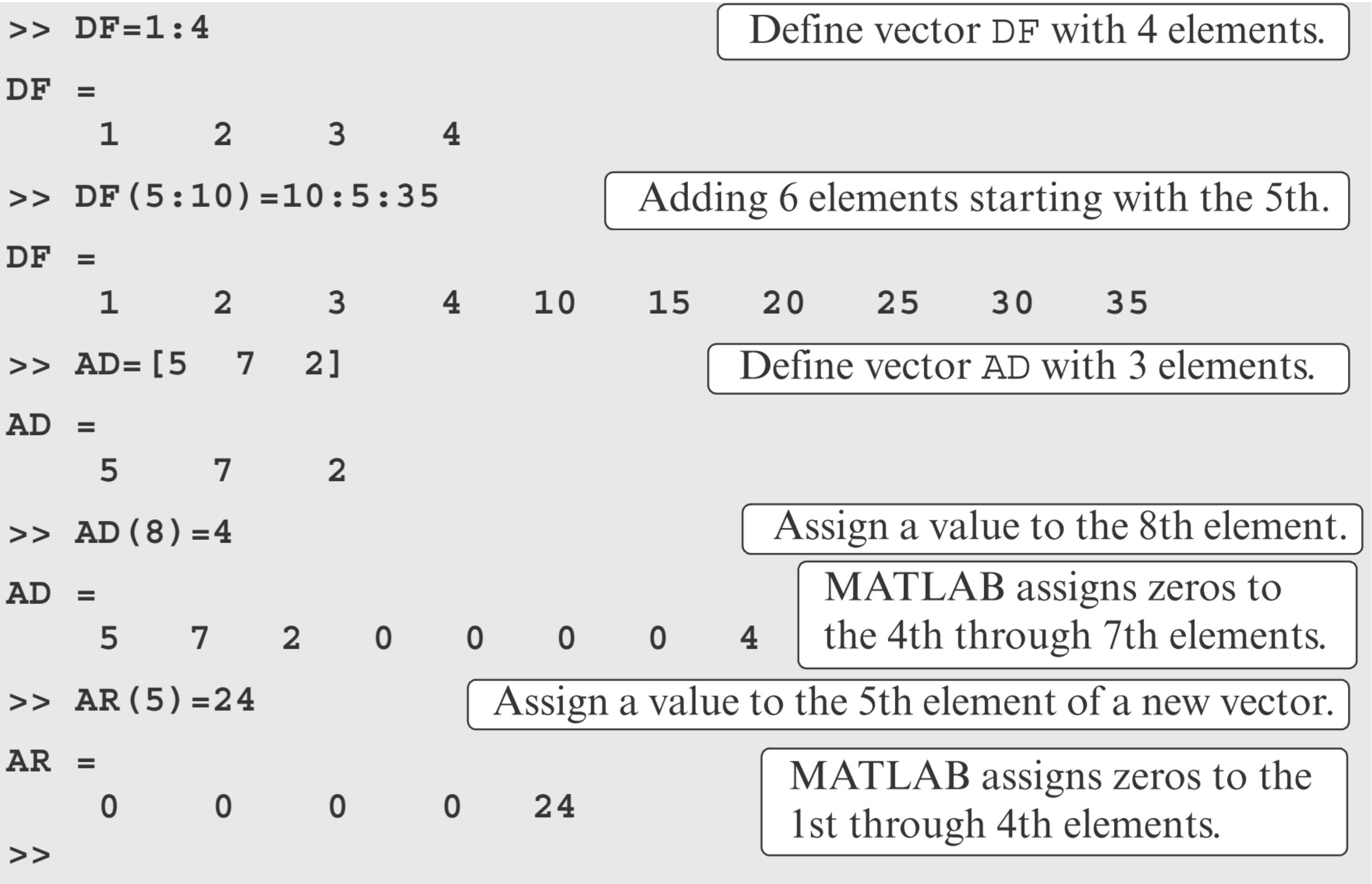
position	01	02	03	04	05	06	07	08	09	10	11	12
vector	6	7	10	13	17	20	22	21	19	14	10	7

position	01	02	03	04	05	06	07	08	09
vector	6	7	10	13	17	20	22	21	19

```
Mashhad_temp = [6,7,10,13,17,20,22,21,19]

Mashhad_temp(10:12) = [14,10,7]
```

ADDING ELEMENTS TO EXISTING VARIABLES

vector



```
>> DF=1:4

DF =
     1     2     3     4

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

DF =
     1     2     3     4    10    15    20    25    30    35

>> AD=[5  7  2]

AD =
     5     7     2

>> AD(8)=4

AD =
     5    7    2    0    0    0    0    4

>> AR(5)=24

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

Define vector DF with 4 elements.

Adding 6 elements starting with the 5th.

Define vector AD with 3 elements.

Assign a value to the 8th element.

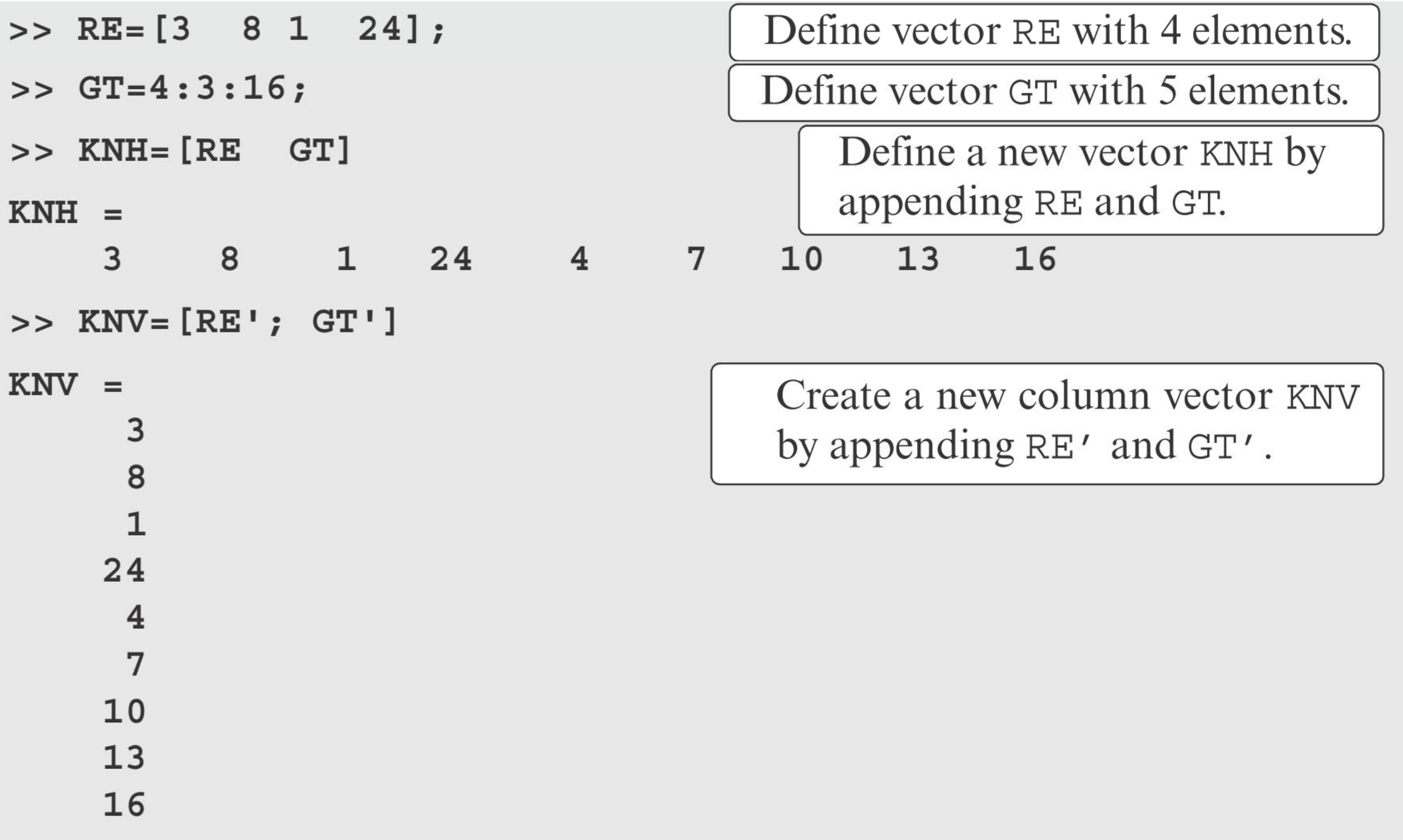
MATLAB assigns zeros to the 4th through 7th elements.

Assign a value to the 5th element of a new vector.

MATLAB assigns zeros to the 1st through 4th elements.

ADDING ELEMENTS TO EXISTING VARIABLES

vector



```
>> RE=[3   8  1   24];
>> GT=4:3:16;
>> KNH=[RE   GT]
KNH =
     3     8     1    24     4     7    10    13    16
>> KNV=[RE'; GT']
KNV =
      3
      8
      1
     24
      4
      7
     10
     13
     16
```

Define vector RE with 4 elements.

Define vector GT with 5 elements.

Define a new vector KNH by appending RE and GT.

Create a new column vector KNV by appending RE′ and GT′.

ADDING ELEMENTS TO EXISTING VARIABLES

```
variable_name = [1st row elements;
                 2nd row elements;
                 3rd row elements;
                 … ;
                 last row elements]
```

maTRIX

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 5 & 3 & 0 \\ 2 & 0 & 8 & 1 \\ 3 & 1 & 5 & 5 \end{array}$$

```
mat = [5, 3, 0; 0, 8, 1; 1, 5, 5]
```

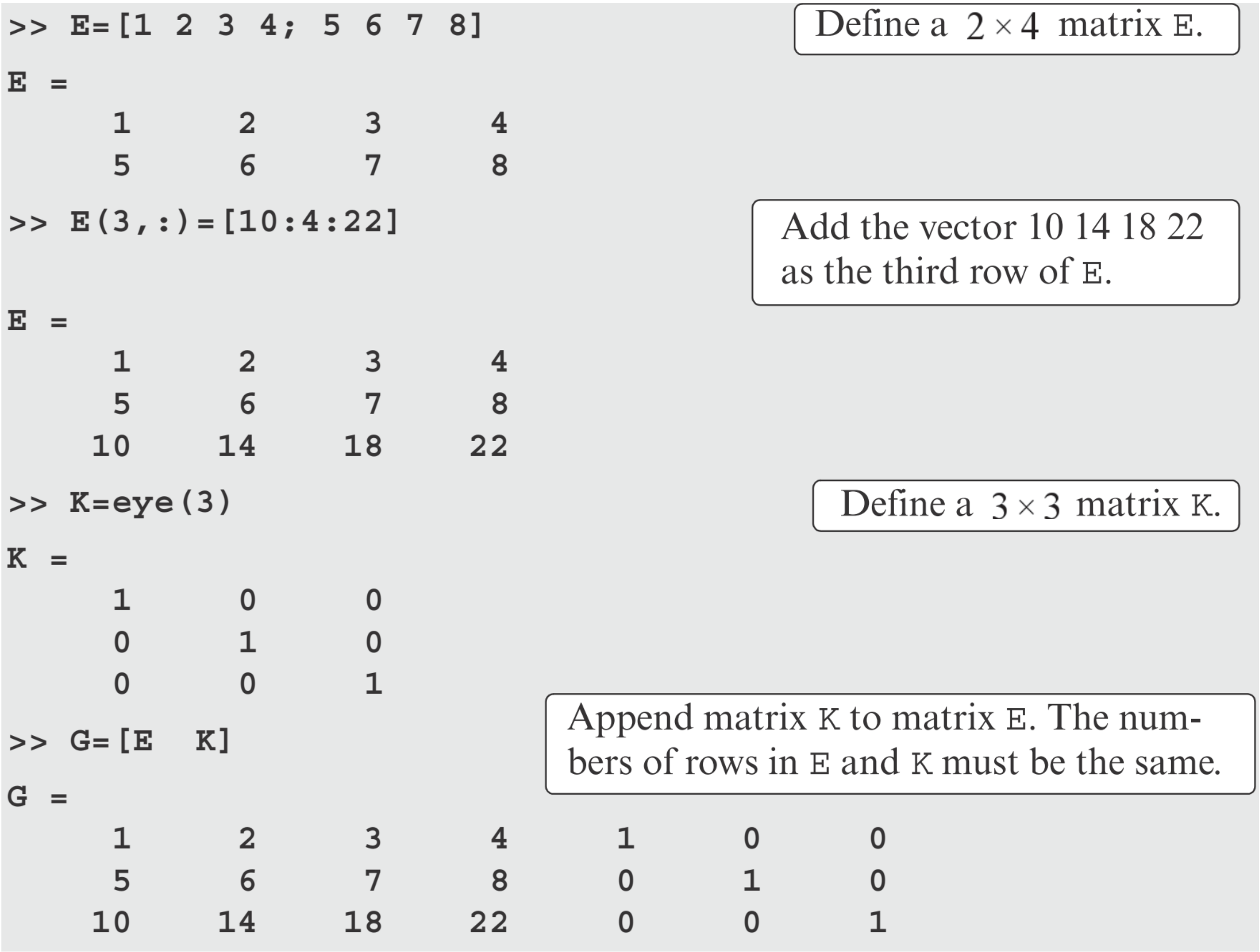
$$\begin{array}{cc} 9 & 4 \\ 2 & 3 \\ 1 & 1 \end{array}$$

```
mat(1:3, 4:5) = [9, 4; 2, 3; 1, 1]
```

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 5 & 3 & 0 & 9 & 4 \\ 2 & 0 & 8 & 1 & 2 & 3 \\ 3 & 1 & 5 & 5 & 1 & 1 \end{array}$$

ADDING ELEMENTS TO EXISTING VARIABLES

matrix



```
>> E=[1 2 3 4; 5 6 7 8]
E =
     1     2     3     4
     5     6     7     8
>> E(3,:)=[10:4:22]

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]
G =
     1     2     3     4     1     0     0
     5     6     7     8     0     1     0
    10    14    18    22     0     0     1
```

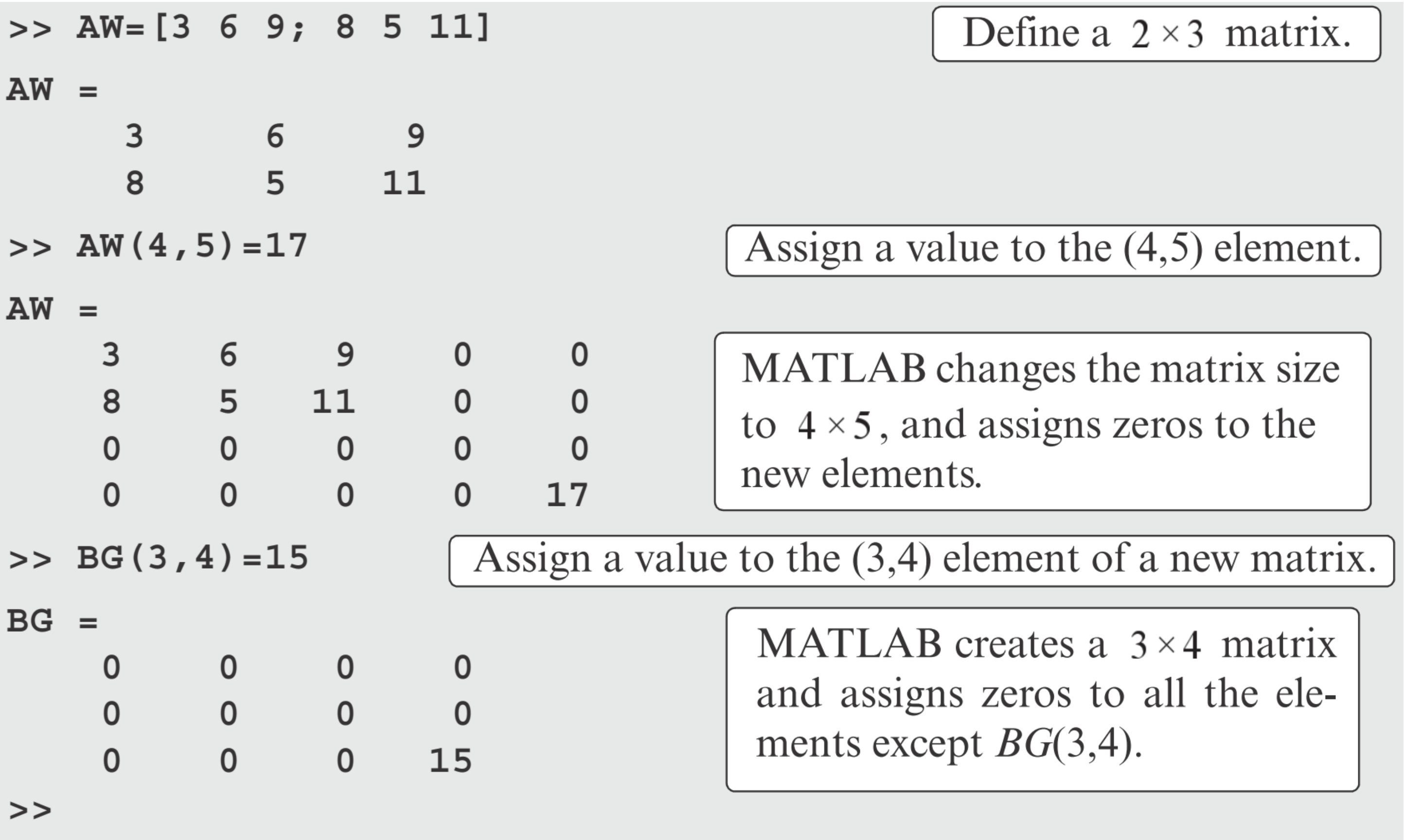
Define a 2×4 matrix E.

Add the vector 10 14 18 22 as the third row of E.

Define a 3×3 matrix K.

Append matrix K to matrix E. The numbers of rows in E and K must be the same.

ADDING ELEMENTS TO EXISTING VARIABLES



```
>> AW=[3 6 9; 8 5 11]

AW =
     3     6     9
     8     5    11

>> AW(4,5)=17

AW =
     3     6     9     0     0
     8     5    11     0     0
     0     0     0     0     0
     0     0     0     0    17

>> BG(3,4)=15

BG =
     0     0     0     0
     0     0     0     0
     0     0     0    15
>>
```

Define a 2 × 3 matrix.

Assign a value to the (4,5) element.

MATLAB changes the matrix size to 4 × 5, and assigns zeros to the new elements.

Assign a value to the (3,4) element of a new matrix.

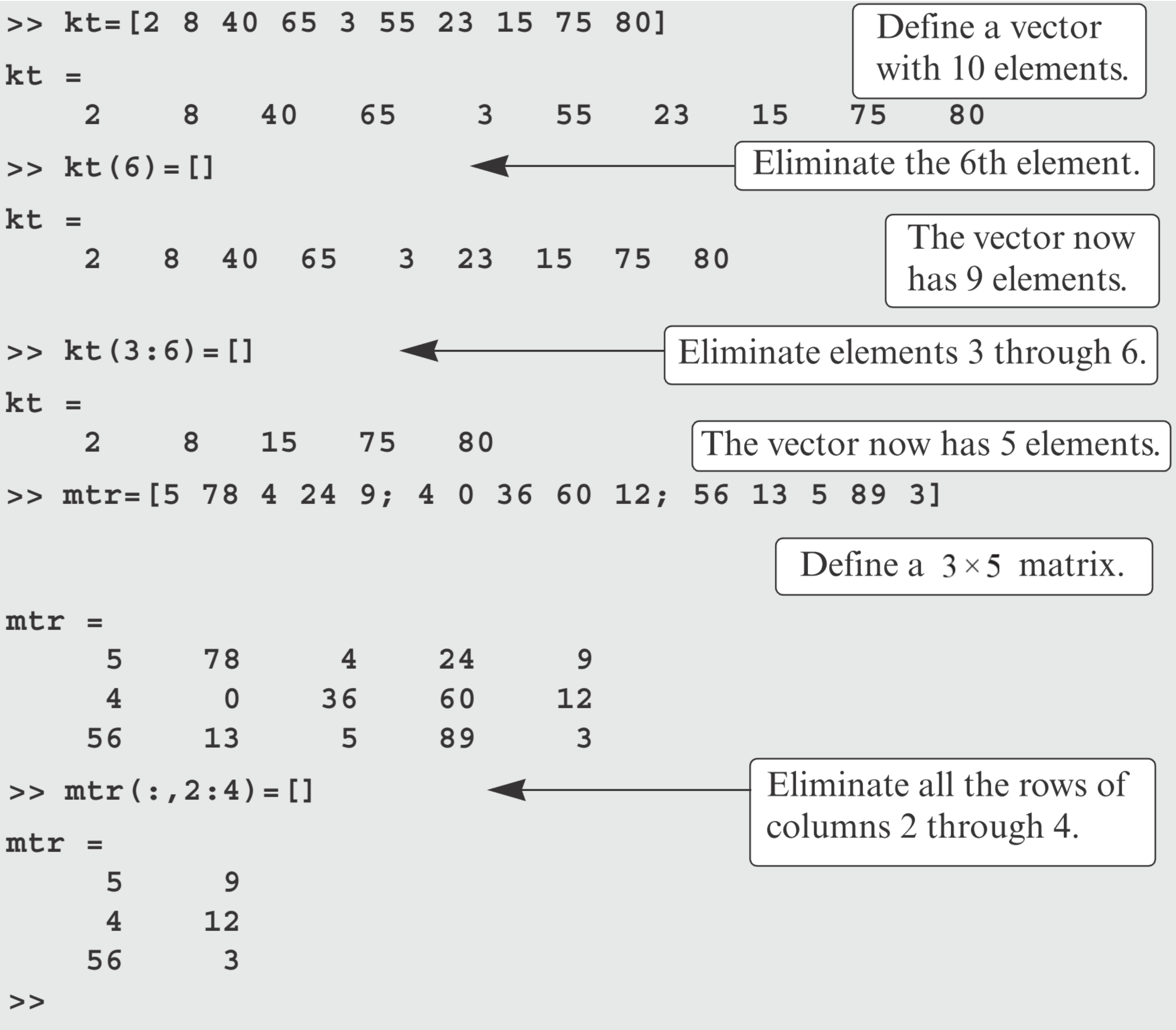
MATLAB creates a 3 × 4 matrix and assigns zeros to all the elements except *BG*(3,4).

DELETING ELEMENTS

```
variable_name = [ type vector elements ]
          variable_name(k) = []
```

```
variable_name = [1st row elements;
                 2nd row elements;
                 3rd row elements;
                 ... ;
                last row elements]

         variable_name(m,n) = []
```

DELETING ELEMENTS



```
>> kt=[2 8 40 65 3 55 23 15 75 80]
kt =
     2     8    40    65     3    55    23    15    75    80
>> kt(6)=[]
kt =
     2    8   40   65    3   23   15   75   80

>> kt(3:6)=[]
kt =
     2     8    15    75    80
>> mtr=[5 78 4 24 9; 4 0 36 60 12; 56 13 5 89 3]


mtr =
      5      78       4      24       9
      4       0      36      60      12
     56      13       5      89       3
>> mtr(:,2:4)=[]
mtr =
      5       9
      4      12
     56       3
>>
```

Define a vector with 10 elements.

Eliminate the 6th element.

The vector now has 9 elements.

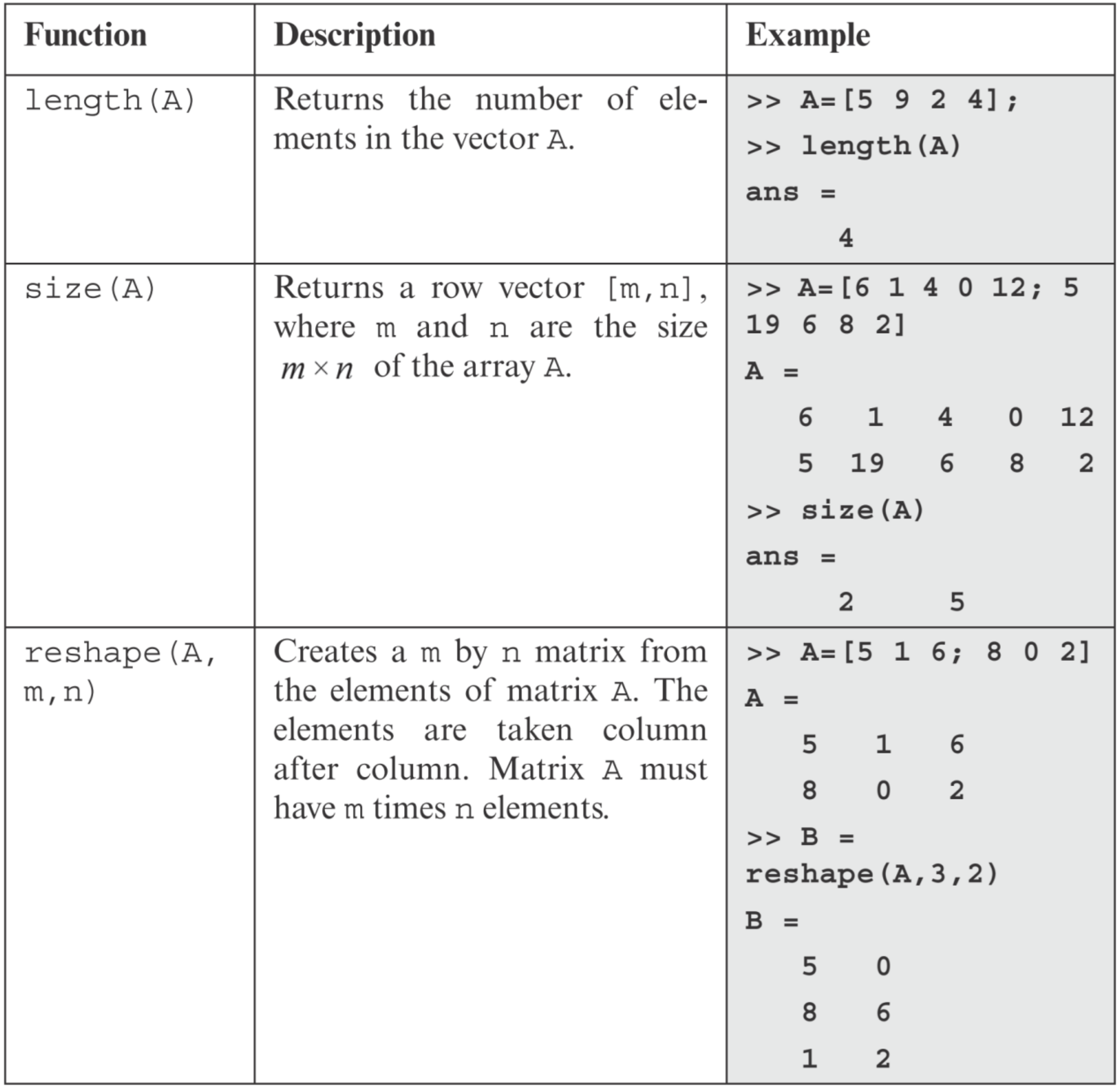
Eliminate elements 3 through 6.

The vector now has 5 elements.

Define a 3 × 5 matrix.

Eliminate all the rows of columns 2 through 4.

BUILT-IN FUNCTIONS FOR HANDLING ARRAYS



Function	Description	Example

`length(A)`	Returns the number of elements in the vector `A`.	`>> A=[5 9 2 4];`
		`>> length(A)`
		`ans =`
		`4`

`size(A)`	Returns a row vector `[m,n]`, where `m` and `n` are the size $m \times n$ of the array `A`.	`>> A=[6 1 4 0 12; 5 19 6 8 2]`
		`A =`
		`6 1 4 0 12`
		`5 19 6 8 2`
		`>> size(A)`
		`ans =`
		`2 5`

`reshape(A,m,n)`	Creates a `m` by `n` matrix from the elements of matrix `A`. The elements are taken column after column. Matrix `A` must have `m` times `n` elements.	`>> A=[5 1 6; 8 0 2]`
		`A =`
		`5 1 6`
		`8 0 2`
		`>> B = reshape(A,3,2)`
		`B =`
		`5 0`
		`8 6`
		`1 2`

BUILT-IN FUNCTIONS FOR HANDLING ARRAYS

Function	Description	Example
diag(v)	When v is a vector, creates a square matrix with the elements of v in the diagonal.	>> v=[7 4 2]; >> A=diag(v) A = 7 0 0 0 4 0 0 0 2
diag(A)	When A is a matrix, creates a vector from the diagonal elements of A.	>> A=[1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 >> vec=diag(A) vec = 1 5 9

STRINGS AND STRINGS AS VARIABLES

- *a **string** is an **array** of characters. It is created by typing the characters within **single quotes**.*

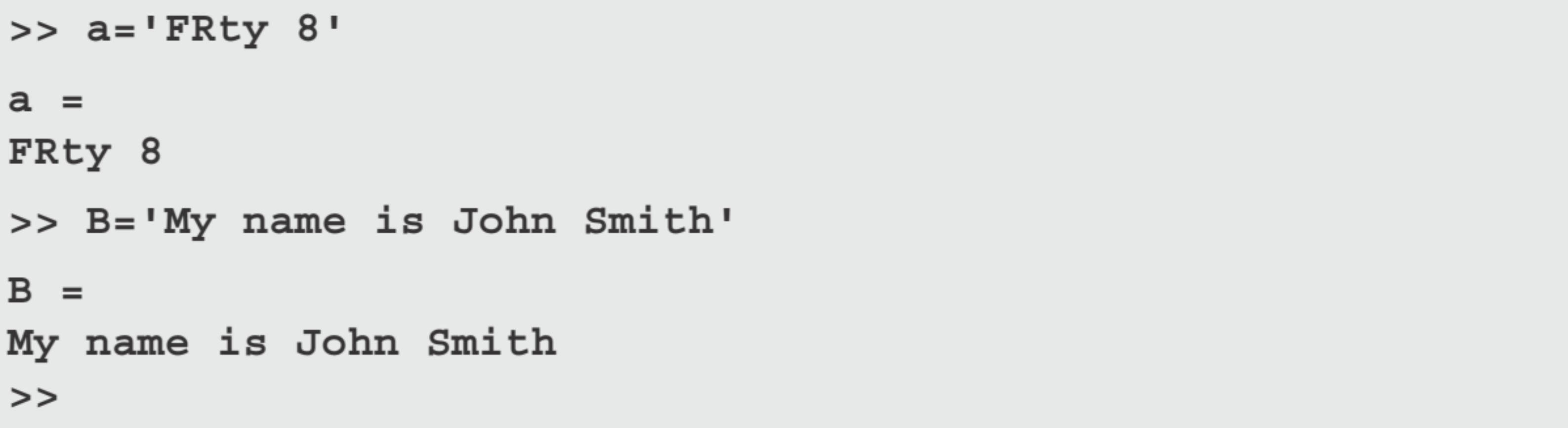
- *Strings can include letters, digits, other symbols, and spaces*

'ad ef ‘ , '3%fr2’ , '{edcba:21!’ , 'MATLAB'

- *A string that contains a single quote is created by typing two single quotes within the string.*

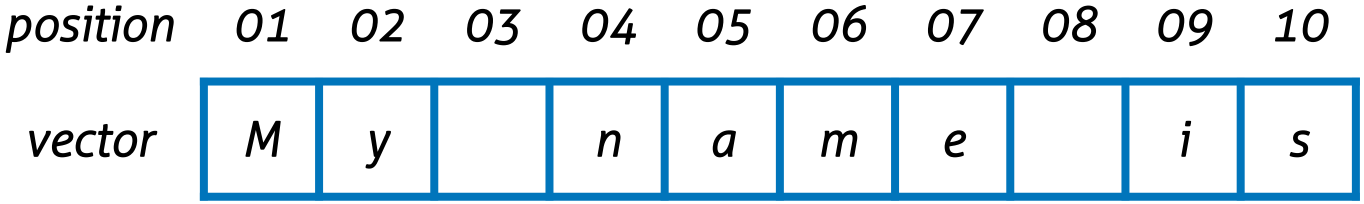
- *maroon and purple*

STRINGS AND STRINGS AS VARIABLES



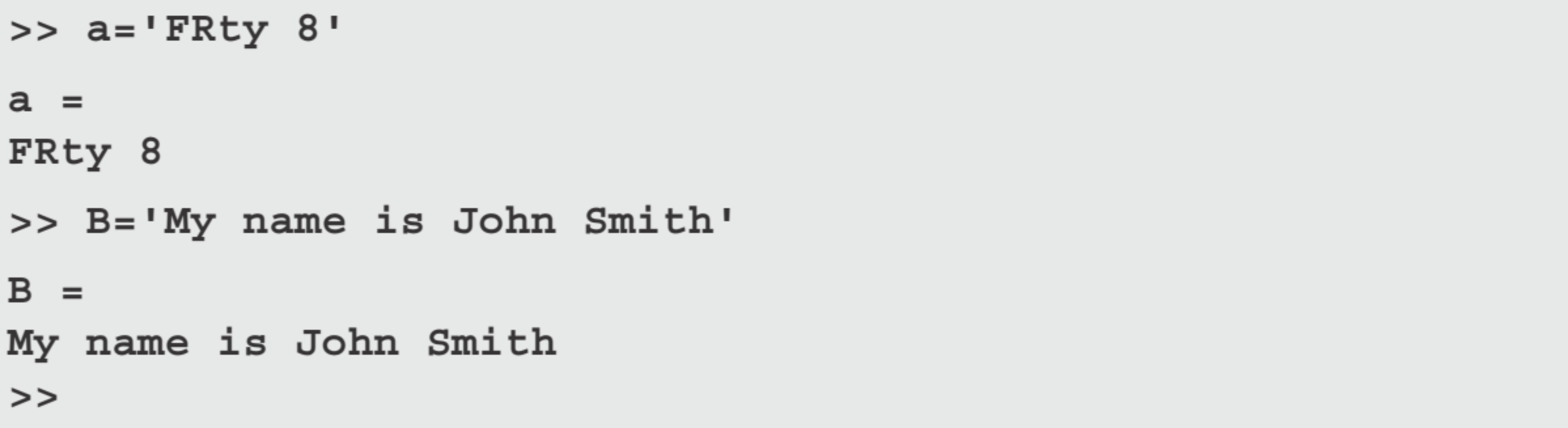
```
>> a='FRty 8'

a =
FRty 8

>> B='My name is John Smith'

B =
My name is John Smith
>>
```

row vector

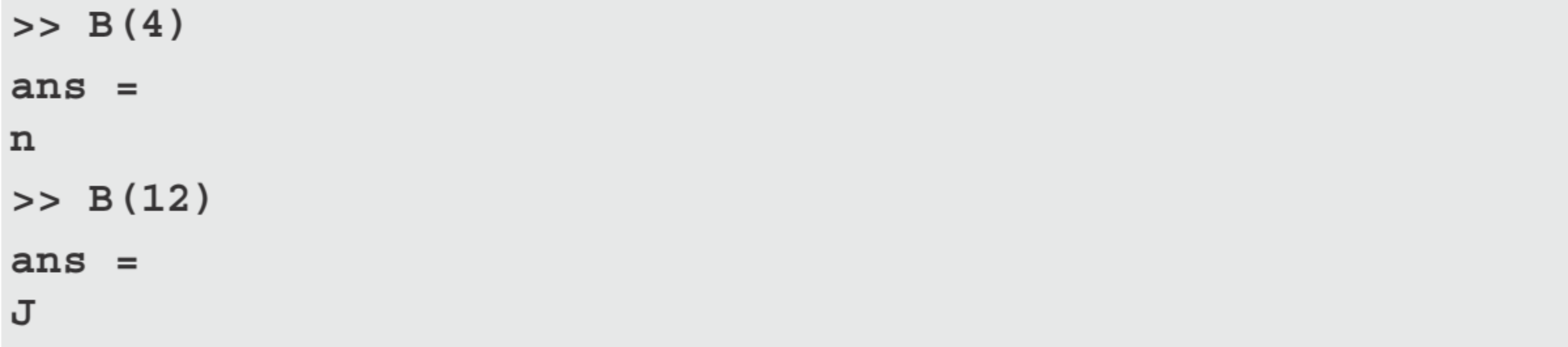


position	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*
vector	*M*	*y*		*n*	*a*	*m*	*e*		*i*	*s*

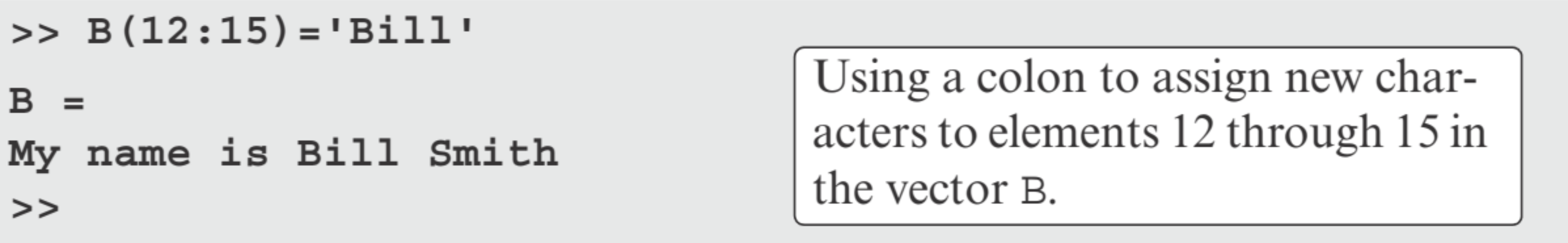
STRINGS AND STRINGS AS VARIABLES



```
>> a='FRty 8'

a =
FRty 8

>> B='My name is John Smith'

B =
My name is John Smith
>>
```



```
>> B(4)

ans =
n

>> B(12)

ans =
J
```



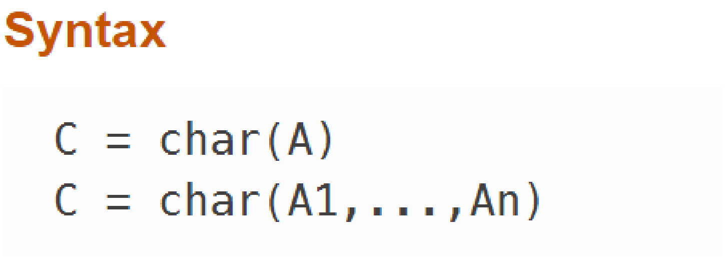
```
>> B(12:15)='Bill'

B =
My name is Bill Smith
>>
```

Using a colon to assign new characters to elements 12 through 15 in the vector B.

strings** can also be placed in a **matrix

all rows must have the same number of elements

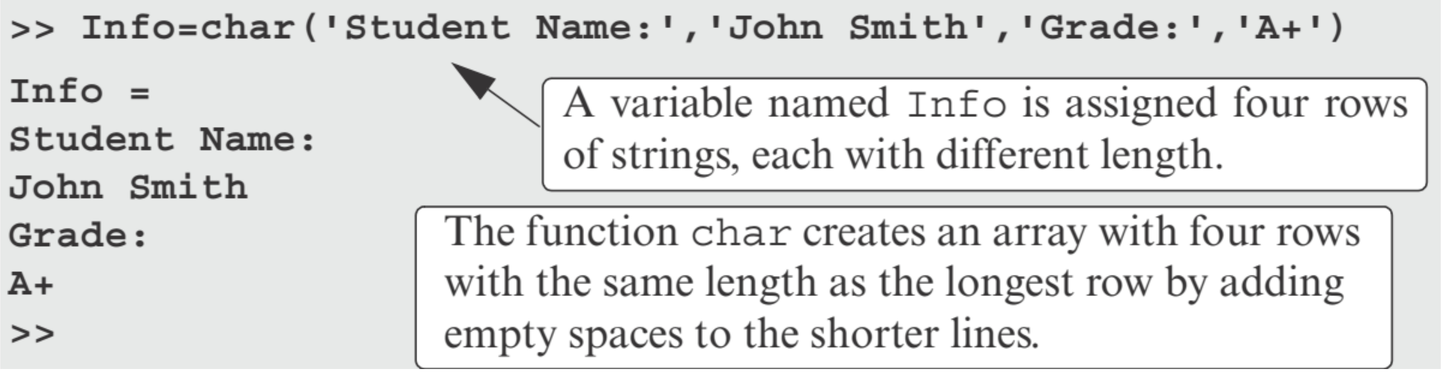
variable_name = char(*'string 1','string 2','string 3'*)



Syntax

```
C = char(A)
C = char(A1,...,An)
```

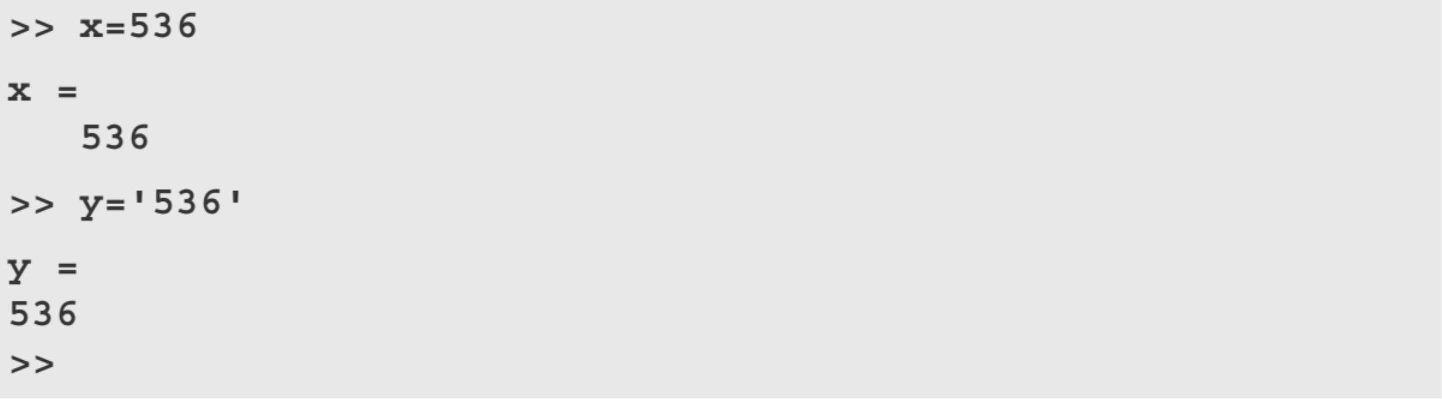
STRINGS AND STRINGS AS VARIABLES



```
>> Info=char('Student Name:','John Smith','Grade:','A+')
Info =
Student Name:
John Smith
Grade:
A+
>>
```

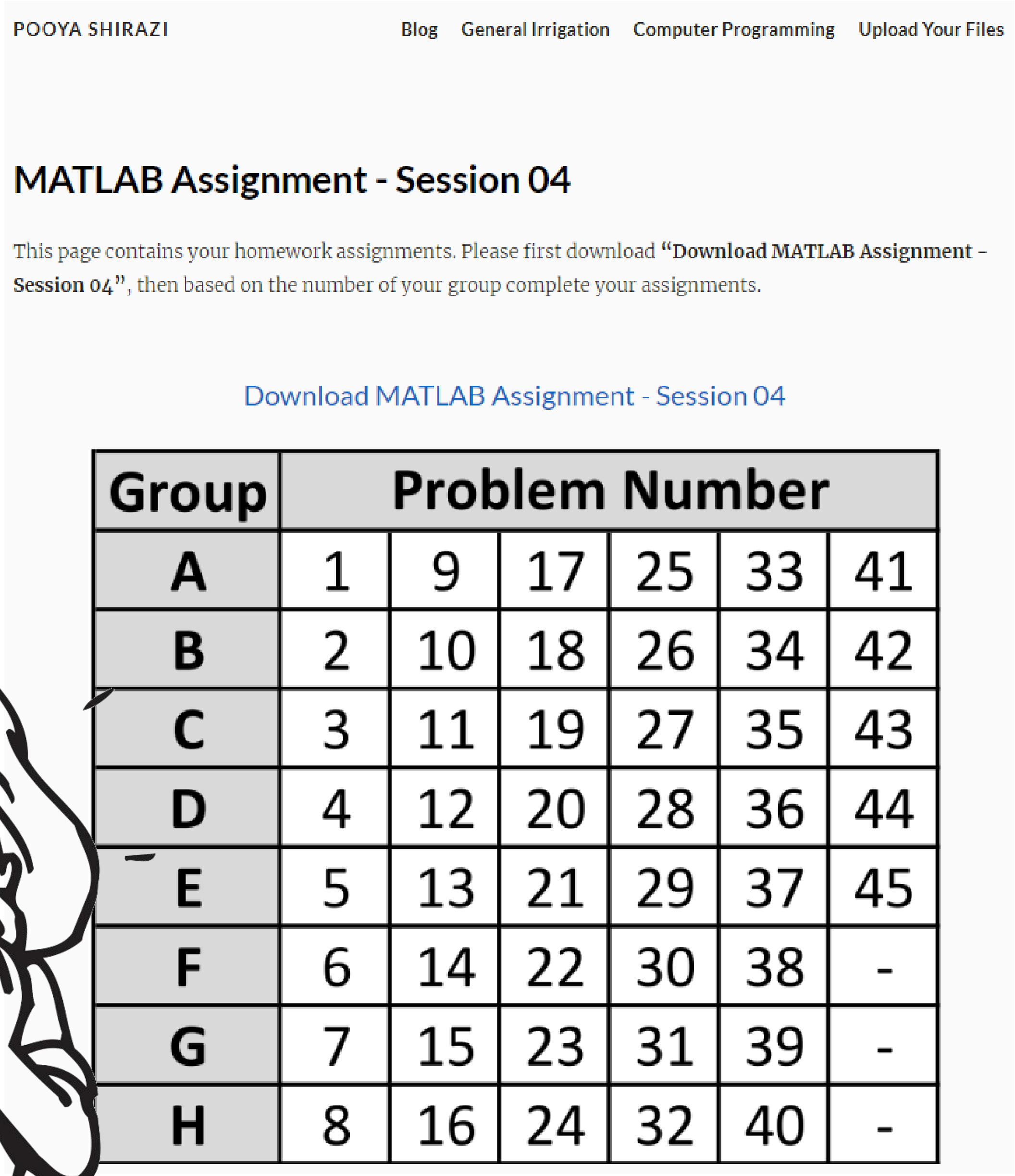
A variable named Info is assigned four rows of strings, each with different length.

The function char creates an array with four rows with the same length as the longest row by adding empty spaces to the shorter lines.



```
>> x=536
x =
   536
>> y='536'
y =
536
>>
```

PROBLEMS



MATLAB Assignment - Session 04

This page contains your homework assignments. Please first download **"Download MATLAB Assignment – Session 04"**, then based on the number of your group complete your assignments.

Download MATLAB Assignment - Session 04

Group	Problem Number					
A	1	9	17	25	33	41
B	2	10	18	26	34	42
C	3	11	19	27	35	43
D	4	12	20	28	36	44
E	5	13	21	29	37	45
F	6	14	22	30	38	-
G	7	15	23	31	39	-
H	8	16	24	32	40	-

QUIZ

Using the ones and zeros commands, create a 4×5 matrix in which the first two rows are 0s and the next two rows are 1s.

Create a 6×6 matrix in which the middle two rows and the middle two columns are 1s and the rest of the entries are 0s.

Given are a 5×6 matrix *A*, a 3×6 matrix *B*, and a 9-element vector *v*.

$$ma=\begin{bmatrix}2&5&8&11&14&17\\3&6&9&12&15&18\\4&7&10&13&16&19\\5&8&11&14&17&20\\6&9&12&15&18&21\end{bmatrix}\qquad B=\begin{bmatrix}5&10&15&20&25&30\\30&35&40&45&50&55\\55&60&65&70&75&80\end{bmatrix}$$

$$v=[99\quad 98\quad 97\quad 96\quad 95\quad 94\quad 93\quad 92\quad 91\ ]$$

Create the three arrays in the Command Window, and then, by writing one command, replace the last four columns of the first and third rows of *A* with the first four columns of the first two rows of *B*, the last four columns of the fourth row of *A* with the elements 5 through 8 of *v*, and the last four columns of the fifth row of *A* with columns 3 through 5 of the third row of *B*.