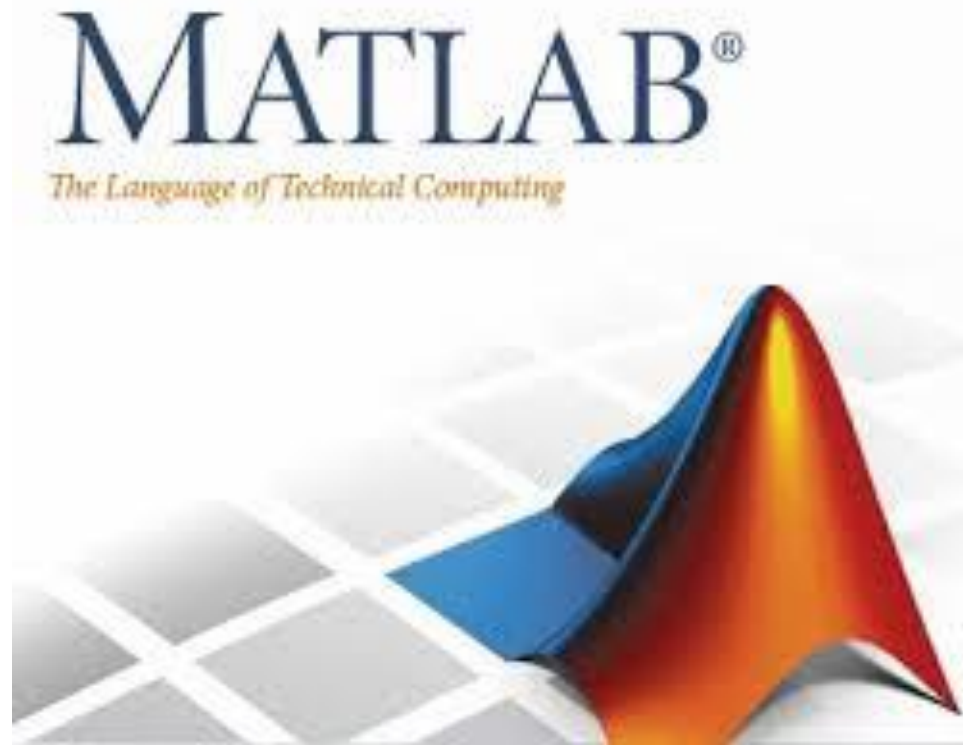




Computer Aided Design (CAD)


Lecture 4

- Conditional statements in Matlab
- Loop statements in Matlab



Reference:

Matlab by Example: Programming Basics, Munther Gdeisat



Chapter 6: Conditional Statements in Matlab

The Construction of an if Statement

Syntax

```
if (expression)
    commands are evaluated if expression is true
end
```

Example 2

Find the value of r in the program

```
x = 1
r = 1
if(x > 0)
    r = 2
end
```

$r = 2$

Example 3

Find the values of r and b in the program

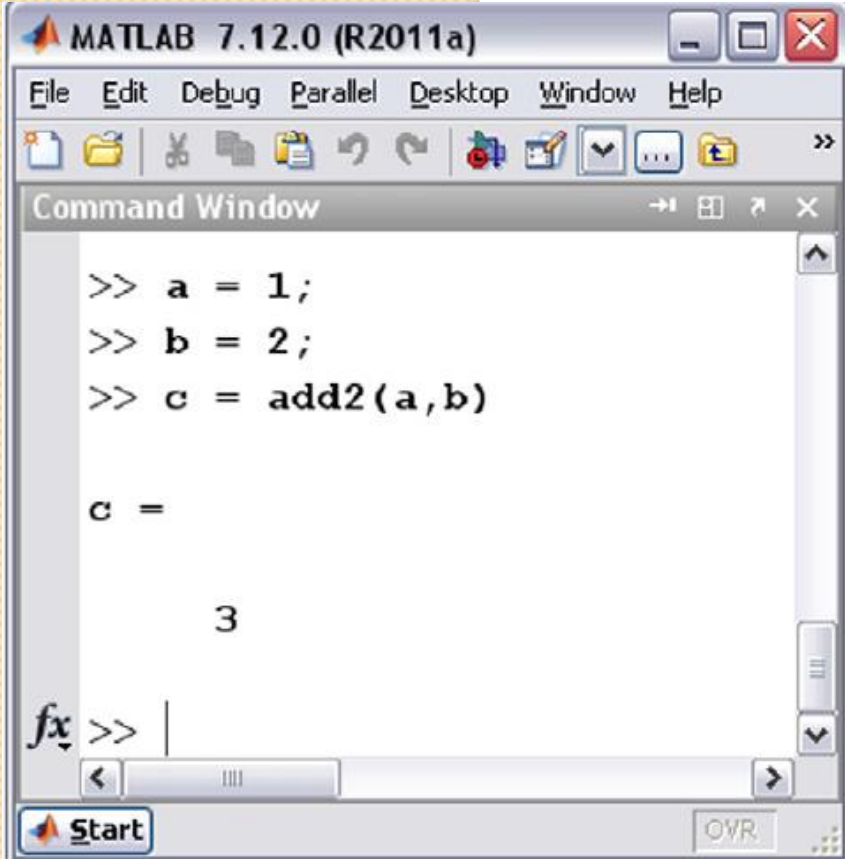
```
x=1;
y=-4;
r=1;
b=0;
if (x>0 & y<3)
    r=3;
    b=b-1;
end
```

$r = 3, b = -1$

Example:

- Write a function that has two arguments and returns a value which is equal to the addition of both arguments.
- This function checks whether both arguments are scalars:
 - If both arguments are scalars, the function performs the addition;
 - Otherwise, it displays a warning message and returns without attempting to perform the addition.

```
function z = add2(x,y)
%This function adds "x" to "y" and returns their addition
%This example shows how to use this function
%a = 1;
%b = 2;
%c = add2(a,b);
%This function returns "3" which is a result of adding "1" and "2"
%This function was written by Dr. Munther Gdeisat on 25/10/2011
%ensure that the both input arguments are scalars (numbers)
if(~isscalar(x) || ~isscalar(y))
    disp('both input arguments must be scalars (numbers)')
    return
end
z = x + y;
end
```



MATLAB 7.12.0 (R2011a)

```
File Edit Debug Parallel Desktop Window Help
```

Command Window

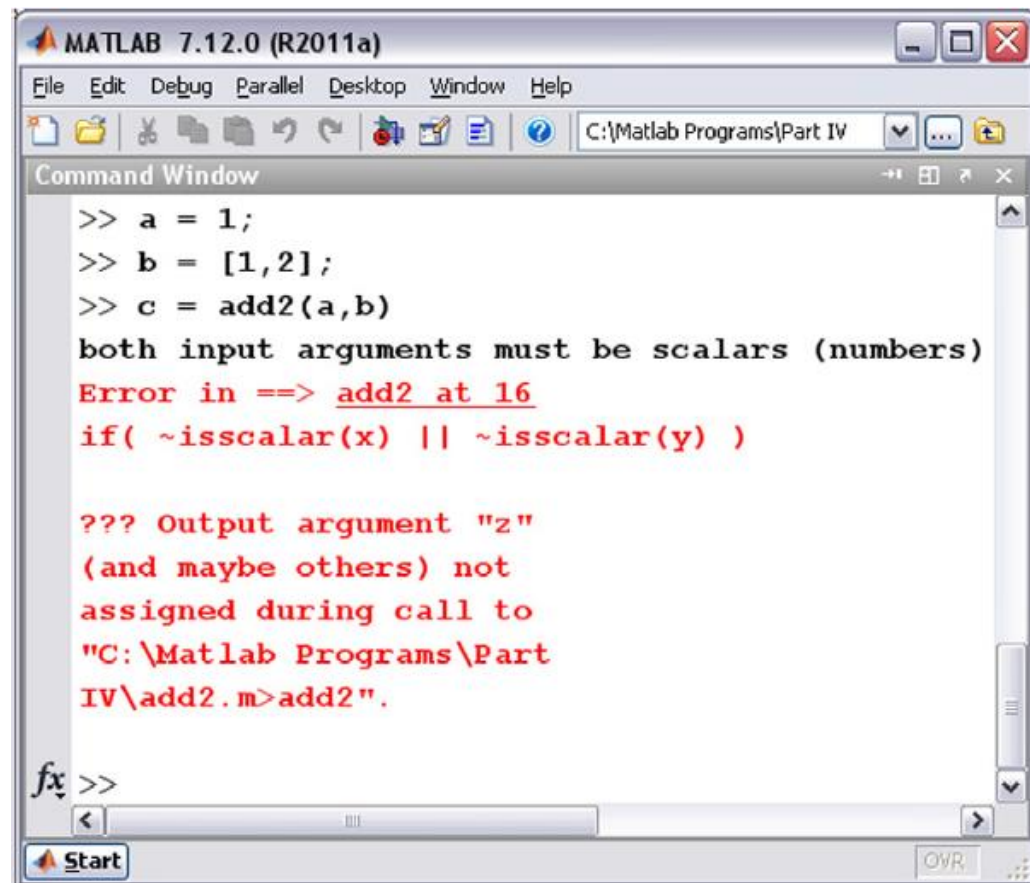
```
>> a = 1;
>> b = 2;
>> c = add2(a,b)

c =

     3
```

fx >> |

Start OVR



MATLAB 7.12.0 (R2011a)

```
File Edit Debug Parallel Desktop Window Help
```

C:\Matlab Programs\Part IV

Command Window

```
>> a = 1;
>> b = [1,2];
>> c = add2(a,b)
both input arguments must be scalars (numbers)
Error in ==> add2 at 16
if( ~isscalar(x) || ~isscalar(y) )

??? Output argument "z"
(and maybe others) not
assigned during call to
"C:\Matlab Programs\Part
IV\add2.m">add2".
```

fx >>

Start OVR

The Construction of an if else Statement

Syntax

```
if (expression)
    commands are evaluated if expression is true
else
    commands are evaluated if expression is false
end
```

Example 1

Find the value of r in the program

```
x = 1;
r = 1;
if (x > 0)
    r = 2;
else
    r = 3;
end
```

$r = 2$

Example 2

Find the value of r in the program

```
x = 1;
r = 1;
if (x > 3)
    r = 2;
else
    r = 3;
end
```

$r = 3$

Example 4

Find the value of r in the program

```
a = 1;  
b = 2;  
r = 0;  
if (a > 0)  
    if (b > 10)  
        r = 1;  
    end  
    r = 2;  
else  
    r = 3;  
end
```

$r = 2$

Example 5

Find the value of r in the program

```
a = 1;  
b = 2;  
r = 0;  
if (a > 0)  
    if (b > 10)  
        r = 1;  
    else  
        r = 2;  
    end  
    r = 3;  
end
```

$r = 3$

The Construction of an if elseif else Statement

Syntax

```
if (expression 1)
    commands are evaluated if expression 1 is true. Then jump to end.
elseif (expression 2)
    commands are evaluated if expression 2 is true. Then jump to end.
    :
elseif (expression n)
    commands are evaluated if expression n is true. Then jump to end.
else
    commands are evaluated if all the expressions 1, 2,...n are false.
end
```

Example 1

Find the value of r in the program

```
x = 1;  
y = 3;  
r = 1;  
if (x > 0)  
    r = 2;  
elseif (y < -2)  
    r = 3;  
else  
    r = 4;  
end
```

$r = 2$

Example 2

Find the value of r in the program

```
x = 1;  
y = -3;  
r = 1;  
if (x > 0)  
    r = 2;  
elseif (y < -2)  
    r = 3;  
else  
    r = 4;  
end
```

$r = 2$

Note that even though condition $y < -2$ is true, the command $r=3$ is not executed

The Construction of an switch case Statement

Syntax

```
switch (expression)
  case {expression 1}
    commands 1 are evaluated if expression 1 is true. Then jump to end.
  case {expression 2}
    commands 2 are evaluated if expression 2 is true. Then jump to end.
    :
    :
    :
  case {expression n}
    commands n are evaluated if expression n is true. Then jump to end.
  otherwise
    commands are evaluated if all the expressions 1, 2,...n are false.
end
```

The term “expression” here can be either a scalar or a string character.

Example 1

Write a Matlab program to convert a distance with units of either kilometers, meters, centimeters, or millimeters into meters.

Suppose that we would like to convert 10 cm to meters.

```
x = 10;  
units = 'cm';
```

```
switch (units)  
    case {'km'}  
        y = 1000 * x  
    case {'m'}  
        y = x  
    case {'cm'}  
        y = x / 100  
    case {'mm'}  
        y = x / 1000  
    otherwise  
        disp(['Unknown Units: ', units])  
end
```

$y = 0.1$

Example 2

The code in Example 1 can be modified as follows

Answer

```
x = 10;
units = 'cm';
switch (units)
    case {'km', 'kilometer'}
        y = 1000 * x;
    case {'m', 'meter'}
        y = x;
    case {'cm', 'centimeter'}
        y = x / 100;
    case {'mm', 'millimeter'}
        y = x / 1000;
    otherwise
        disp(['Unknown Units: ', units])
end
y
```

The expression `case {'km', 'kilometer'}` means that the command `y=1000*x;` is **evaluated** when `units` variable is equal to **either** 'km' **or** 'kilometer'.

Chapter 7: Loop Statements in Matlab

The Construction of a for Loop Statement

- The **for** keyword is used to run a piece of code for a **specific number of times**.

Syntax

```
for iteration Variable = initial value: increment: final value  
    commands  
end
```

$$\text{The number of iterations for the for loop} = \left\lfloor \frac{\text{final value} - \text{initial value}}{\text{increment}} \right\rfloor + 1$$

Where $\lfloor \cdot \rfloor$ approximates down a real number toward the nearest lower integer

- The loop stops executing the commands when the value of the **iteration Variable** is **greater than** the **final value**.

Example 3

In the following program

```
f(1) = 0;  
f(2) = 1;  
for iNo = 3:1:7  
    f(iNo) = f(iNo-1) + f(iNo-2);  
end
```

Example 4

In the following program

```
f(1) = 0;  
f(2) = 1;  
for iNo = 3:7  
    f(iNo) = f(iNo-1) + f(iNo-2);  
end
```

The default increment for the iteration variable is 1.

the iteration variable is `iNo`.

The initial value for the iteration variable is 3.

The increment for the iteration variable is 1.

The final value for the iteration variable is 7.

The command in the `for` statement is `f(iNo) = f(iNo-1) + f(iNo-2);`

The number of iterations for the `for` loop = $\lfloor \frac{7-3}{1} \rfloor + 1 = 5$.

The values of `iNo` are 3, 4, 5, 6, and 7.

```
>> f  
  
f =  
  
    0    1    1    2    3    5    8
```


Example 5

In the following program

```
x = 1;  
for ix = 0:2:10  
    x = 0.9 * x;  
end
```

The iteration variable is ix .

The initial value for the iteration variable is 0.

The increment for the iteration variable is 2.

The final value for the iteration variable is 10.

The command in the `for` statement is $x = 0.9 * x$;

The number of iterations for the `for` loop = $\lfloor \frac{10-0}{2} \rfloor + 1 = 6$.

The values of ix are 0, 2, 4, 6, 8, and 10.

Example 6

In the following program

```
x = 1;  
for ix = 1:2:10  
    x = 0.9 * x;  
end
```

The iteration variable is ix .

The initial value for the iteration variable is 1.

The increment for the iteration variable is 2.

The final value for the iteration variable is 10.

The command in the `for` statement is $x = 0.9 * x$;

The number of iterations for the `for` loop $= \left\lfloor \frac{10-1}{2} \right\rfloor + 1 = 4 + 1 = 5$.

The values of ix are 1, 3, 5, 7, and 9.

Example 7

In the following program

```
x = 1;  
for ix = 1:2:1  
    x = 0.9 * x;  
end
```

The iteration variable is `ix`.

The initial value for the iteration variable is 1.

The increment for the iteration variable is 2.

The final value for the iteration variable is 1.

The command in the `for` statement is `x = 0.9 * x;`

The number of iterations for the `for` loop = $\left\lfloor \frac{1-1}{2} \right\rfloor + 1 = 0 + 1 = 1$.

The value of `ix` is 1.

Example 8

In the following program

```
x = 1;  
for ix = 1:2:-3  
    x = 0.9 * x  
end
```

The iteration variable is ix .

The initial value for the iteration variable is 1.

The increment for the iteration variable is 2.

The final value for the iteration variable is -3 .

The command in the `for` statement is $x = 0.9 * x$;

The number of iterations for the `for` loop $= \left\lfloor \frac{-3-1}{2} \right\rfloor + 1 = -2 + 1 = -1$.

- Matlab will not execute the commands inside the loop since the number of iterations is negative. The value of ix is $[]$ i.e., empty matrix

Example 9

In the following program

```
x = 1;  
for ix = 6:-2:-6  
    x = 0.9 * x  
end
```

The initial value for the iteration variable is 6.

The increment for the iteration variable is -2 .

The final value for the iteration variable is -6 .

The command in the `for` statement is `x = 0.9 * x;`

The number of iterations of a for loop = $\left\lfloor \frac{-6-6}{-2} \right\rfloor + 1 = 6 + 1 = 7$

The values of iteration variable are 6, 4, 2, 0, -2, -4, -6

Example 10

Find the value of x produced by this program.

```
x = 1;  
for ix = 1:2:10  
    x = 0.9 * x*ix  
end
```

| Iteration | ix | x |
|------------------|-----------|--|
| First | 1 | $0.9 \times 1 \times 1 = 0.9$ |
| Second | 3 | $0.9 \times 0.9 \times 3 = 2.43$ |
| Third | 5 | $0.9 \times 2.43 \times 5 = 10.9350$ |
| Fourth | 7 | $0.9 \times 10.9350 \times 7 = 68.8905$ |
| Fifth | 9 | $0.9 \times 68.8905 \times 9 = 558.0131$ |

You can follow the execution of the program on a step-by-step basis using the debug tools available in Matlab to check the results for each iteration

Example 12

Write a Matlab program to calculate the factorial of an integer number n . The factorial of a number n is defined as

$$n! = n \times (n - 1) \times (n - 2) \times (n - 3) \dots \dots \dots (3) \times (2) \times (1)$$

- The factorial of the number 1 is 1 by definition.
- The factorial of 2 is $2 \times 1 = 2$.
- The factorial of 3 is $3 \times 2 \times 1 = 6$ and so on.
- The factorial 4 is $4 \times 3 \times 2 \times 1 = 24$ and so on.

Answer

This program calculates the factorial of the number $n = 4$.

```
%This program calculates the factorial of n
n = 4;
%calculate the factorial of n
factorial_of_n = 1;
for in = 2:n
    factorial_of_n = factorial_of_n * in;
end
```

| Iteration | in | factorial_of_n |
|-----------|----|-------------------|
| First | 2 | $1 \times 2 = 2$ |
| Second | 3 | $2 \times 3 = 6$ |
| Third | 4 | $6 \times 4 = 24$ |

The Construction of a Nested for Statement

Syntax

```
for iteration Variable1 = initial value1: increment1: final value1
    for iteration Variable2 = initial value2: increment2: final value2
        commands
    end
end
end
```

Number of iterations of outer loop $N1 = \left\lfloor \frac{\text{final value1} - \text{initial value1}}{\text{increment1}} \right\rfloor + 1$

Number of iterations of inner loop $N2 = \left\lfloor \frac{\text{final value2} - \text{initial value2}}{\text{increment2}} \right\rfloor + 1$

Number of total iterations of nested loop $N = N1 \times N2$

Example 19

Write a Matlab program to calculate the summation of an array. Your program should be able to calculate the summation of the array irrespective of its dimensions. The summation of the array is given by the following equation:

$$\text{summation} = \sum_{i=1}^m \sum_{j=1}^n V(i,j)$$

m: Number of rows
n: Number of columns

Answer of Example 19

The following Matlab program calculates the summation of an array:

```
V = [1, 3, 6; 4, 9 10];  
m = size(V,1);  
n = size(V,2);  
summation = 0;  
for i = 1:m  
    for j = 1:n  
        summation = summation + V(i,j);  
    end  
end
```

- It is just an explanatory example on nested for loop
- Actually, we don't have to use for loops to sum elements of array.
- We can use the following command directly: `sum(sum(V))`

Try the two methods and check the time

- Nested for loop is useful in many other applications

The Construction of Combined for and if Statement

Example 3

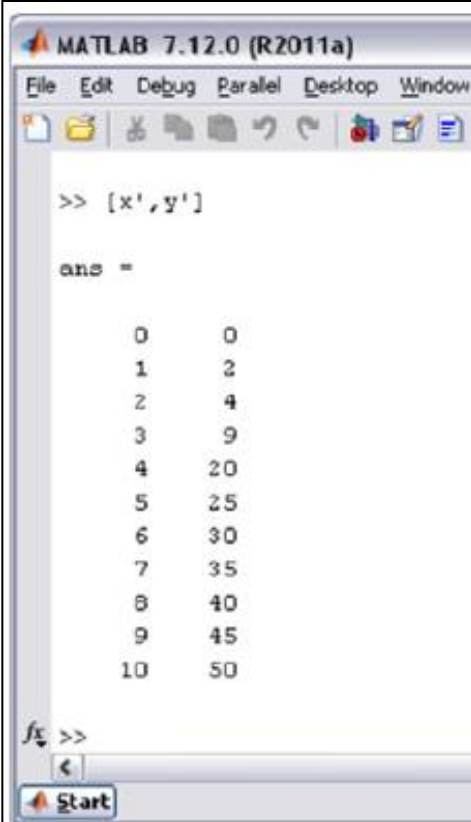
Write a Matlab program to calculate $y(x)$ according to the equation

$$y(x) = \begin{cases} 2x & x < 3 \\ 3x & x = 3 \\ 5x & x > 3 \end{cases}$$

where x is a vector and it is given as $x = 0:1:10$;

Answer

```
x = 0:1:10;
N = length(x);
for k = 1:N
    if (x(k) < 3)
        y(k) = 2*x(k);
    elseif (x(k) == 3)
        y(k) = 3*x(k);
    else
        y(k) = 5*x(k);
    end
end
```



```
MATLAB 7.12.0 (R2011a)
File Edit Debug Parallel Desktop Window

>> [x',y']

ans =

     0     0
     1     2
     2     4
     3     9
     4    20
     5    25
     6    30
     7    35
     8    40
     9    45
    10    50

fx >>
<
Start
```

The continue Keyword

- The continue keyword **passes control** to the **next iteration** of the for loop in which it appears, skipping any remaining statements in the body of the for loop.

Example 5

Write a Matlab program to calculate y according to the equation

$$y = \sum_{k=-10}^{10} \frac{1}{k^2 + 2k} \quad \text{where } k \text{ is an integer and } k \neq 0, -2$$

Answer

```
y = 0;
for k = -10:10;
    if k == 0 || k == -2
        continue
    end
    y = y + 1 / (k^2 + 2*k);
end
```

The break Keyword

- The break keyword **terminates the execution** of for loop.
- In **nested loops**, break exits from the **innermost** loop.

Example 7

Write a Matlab program to produce the numbers of a Fibonacci series whose values are less than 100.

$$F_n = F_{n-1} + F_{n-2}$$

The first seven numbers of a Fibonacci series are

0, 1, 1, 2, 3, 5, 8

Answer

```
f(1) = 0;  
f(2) = 1;  
for i = 3:10000  
    c = f(i-1) + f(i-2);  
    if c >= 100  
        break  
    end  
    f(i) = c;  
end
```

The Construction of while Loop

- Matlab executes the commands inside the while loop **as long as statement remains true.**

Syntax

```
while (statement)
    commands
end
```

Example 1

Find the value of r that is produced by the following program:

```
r = 2;
while (r < 10)
    r = 2*r;
end
```

| Iteration | r |
|-----------|-----|
| 1 | 4 |
| 2 | 8 |
| 3 | 16 |

- With $r = 16$ (on the third iteration), the statement ($r < 10$) is now false.
- Matlab therefore terminates the while loop.
- The command $r = 2*r$ in the loop body is therefore not evaluated and the final value of r remains at 16.

Example 2

Find the value of r that is produced by the following program:

```
r = 2;  
while (r < 10)  
    if (r == 8)  
        r = r - 1  
        continue  
    end  
    r = 2*r  
end
```

| Iteration | r |
|-----------|----|
| 1 | 4 |
| 2 | 8 |
| 3 | 7 |
| 4 | 14 |

- When the statement $(r == 8)$ is true, the two commands in the body of the if statement are now evaluated.
- The command $r = r - 1$ is therefore evaluated. The new value of r is 7.
- Matlab then executes the **continue** command, passes control to the next iteration of the while loop, and skips the remaining commands in the body of the while loop.

Example 3

Find the value of r that is produced by the following program:

```
r = 2;
while (r < 10)
    if (r == 8)
        r = r - 1;
        break;
    end
    r = 2*r;
end
```

| Iteration | r |
|-----------|---|
| 1 | 4 |
| 2 | 8 |
| 3 | 7 |

- When the statement $(r == 8)$ is now true, the two commands in the body of the if statement are evaluated.
- The command $r = r - 1$ is therefore evaluated, The new value of r is 7.
- Matlab then executes the **break command**, and terminates the execution of while loop.

Example 4

Write a Matlab program to produce the numbers of a Fibonacci series whose values are less than 100.

Answer

```
f(1) = 0;  
f(2) = 1;  
counter = 3;  
while( ( f(counter-1) + f(counter-2) ) < 100)  
    f(counter) = f(counter-1) + f(counter-2);  
    counter = counter + 1;  
end
```

Final results after executing the loop:

f =

0 1 1 2 3 5 8 13 21 34 55 89

Numbers

1) Fixed point notation:

e.g 1.2345, -100.5243

2) Scientific (floating point) notation:

e.g 1.2345×10^9 can be expressed as $1.2345e + 9$



mantissa

exponent

Examples

1.234×10^5 , -8.765×10^{-4} , 10^{-15} , -10^{12} ,
($1.234e + 05$, $-8.765e-04$, $1e-15$, $-1e + 12$).



Thanks for attention