# *Creating :*
# *ARRAYS*

## *CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)*

A one-dimensional array is a list of numbers arranged in a row or a column.

Any list of numbers can be set up as a vector.

### 1- Creating a vector from a known list of numbers:

The vector is created by typing the elements (numbers) inside square brackets [ ].

```
variable_name = [ type vector elements ]

X=[1 2 3 4 5]
Y=[3 5 7 9 11 13]
```

**Row vector:**

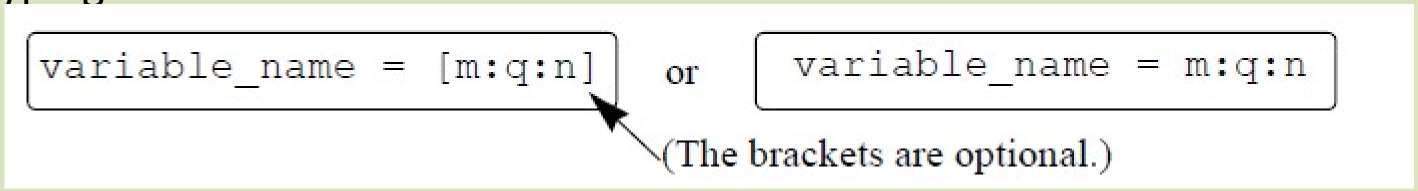
**To create a row vector type the elements with a space or a comma** between the elements inside the square brackets.

**Column vector:**

**To create a column vector type the left square bracket [ and then enter the elements with a semicolon between them, or press the Enter key after each element. Type the right square bracket ] after the last element.**

## 2- Creating a vector with constant spacing by specifying the first term, the spacing, and the last term:

In a vector with constant spacing the difference between the elements is the same. For example, in the vector *v = 2 4 6 8 10, the spacing between the elements is* 2. A vector in which the first term is *m, the spacing is q, and the last term is n is* created by typing:

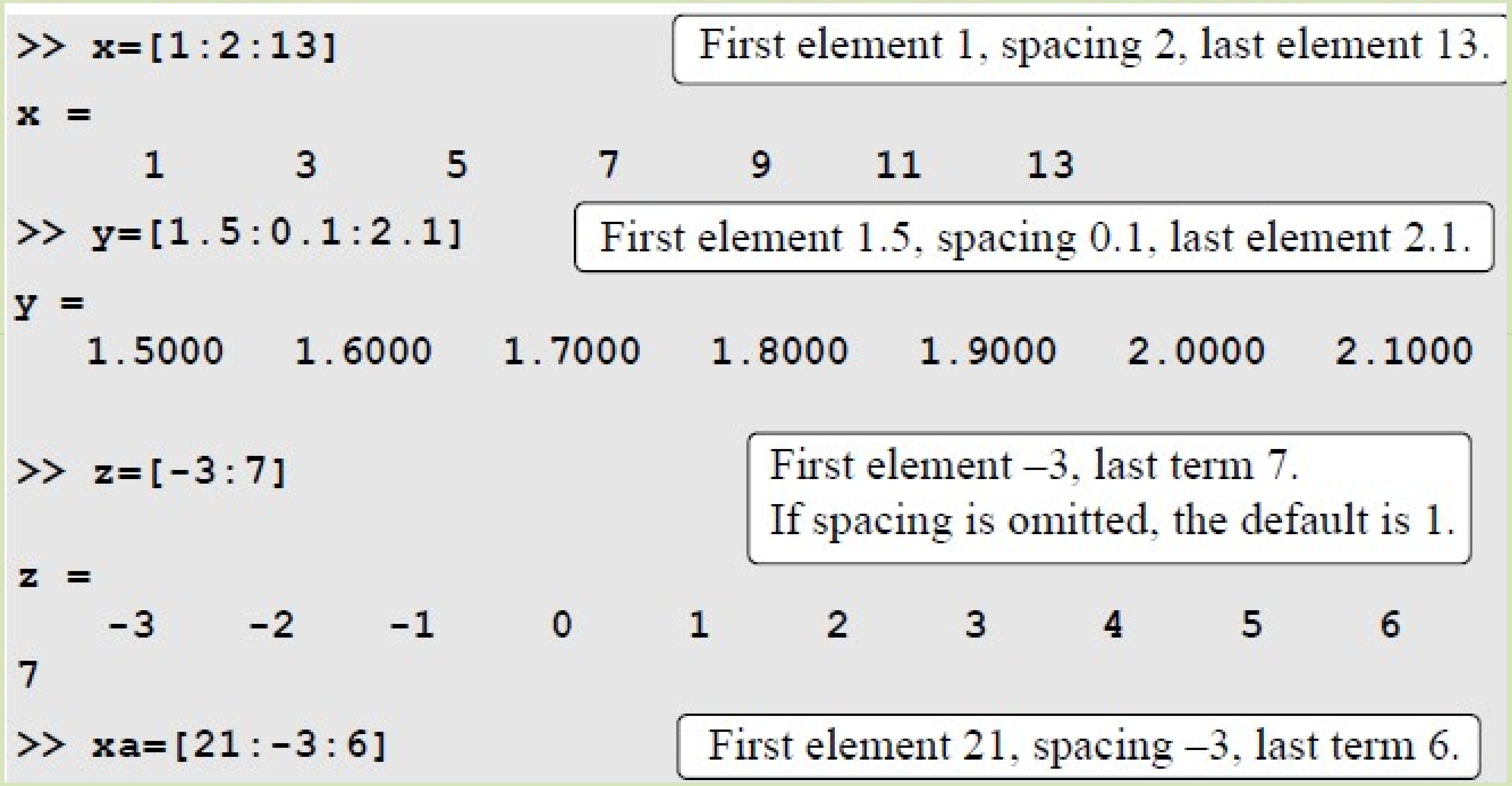


```
variable_name = [m:q:n]
```

or

```
variable_name = m:q:n
```

(The brackets are optional.)



```
>> 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]
```

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.

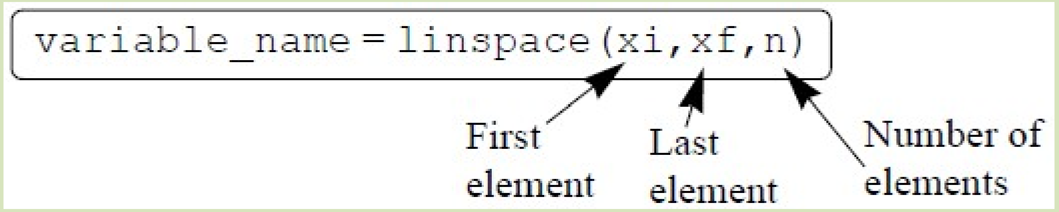
## Notes

**• If the numbers m, q, and n are such that the value of n cannot be obtained by** adding q's to m, then (for positive n) the last element in the vector will be the last number that does not exceed n.

**• If only two numbers (the first and the last terms) are typed (the spacing is omitted),**
then the default for the spacing is 1.

## 3- Creating a vector with linear (equal) spacing by specifying the first and last terms, and the number of terms:

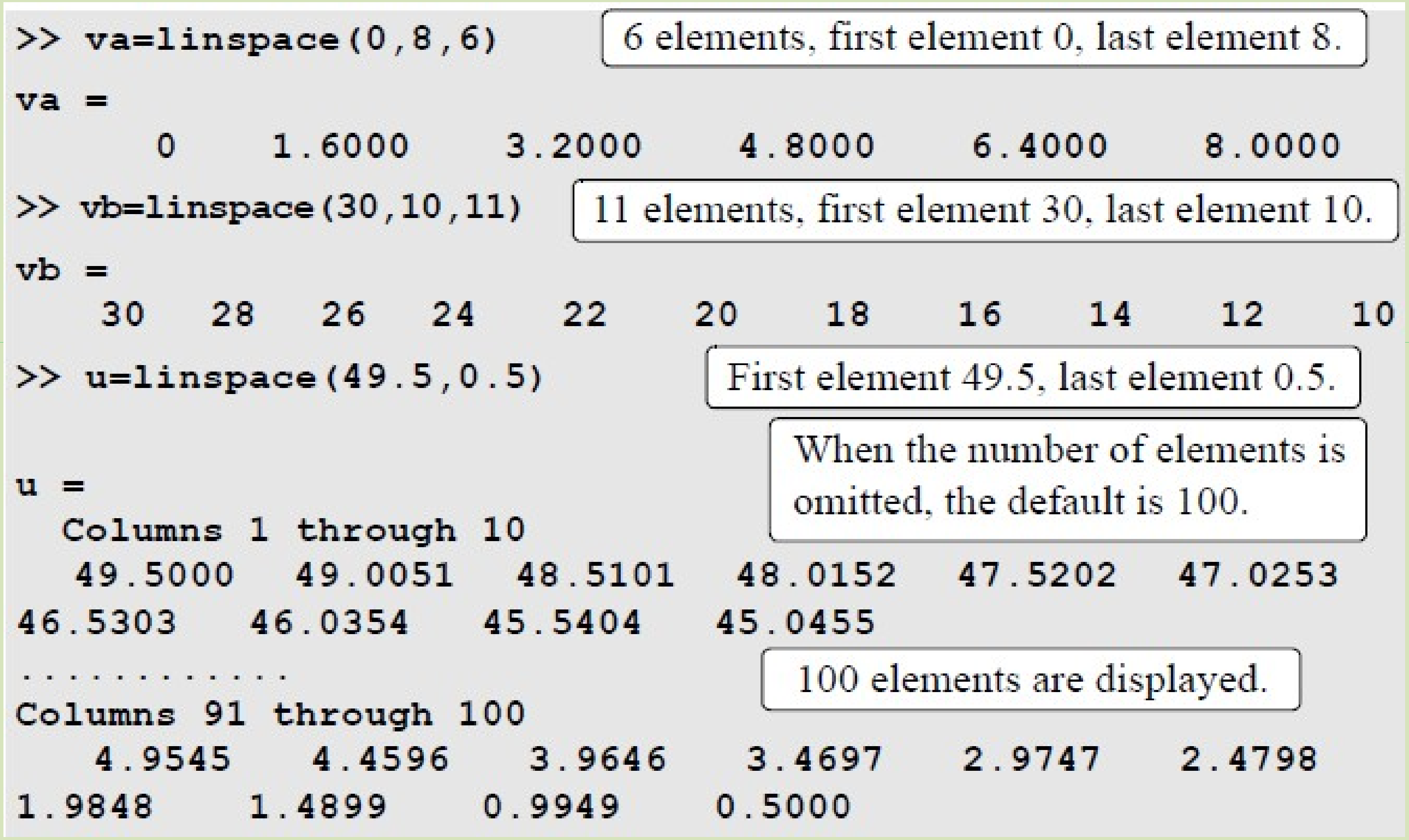
A vector with *n elements that are linearly (equally) spaced in which the first element* is *xi and the last element is xf can be created by typing the linspace command* (MATLAB determines the correct spacing):



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

**Note : When the number of elements is omitted, the default is 100**.

## Examples



```
>> va=linspace(0,8,6)

va =
         0    1.6000    3.2000    4.8000    6.4000    8.0000

>> vb=linspace(30,10,11)

vb =
    30   28   26   24    22    20    18    16    14    12    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
```

6 elements, first element 0, last element 8.

11 elements, first element 30, last element 10.

First element 49.5, last element 0.5.

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

100 elements are displayed.

## *TWO-DIMENSIONAL ARRAY (MATRIX)*

A two-dimensional array, also called a matrix, has numbers in rows and columns. Matrices can be used to store information like the arrangement in a table. Matrices play an important role in linear algebra and are used in science and engineering to describe many physical quantities.

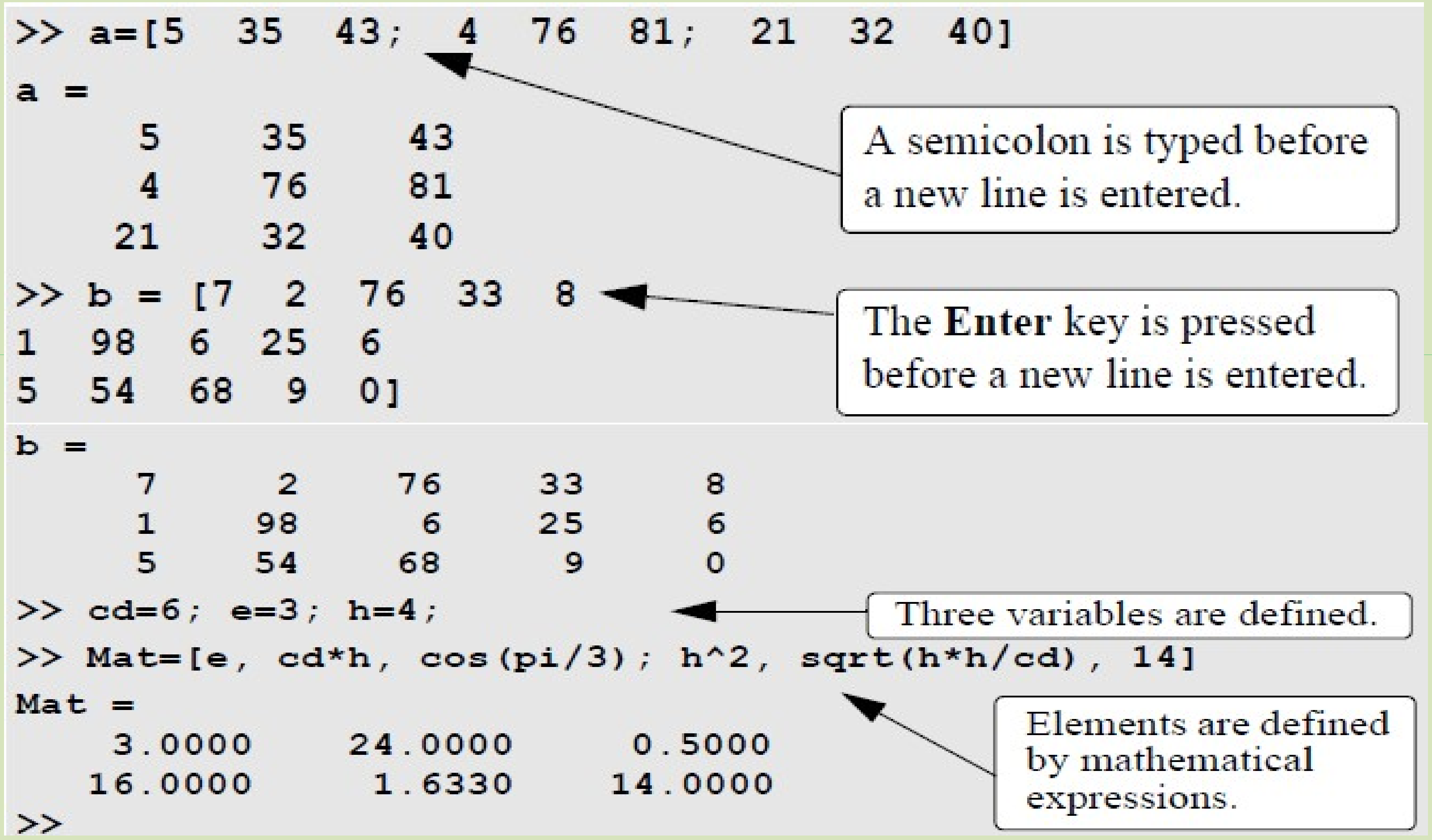
A $m \times n$ matrix has *m rows and n columns, and m* by *n is called the size of the matrix.*

A matrix is created by assigning the elements of the matrix to a variable.
This is done by typing the elements, row by row, inside square brackets [ ].
First type the left bracket [ then type the first row, separating the elements with spaces or commas. To type the next row type a semicolon or press **Enter.**
Type the right bracket ] at the end of the last row.

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

*Note : All the rows must have the same number of elements.*

## Examples :



```
>> 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
>>
```

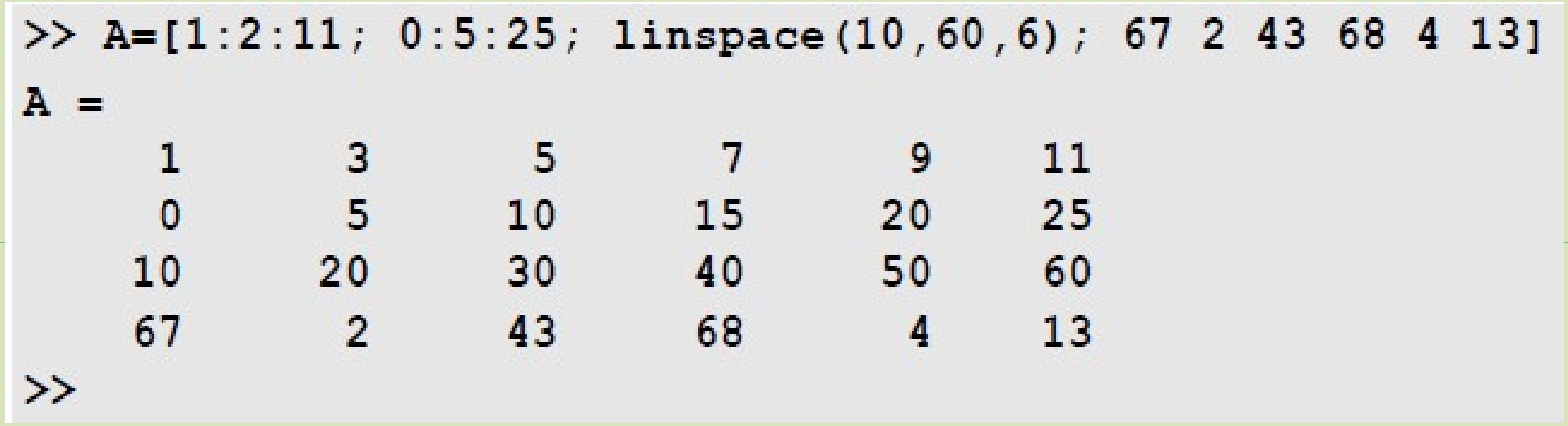
A semicolon is typed before a new line is entered.

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

Three variables are defined.

Elements are defined by mathematical expressions.

Rows of a matrix can also be entered as vectors using the notation for creating vectors with constant spacing, or the linspace command. For example:



```
>> 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
>>
```

In this example

1- the first two rows were entered as vectors using the notation of constant spacing,

2- the third row was entered using the linspace command,

3- in the last row the elements were entered individually.

## *The zeros, ones and eye Commands*

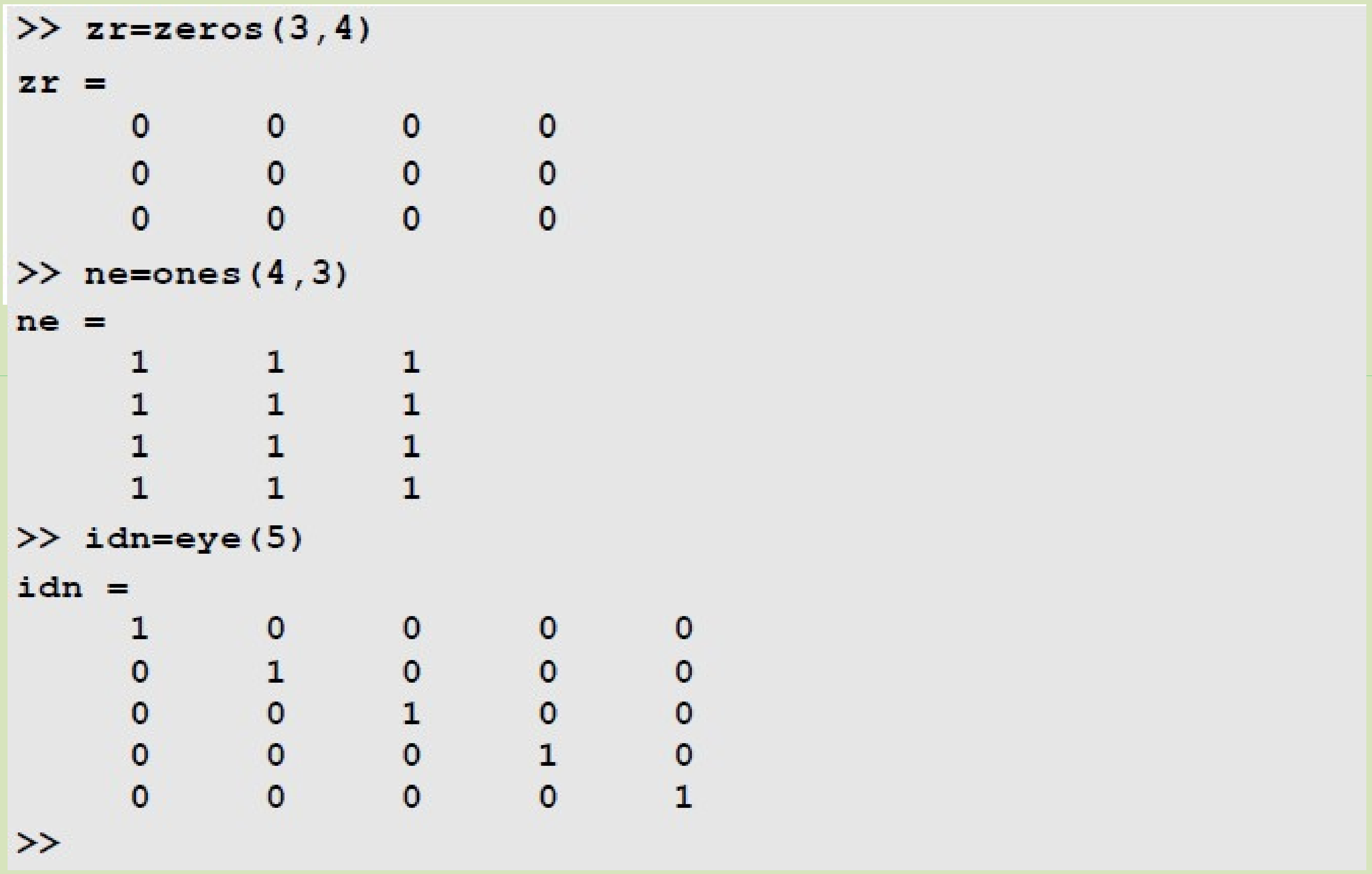
commands can be used to create matrices that have elements with special values.

**The zeros(m,n)**
create a matrix with *m rows and n columns in which all* elements are the number 0

**The ones(m,n)**
create a matrix with *m rows and n columns in which all* elements are the number 1,.

**The eye(n)**
command creates a square matrix with *n rows and n columns in which the diagonal elements are equal* to 1 and the rest of the elements are 0. This matrix is called the identity matrix.

## Examples



```
>> 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.**
  - ❖ A scalar is an array with one element,
  - ❖ a vector is an array with one row or one column of elements, and
  - ❖a matrix is an array with elements in rows and columns.
- **The variable (scalar, vector, or matrix) is defined by the input when the variable is assigned.**

There is no need to define the size of the array (single element for a scalar, a row or a column of elements for a vector, or a two-dimensional array of elements for a matrix) before the elements are assigned.

- **Once a variable exists[as a scalar, vector, or matrix]it can be changed to any other size, or type, of variable**. For example,
  - ❖ a scalar can be changed to a vector
  - ❖ a matrix; a vector can be changed to a scalar, a vector of different length, a matrix;
  - ❖ a matrix can be changed to have a different size, or reduced to a vector or a scalar.
  - ❖***These changes are made by adding or deleting elements.***

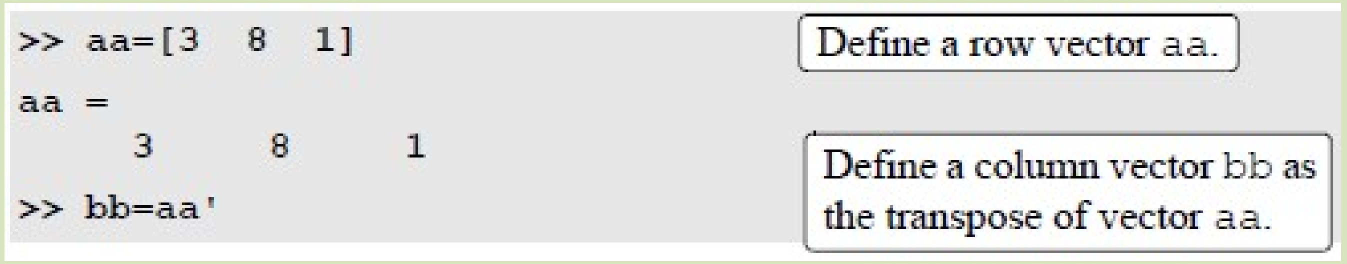
## *THE TRANSPOSE OPERATOR*

The transpose operator, when applied to a vector,

- ❖It switches a row vector to a column vector
- ❖It also switches column vector to a row vector.
- ❖When applied to a matrix, it switches the rows (columns) to columns (rows).

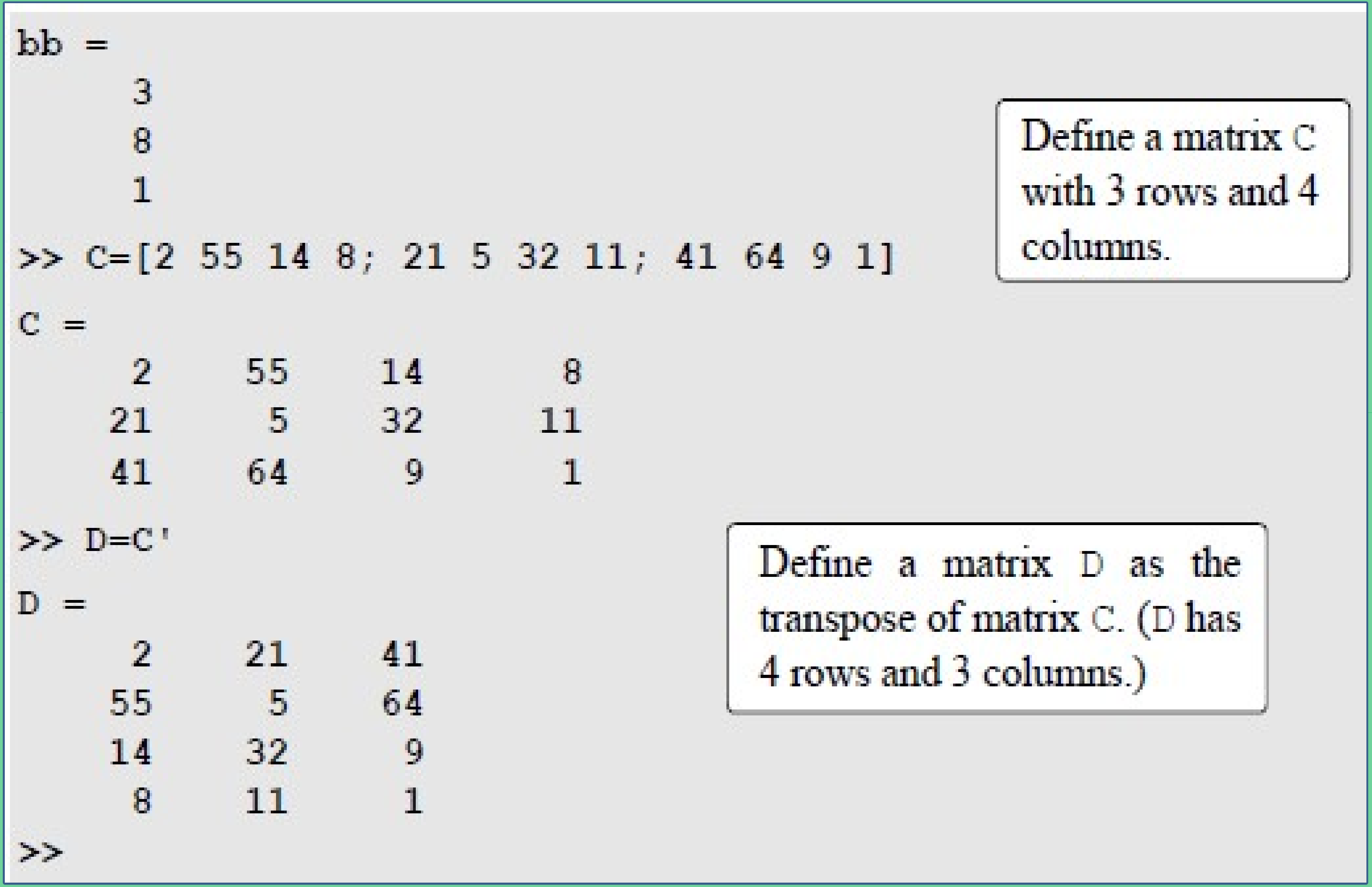
The transpose operator is applied by typing a single quote ’ following the variable to be transposed.

**Examples**



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

aa =
        3       8       1

>> bb=aa'
```

## Computer Programming EEN170



```
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 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.)

## ***ARRAY ADDRESSING***

❖Elements in an array (either vector or matrix) can be addressed individually or in subgroups.

❖ This is useful when there is a need to redefine only some of the elements,

when specific elements are to be used in calculations, or when a subgroup of the elements is used to define a new variable.

### ***Vector***

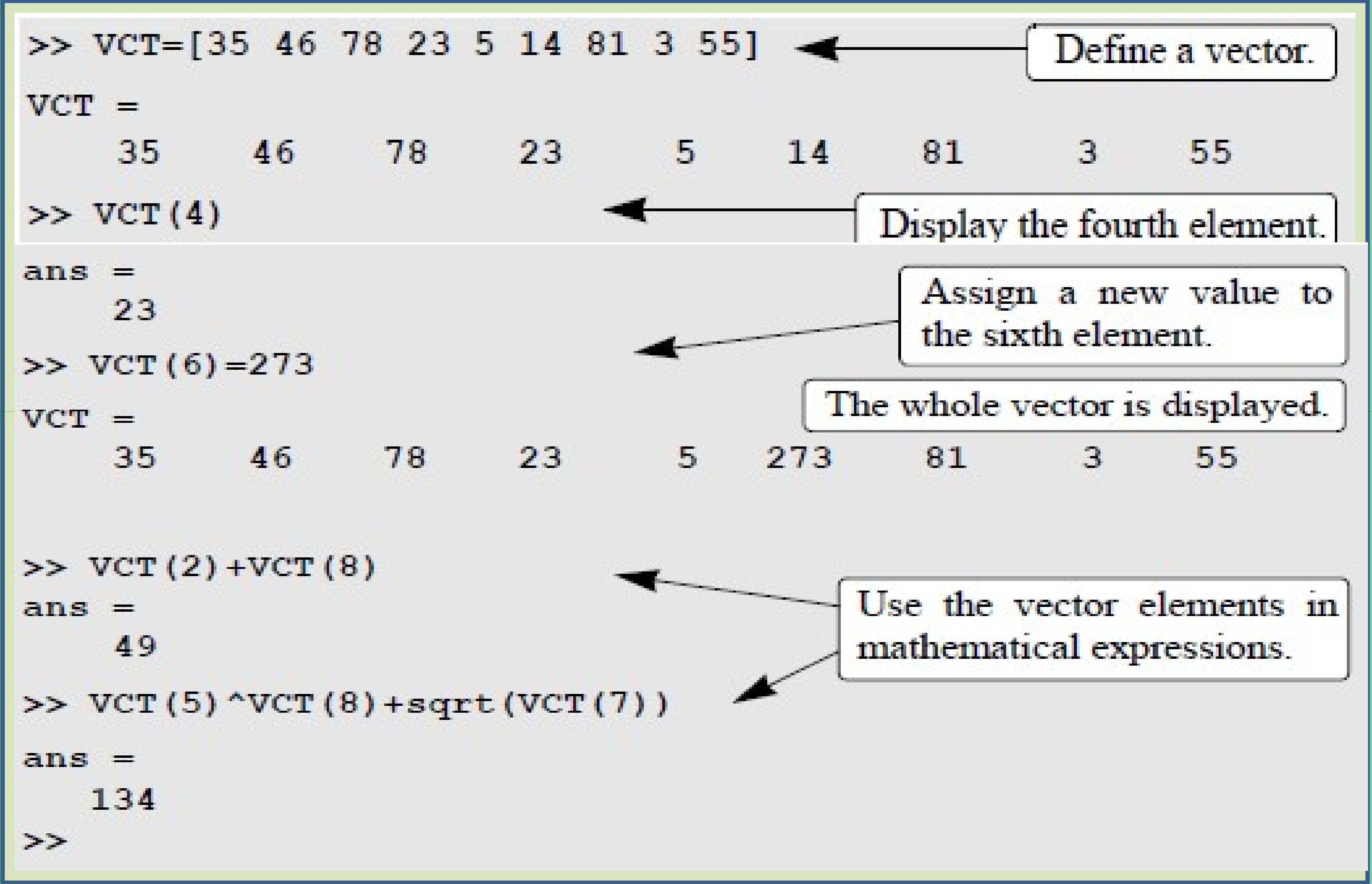
The address of an element in a vector is its position in the row (or column). For a vector named ***ve***, ***ve(k)*** *refers to the element in position k. The first position is* 1.

***For example***, if the vector *ve has nine elements:*

*ve = 35 46 78 23 5 14 81 3 55*

Then: *ve(4) = 23, ve(7) = 81, and ve(1) = 35.*

**A single vector element, *v(k), can be used just as a variable. For example, it* is possible to change the value of only one element of a vector by assigning a new value to a specific address.**



```
>> 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.

## Matrix

❖The address of an element in a matrix is its position,

❖defined by the row number and the column number where it is located.

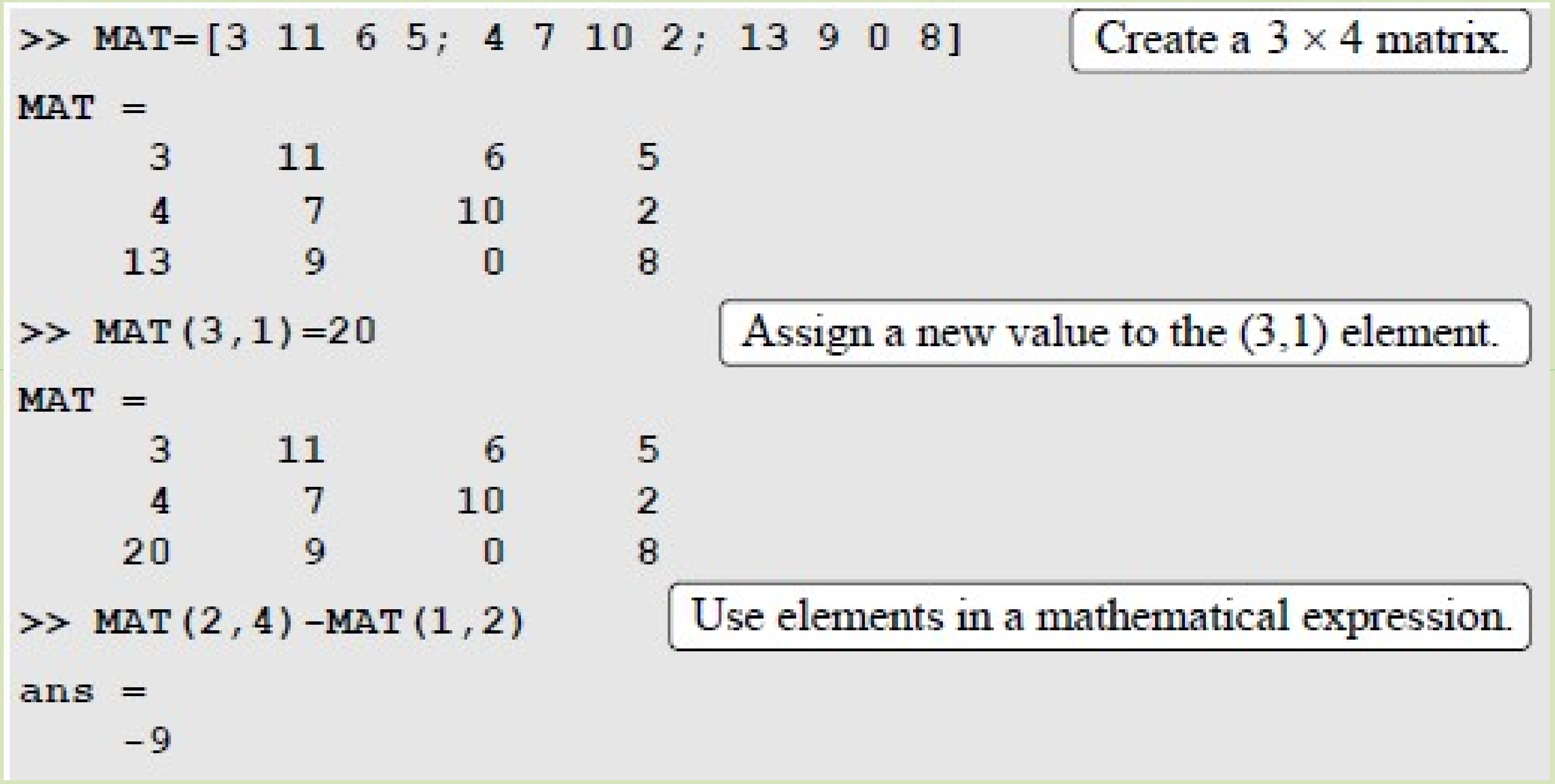
❖ For a matrix assigned to a variable *ma,ma(k,p) refers to the element in row k and column p.*

**For example**, if the matrix is:

$$ma = \begin{bmatrix} 3 & 11 & 6 & 5 \\ 4 & 7 & 10 & 2 \\ 13 & 9 & 0 & 8 \end{bmatrix}$$

then *ma(1,1) = 3 and ma(2,3) = 10.*
As with vectors, it is possible to change the value of just one element of a matrix by assigning a new value to that element. Also, single elements can be used like variables in mathematical expressions and functions

## Examples:



```
>> 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 $3 \times 4$ matrix.

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

Use elements in a mathematical expression.

## *USING A COLON : IN ADDRESSING ARRAYS*

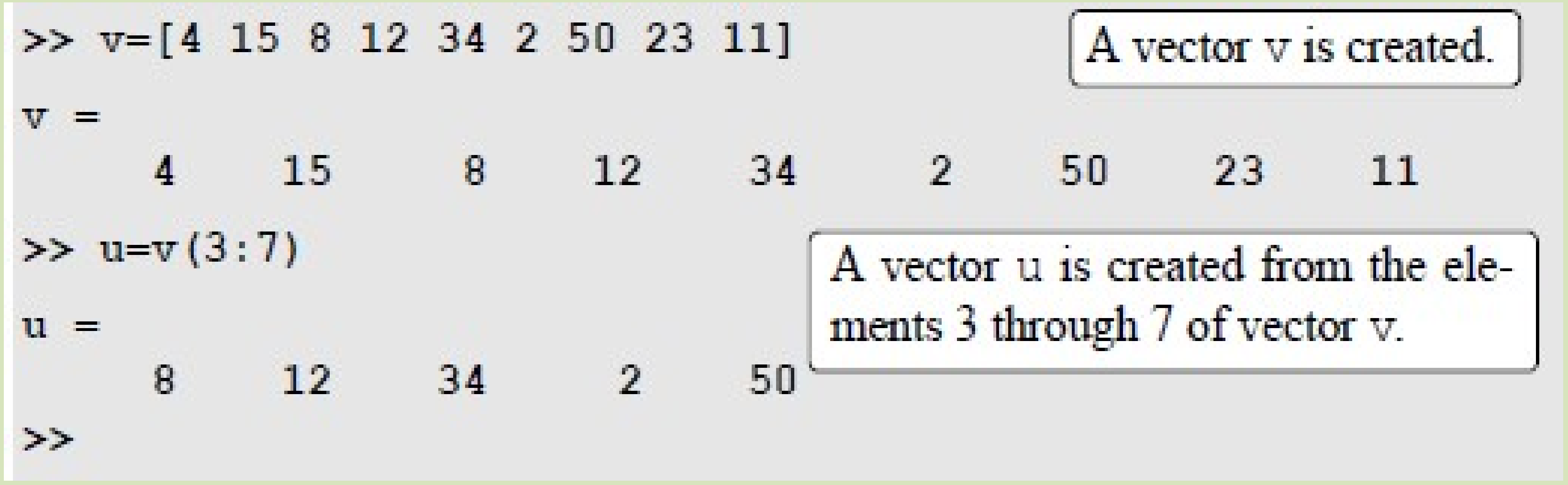
A colon can be used to address a range of elements in a vector or a matrix.

**For a vector:**

*va(:) Refers to all the elements of the vector va (either a row or a column vector).*

*va(m:n) Refers to elements m through n of the vector va.*

**Example**:



```
>> 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

>>
```

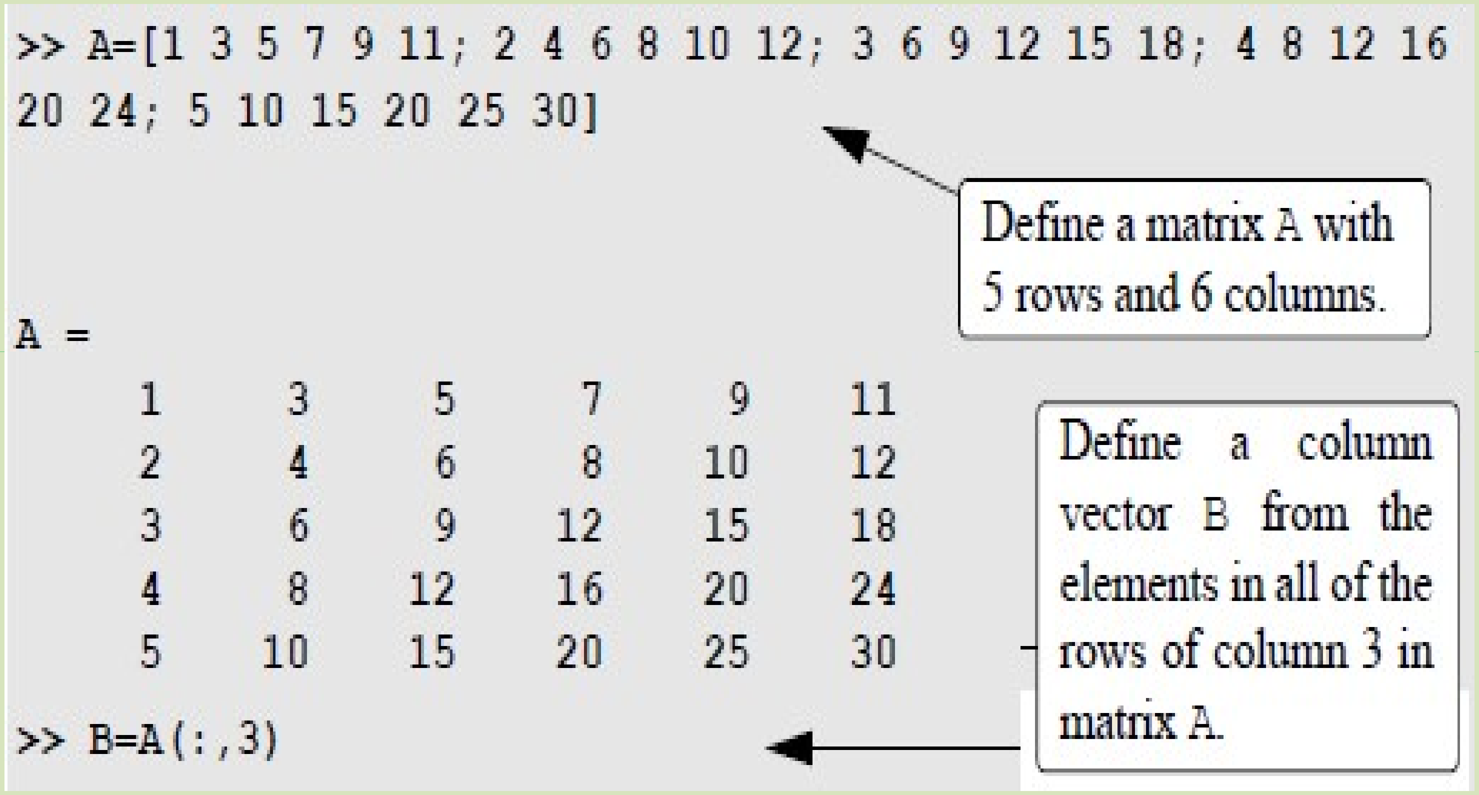
A vector v is created.

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

## For a matrix :

- *A(:,n) Refers to the elements in all the rows of column n of the matrix A.*
- *A(n,:) Refers to the elements in all the columns of row n of the matrix A.*
- *A(:,m:n) Refers to the elements in all the rows between columns m and n of the* matrix *A.*
- *A(m:n,:) Refers to the elements in all the columns between rows m and n of the* matrix *A.*
- A(*m:n,p:q) Refers to the elements in rows m through n and columns p through q of the matrix A.*

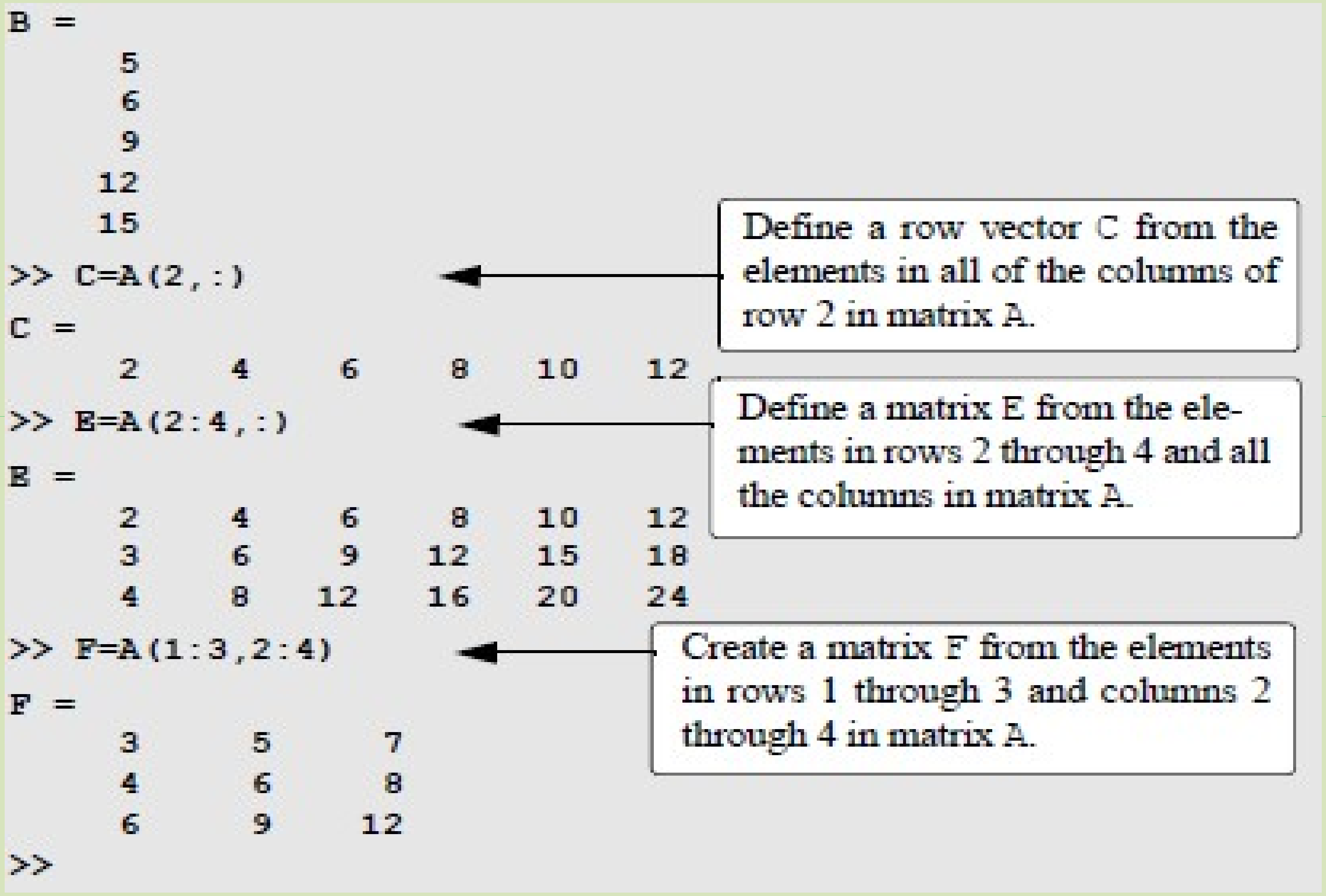
**The use of the colon symbol in addressing elements of matrices is demonstrated In following examples .**



```
>> 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.

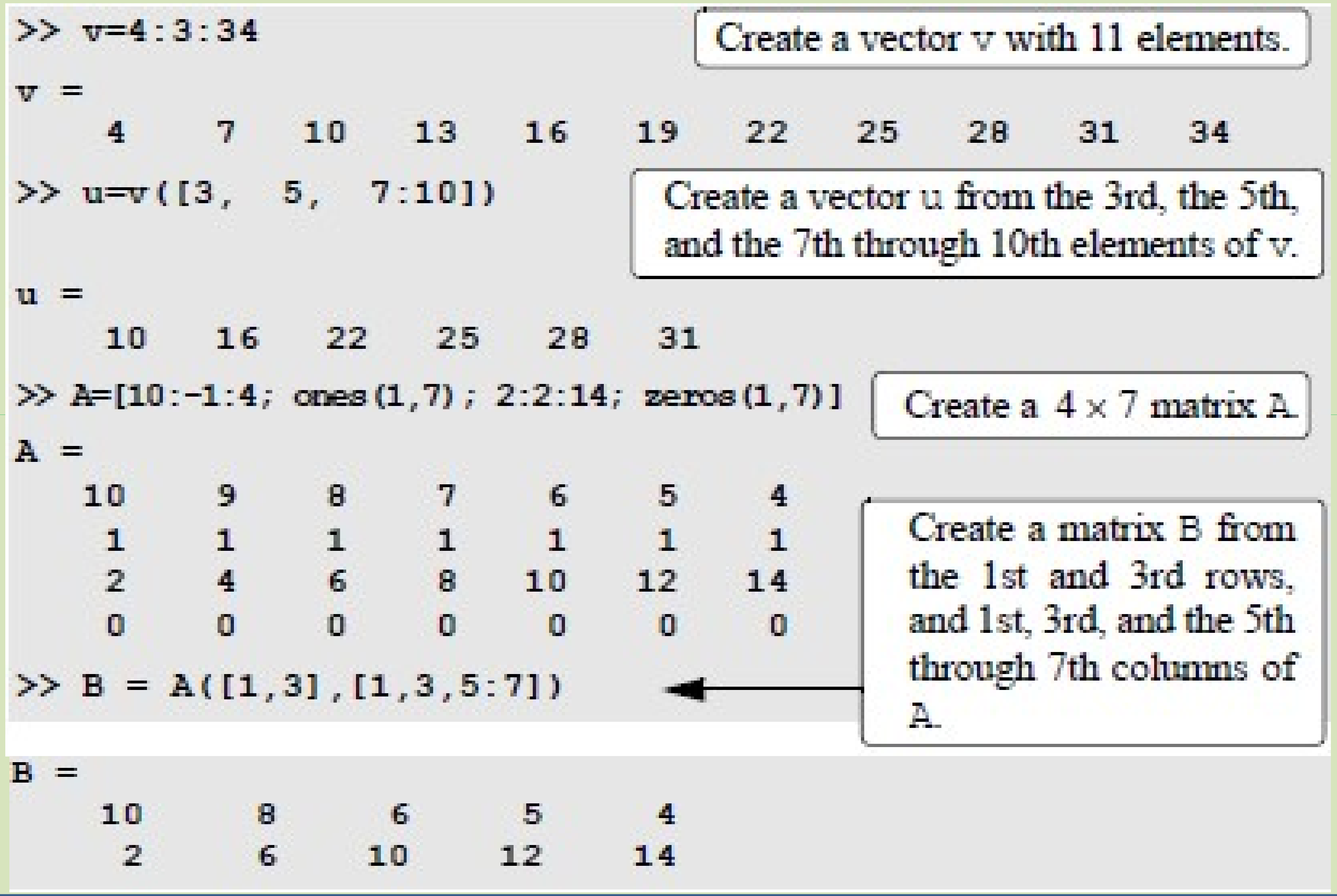


```
B =
     5
     6
     9
    12
    15
>> C=A(2,:)
C =
     2     4     6     8    10    12
>> E=A(2:4,:)
E =
     2     4     6     8    10    12
     3     6     9    12    15    18
     4     8    12    16    20    24
>> F=A(1:3,2:4)
F =
     3     5     7
     4     6     8
     6     9    12
>>
```

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.



```
>> 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 $4 \times 7$ 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*

❖A variable that exists as a vector, or a matrix, can be changed by adding elements to it (remember that a scalar is a vector with one element).

❖A vector can be changed to have more elements, or it can be changed to be a matrix.

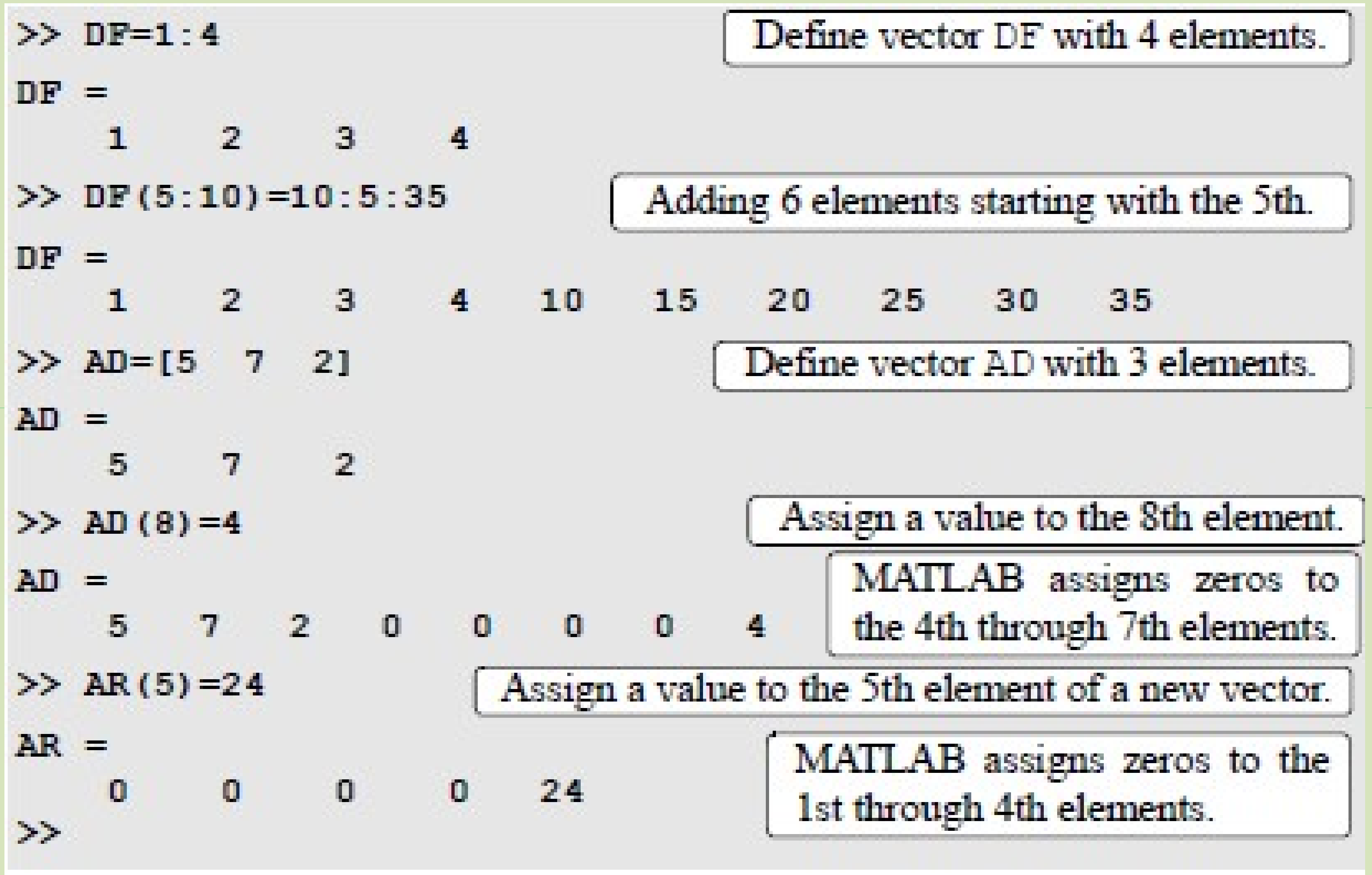
❖Rows and/or columns can also be added to an existing matrix to obtain a matrix of different size.

❖ The addition of elements can be done by simply assigning values to the additional elements, or by appending existing variables.

### Adding elements to a vector:

❖Elements can be added to an existing vector by assigning values to the new elements. For example, if a vector has 4 elements, the vector can be made longer by assigning values to elements 5, 6, and so on.

**If a vector has *n elements and a new* value is assigned to an element with an address of or larger, MATLAB assigns zeros to the elements that are between the last original element and the new element**



```
>> 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.

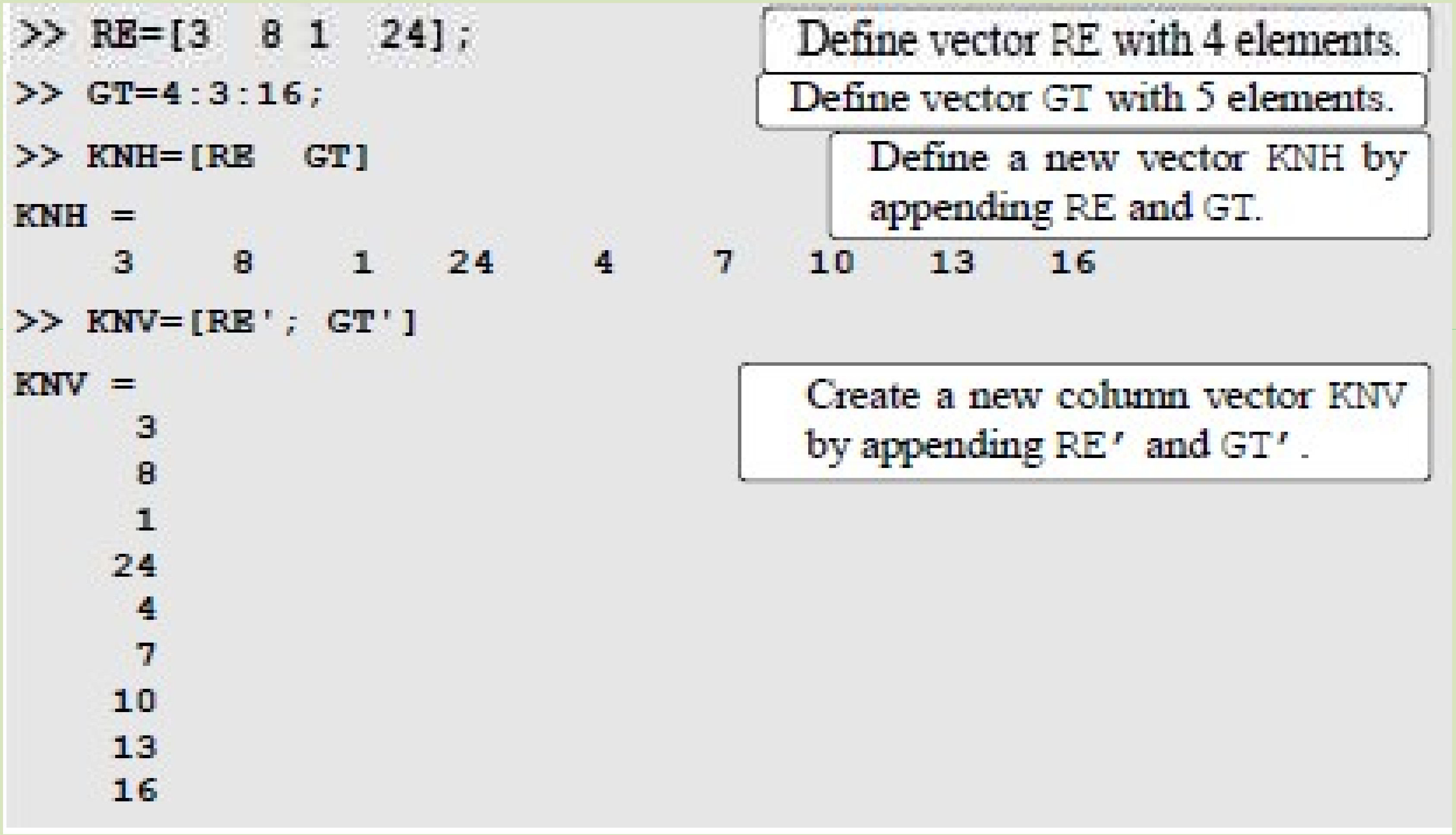
Assign a value to the 8th element.

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.

**Elements can also be added to a vector by appending existing vectors. Two examples are:**



```
>> 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.

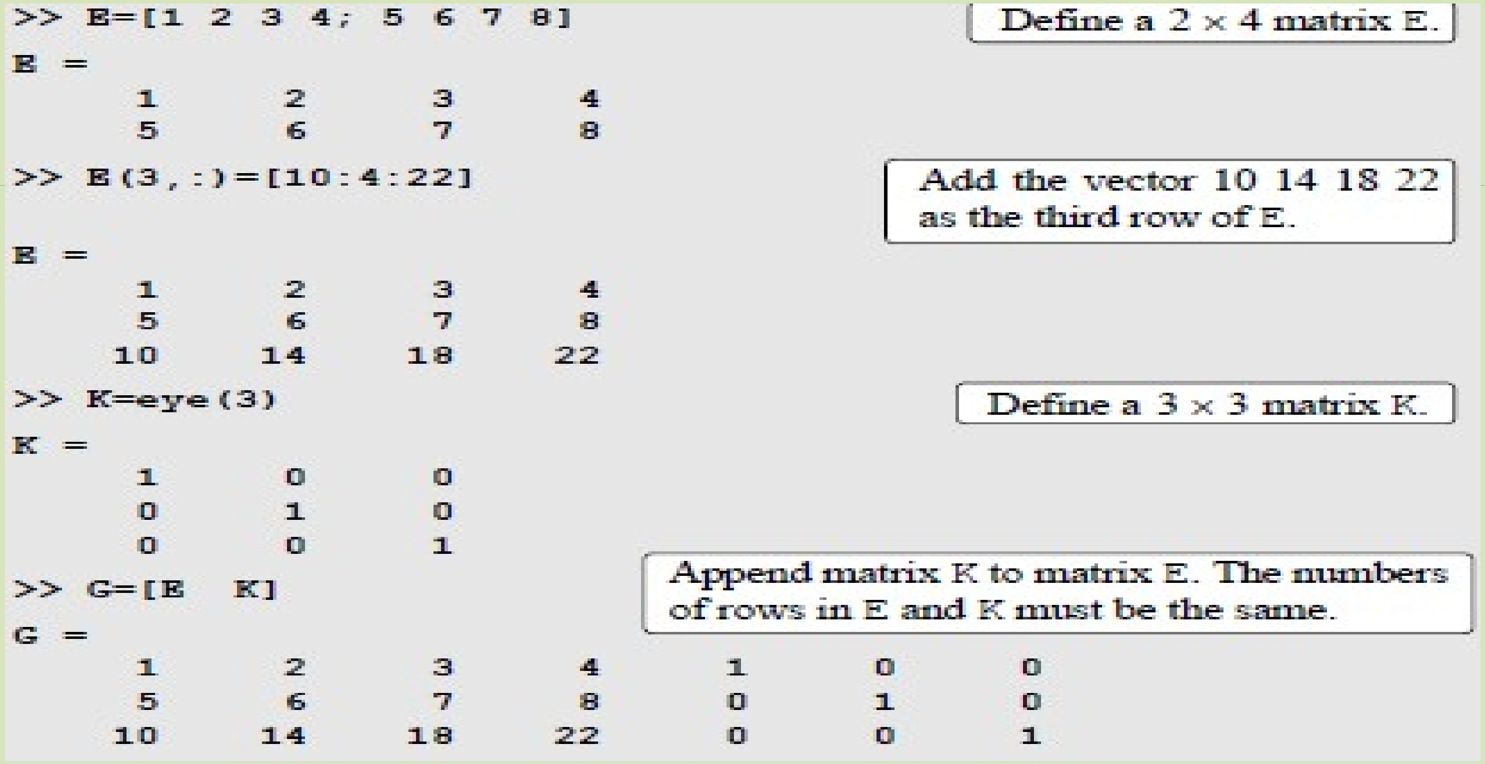
Define vector GT with 5 elements.

Define a new vector KNH by appending RE and GT.

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

## Adding elements to a matrix:

Rows and/or columns can be added to an existing matrix by assigning values to the new rows or columns. This can be done by assigning new values, or by appending existing variables. ***This must be done carefully since the size of the added rows or columns must fit the existing 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.

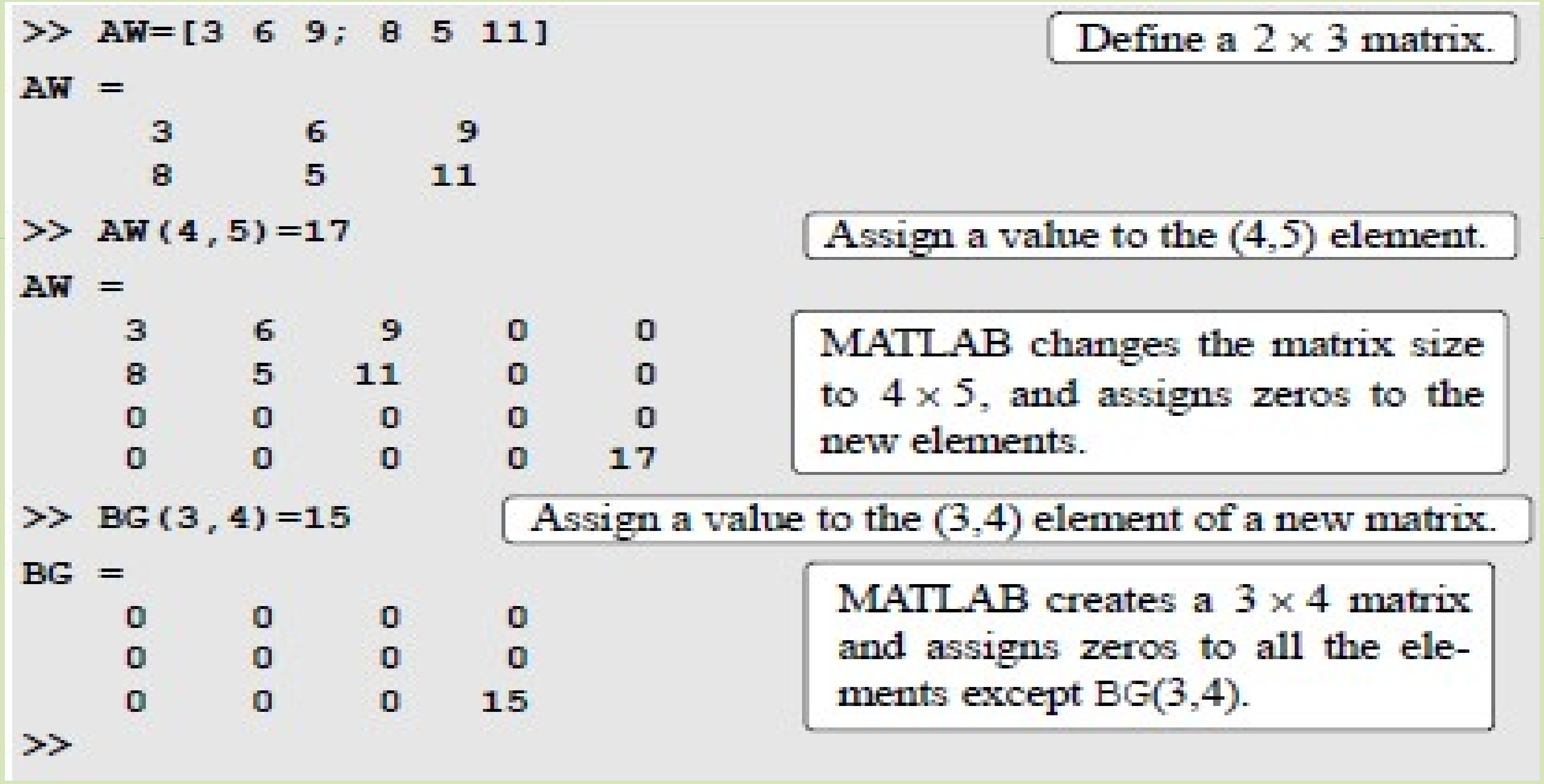
Add the vector 10 14 18 22 as the third row of E.

Define a 3 × 3 matrix K.

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

Computer Programming EEN170

**If a matrix has a size of , and a new value is assigned to an element with an address beyond the size of the matrix, MATLAB increases the size of the matrix to include the new element. Zeros are assigned to the other elements that are added**



```
>> 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 $2 \times 3$ matrix.

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

MATLAB changes the matrix size to $4 \times 5$, and assigns zeros to the new elements.

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

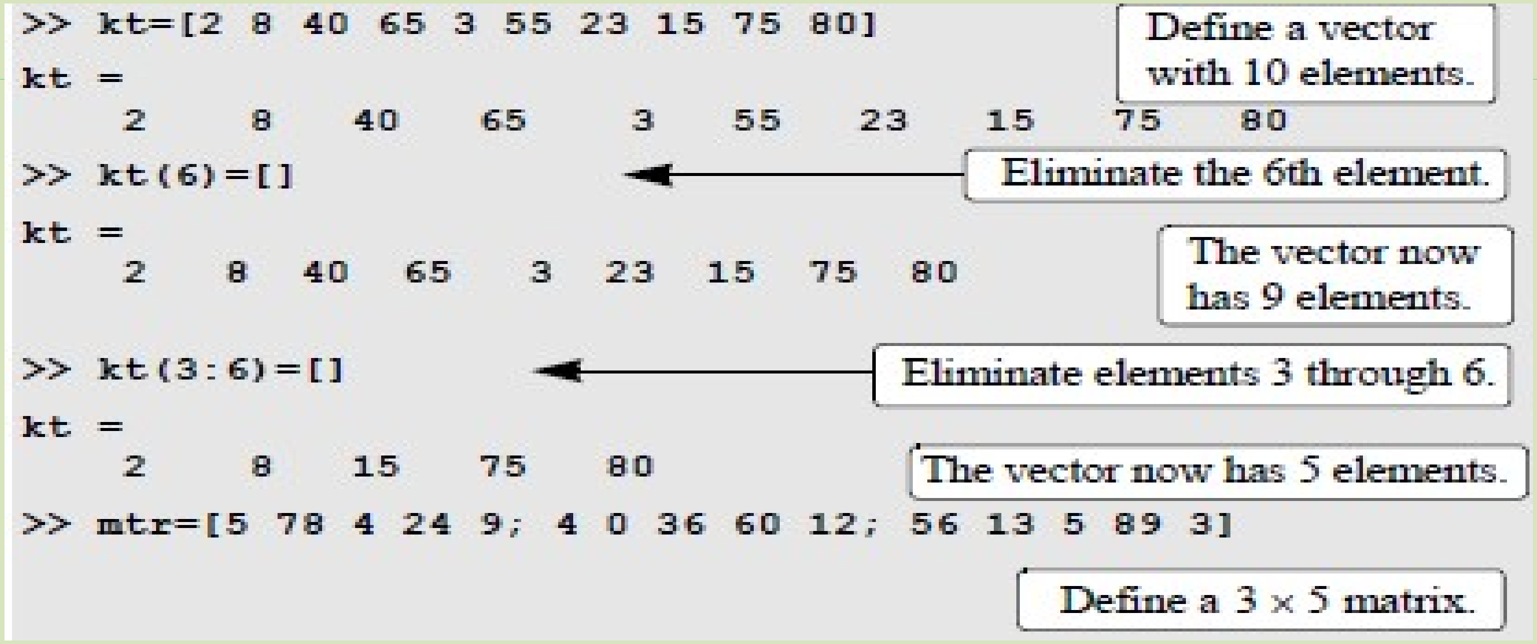
MATLAB creates a $3 \times 4$ matrix and assigns zeros to all the elements except BG(3,4).

## DELETING ELEMENTS

❖An element, or a range of elements, of an existing variable can be deleted by reassigning nothing to these elements.

❖ This is done by using square brackets with nothing typed in between them.

❖By deleting elements a vector can be made shorter and a matrix can be made to have a smaller size.

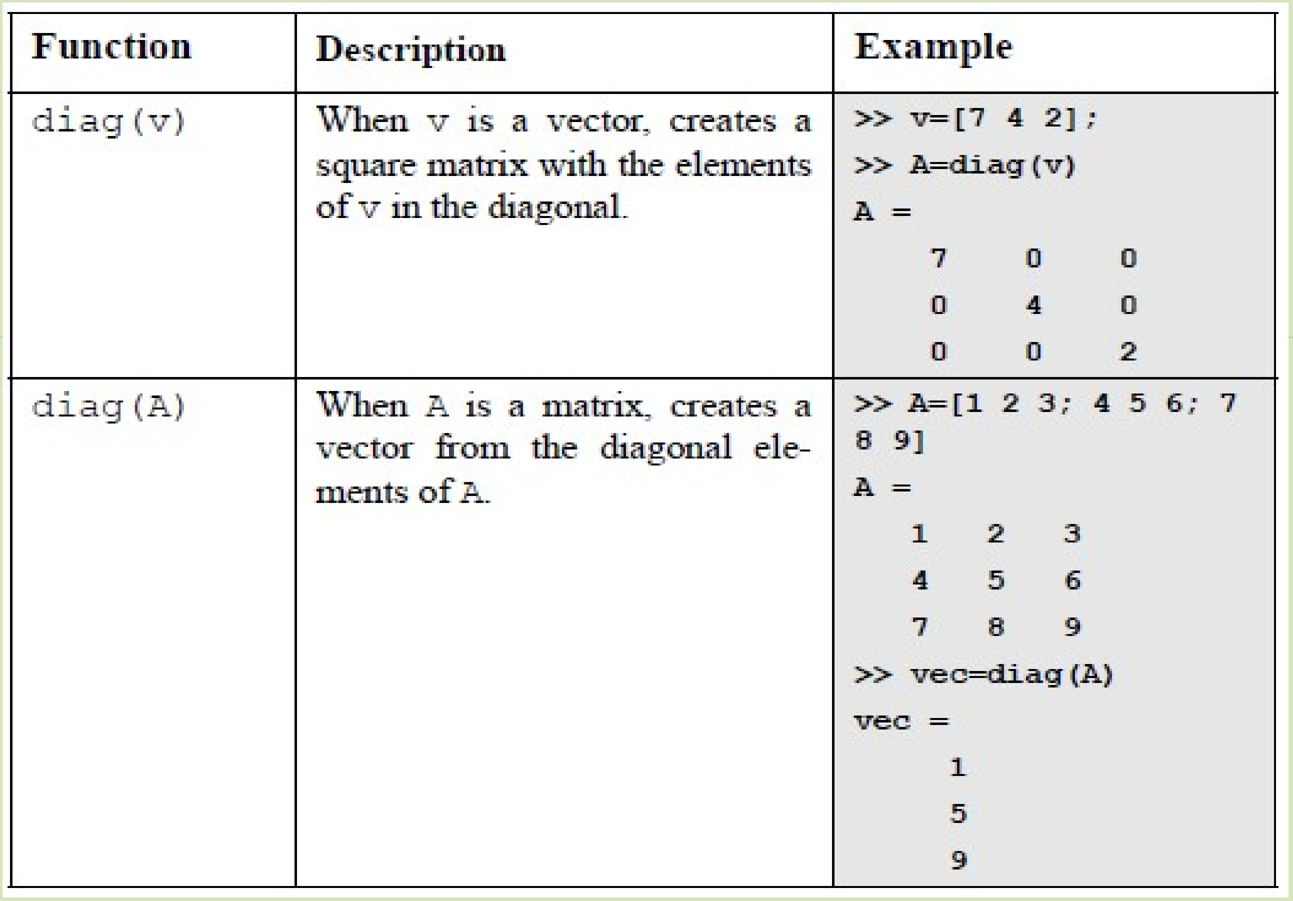


```
>> kt=[2 8 40 65 3 55 23 15 75 80]
kt =
     2     8    40    65     3    55    23    15    75    80
>> kt(6)=[]
kt =
     2     8    40    65     3    23    15    75    80

>> 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]
```

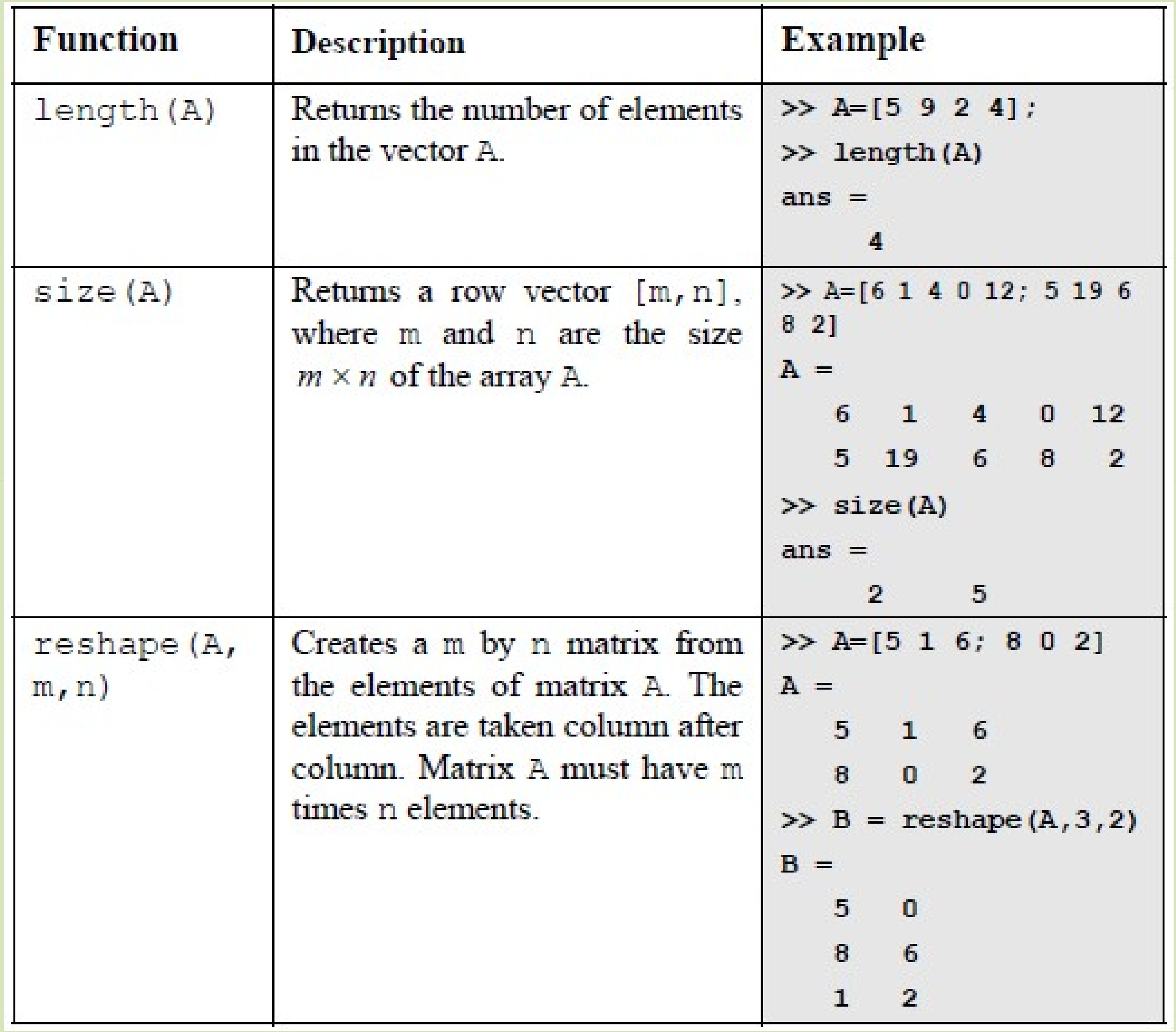
Define a vector with 10 elements.
Eliminate the 6th element.
The vector now has 9 elements.
Eliminate elements 3 through 6.
The vector now has 5 elements.
Define a 3 × 5 matrix.

## *BUILT-IN FUNCTIONS FOR HANDLING ARRAYS*

MATLAB has many built-in functions for managing and handling arrays. Some of these are listed below:



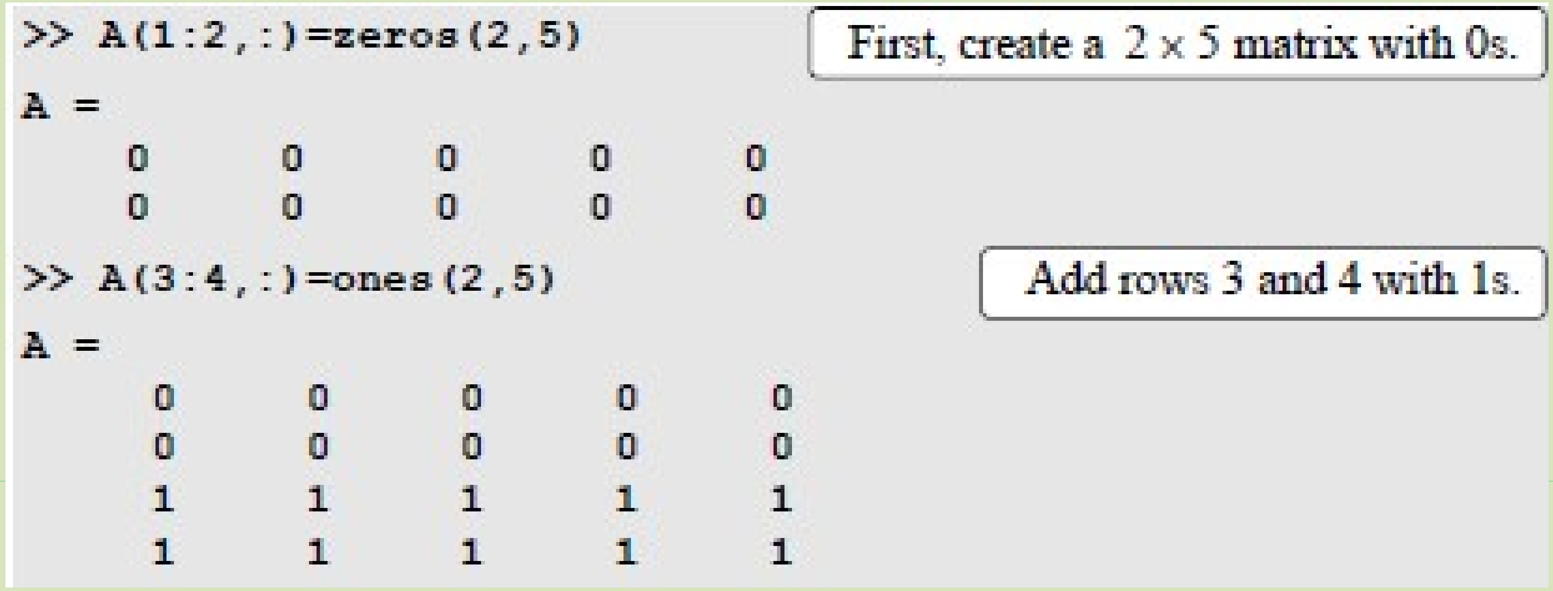
| Function | Description | Example |
|---|---|---|
| `diag(v)` | When `v` is a vector, creates a square matrix with the elements of `v` in the diagonal. | `>> v=[7 4 2];`<br>`>> A=diag(v)`<br>`A =`<br>`     7     0     0`<br>`     0     4     0`<br>`     0     0     2` |
| `diag(A)` | When `A` is a matrix,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



| Function | Description | Example |
|---|---|---|
| `length(A)` | Returns the number of elements in the vector `A`. | `>> A=[5 9 2 4];`<br>`>> length(A)`<br>`ans =`<br>`     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]`<br>`A =`<br>`   6   1   4   0  12`<br>`   5  19   6   8   2`<br>`>> size(A)`<br>`ans =`<br>`     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]`<br>`A =`<br>`   5   1   6`<br>`   8   0   2`<br>`>> B = reshape(A,3,2)`<br>`B =`<br>`   5   0`<br>`   8   6`<br>`   1   2` |

## Example

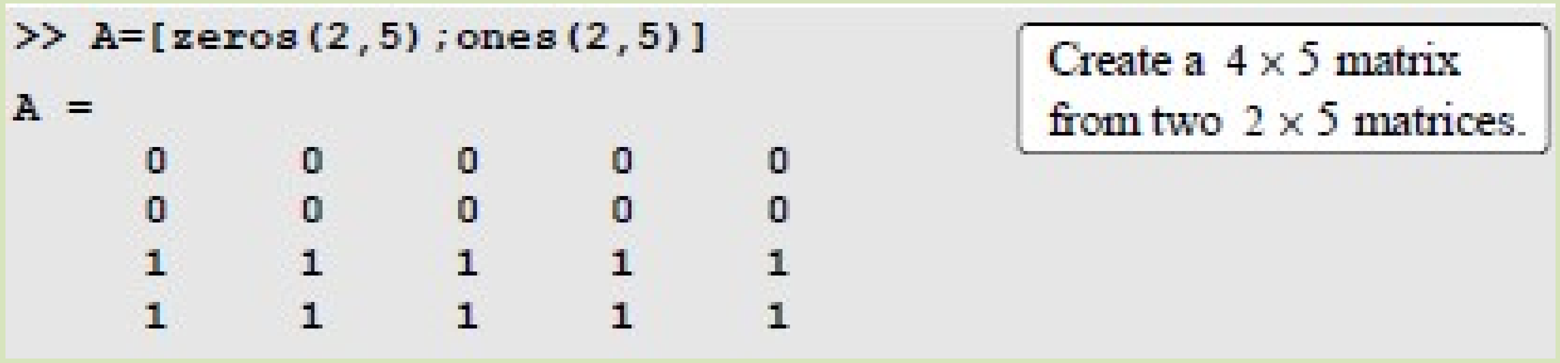
Using the ones and zeros commands, create a 4X5 matrix in which the first two rows are 0s and the next two rows are 1s.



```
>> 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
```

First, create a $2 \times 5$ matrix with 0s.

Add rows 3 and 4 with 1s.

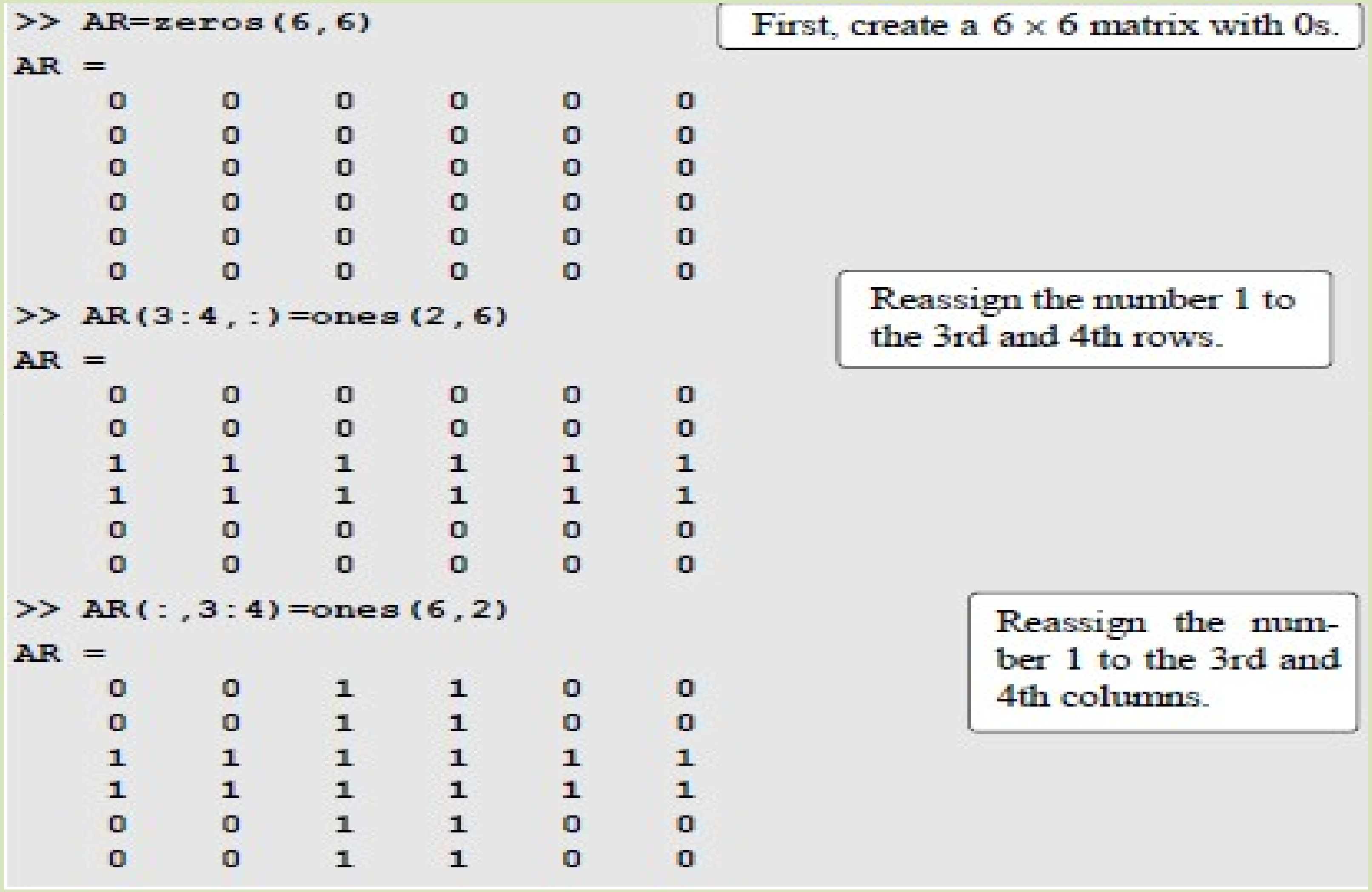


```
>> A=[zeros(2,5);ones(2,5)]

A =
     0     0     0     0     0
     0     0     0     0     0
     1     1     1     1     1
     1     1     1     1     1
```

Create a $4 \times 5$ matrix from two $2 \times 5$ matrices.

**Create a 6x6 matrix in which the middle two rows and the middle two columns are 1s, and the rest of the entries are 0s.**

Computer Programming EEN170

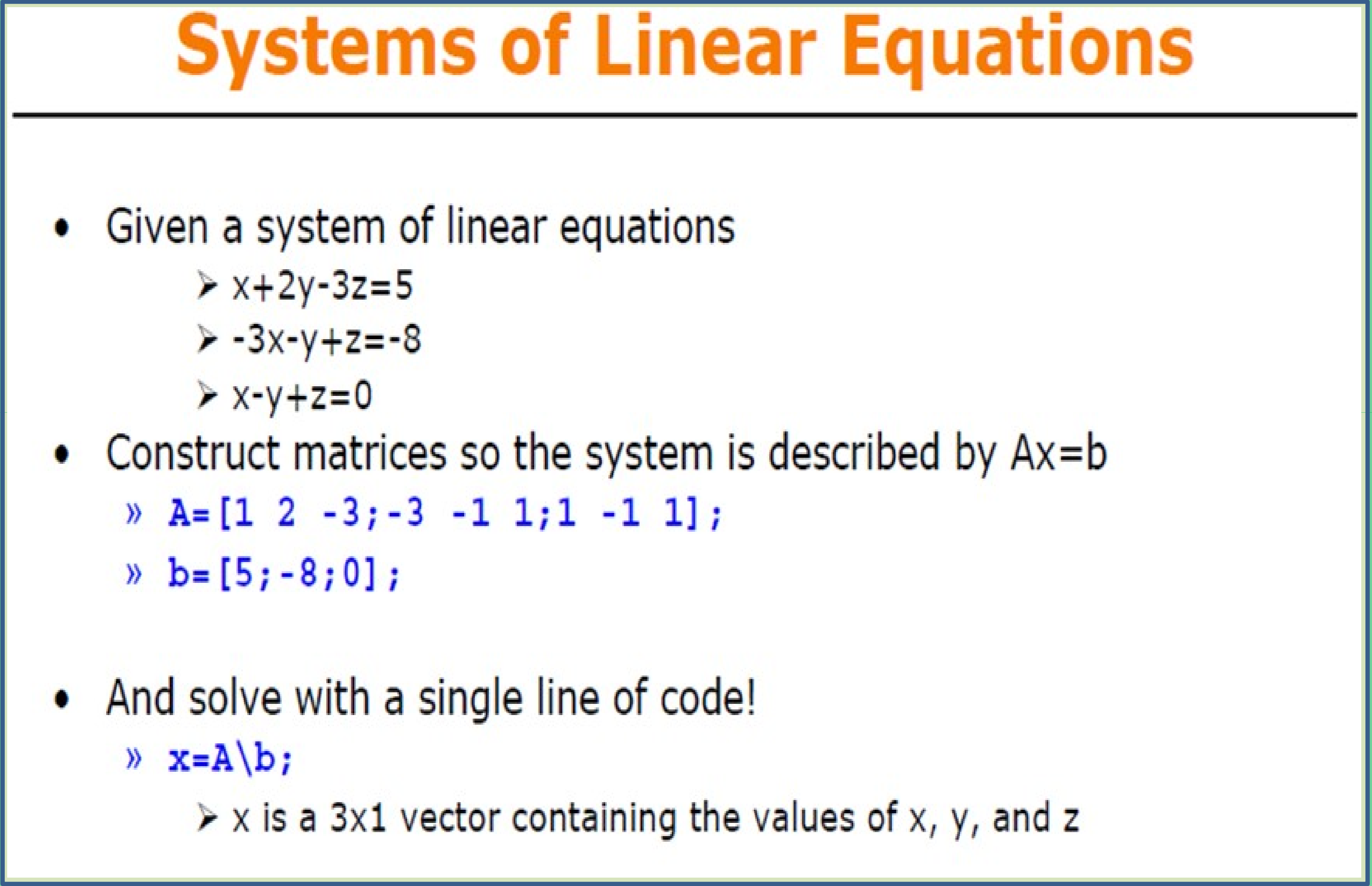


```
>> AR=zeros(6,6)

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
```

First, create a 6 × 6 matrix with 0s.

Reassign the number 1 to the 3rd and 4th rows.

Reassign the number 1 to the 3rd and 4th columns.



## Systems of Linear Equations

- Given a system of linear equations
  - x+2y-3z=5
  - -3x-y+z=-8
  - x-y+z=0
- Construct matrices so the system is described by Ax=b

```
» A=[1 2 -3;-3 -1 1;1 -1 1];
» b=[5;-8;0];
```

- And solve with a single line of code!

```
» x=A\b;
```

  - x is a 3x1 vector containing the values of x, y, and z

## Exercise: Linear Algebra

- Solve the following systems of equations:
  - System 1:

$$\begin{aligned} x+4y&=34 \\ -3x+y&=2 \end{aligned}$$

  - System 2:

$$\begin{aligned} 2x-2y&=4 \\ -x+y&=3 \\ 3x+4y&=2 \end{aligned}$$

## Exercise: Linear Algebra

- Solve the following systems of equations:
  - System 1:

$$x + 4y = 34$$
$$-3x + y = 2$$

```
» A=[1 4;-3 1];
» b=[34;2];
» x=inv(A)*b
```

  - System 2:

$$2x - 2y = 4$$
$$-x + y = 3$$
$$3x + 4y = 2$$

```
» A=[2 -2;-1 1;3 4];
» b=[4;3;2];
» x1=A\b
```