THE UNIVERSITY OF RHODE ISLAND

EGR 106
Foundations of Engineering II

Lecture 4 – Part B

Logical Arrays, Relational Operators, Conditional Statements and Loops

THINK BIG WE DO℠

This Week's Topics

Programming in MATLAB

- More on scripts
- Input / output
- Logical Arrays
- Relational Operators
- Conditional Statements
- "for-end" loops

Recall - Useful Script Commands

clc – clears the command window

clear – clears variables from memory

close all – closes all figure windows

Good to use at beginning of code

% - creates comment, everything to the right of % is ignored

pause – stops operation and waits for a key press

pause(n) – stops operation and waits for n seconds

Other "Tips" for MATLAB Scripts

To continue a long line of code use three periods at end (…), hit enter and continue typing the remainder of text on next line

```
% example of a long line of code
clc; clear all;
a=[1 2 3 4 5 6 7 8 ...
    9 10 11 12 13 14 ...
    15 16 17 18 19 20]
```

```
a =

     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20
```

Typing "Ctrl-c" in command window will interrupt the current computation

Input – Allowing Users to Enter Data

Using the “input” command:

Numeric:

```
x = input(‘Please enter a number: ')
```

String:

```
x = input(' Please enter a string of characters: ', 's')
```

Output - How scripts show or output data:

Simply type array name – will show in Command Window

Use of `disp` command to display results:

- Existing array (a single array only – if necessary, use concatenation)

  disp(x) or disp([x,y])

- Text

  disp(' The task is done ')

- Concatenate text strings with numerical results using 'num2str'

  (see last week's lecture)

Formatted output - use of `fprintf` command (we'll skip for now)

Example – `disp` command

```
>> x = rand(3,2);
>> x

x =

    0.9501    0.4860
    0.2311    0.8913
    0.6068    0.7621

>> disp(x)
    0.9501    0.4860
    0.2311    0.8913
    0.6068    0.7621
```

Note that disp shortens the resulting output by dropping the array name and removing blank lines

Example – Input/Output (I/O)

Editor - C:\Users\taggart\Documents\egr106\Week_5_demo_1.m

Week_5_demo_1.m

```
% sample script showing I/O using 'input' and 'disp' commands
%
clc; clear
%
% ask user for input:
%
base=input('How long is the base? ');
height=input('How high is the triangle? ');
%
% compute area
%
area=0.5*base*height;
%
% display result using just the disp command
%
disp('The area is ');
disp(area)
%
% display result using 'num2str' command
%
disp(['The area is ' num2str(area)]);
```

input

disp

num2str (see last week's lecture)

Command Window Output

```
How long is the base? 3
How high is the triangle? 4
The area is
     6
The area is 6
>>
```

Programming in MATLAB

So far, we've looked at simple programming using scripts of sequentially evaluated commands

In more advanced programs, commands are not always executed in order. Examples:

- Decision-making within the program controls program flow
- Repeated sequence of commands (loops)
- Calls to separate subprograms (user-defined functions)

Logical Variables / Arrays

Matlab variable types:

- numbers
- character strings
- logical (0=false, 1=true)

Command Window

```
>> a=10
a =
    10
>> b=['hello  ';'goodbye']
b =
hello
goodbye
>> c=true
c =
     1
>> d=false
d =
     0
>> e=[false true true;true false true]
e =
     0     1     1
     1     0     1
>> whos
  Name      Size            Bytes  Class

  a         1x1                 8  double
  b         2x7                28  char
  c         1x1                 1  logical
  d         1x1                 1  logical
  e         2x3                 6  logical
```

Relational Operators

Compares two numbers or arrays and decides if relation is true (1) or false (0).

Relational Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Relational Operators (cont.)

Examples:

```
>> 3<4
ans =
     1
>> 3>4
ans =
     0
>> 3==4
ans =
     0
>> 3==3
ans =
     1
>> 3~=4
ans =
     1
```

Is 3 less than 4 ? - true

Is 3 greater than 4 ? - false

Is 3 equal to 4 ? - false

Is 3 equal to 3 ? - true

Is 3 not equal to 4 ? - true

Relational Operators – Applied to Arrays

For arrays, relational operations are applied term-by-term

Examples:

```
>> a=[3 2 6 8]
a =
     3     2     6     8
>> b=[5 2 4 8]
b =
     5     2     4     8
>> a==b
ans =
     0     1     0     1
>> a>b
ans =
     0     0     1     0
>> a~=b
ans =
     1     0     1     0
```

Relational Operators – More Examples

Combining math and relational operations:

```
>> b=(6<10)+(7<8)+(5*3==60/4)
b =
     3
```

How many s's in Mississippi?

```
>> a='Mississippi'
a =
Mississippi
>> sum(a=='s')
ans =
     4
```

Logical Operators

AND, OR, XOR and NOT:

Logical Operator	Name	Description
& Example A&B	AND	True if A and B is true
| Example A|B	OR	True if A or B is true
xor Example A xor B	XOR	Exclusive OR – true if A or B is true but not both A and B are true
~ Example: ~A	NOT	If A is true, ~A is false If A is false, ~A is true

Logical Operators - Examples

```
>> A=true
A =
     1
>> B=false
B =
     0
>> A&B
ans =
     0
>> A|B
ans =
     1
```

```
>> A=true
A =
     1
>> B=true
B =
     1
>> xor(A,B)
ans =
     0
>> B=false
B =
     0
>> xor(A,B)
ans =
     1
```

```
>> A=true
A =
     1
>> ~A
ans =
     0
>> B=false
B =
     0
>> ~B
ans =
     1
```

Other Logical Operators

Logical Operator	Description
all Example: all(A)	True if <u>all</u> elements of A are true
any Example: any(A)	True if <u>any</u> elements of A are true
find Example: find(A)	Returns the indices of nonzero (true) elements of A

Example:

```
>> A = [ 5 6 7 2 10 ];
>> find(A>5)

ans =

     2     3     5
```

Logical Operators Applied to Numbers

A and B are interpreted as logical (binary):

Numeric 0 is interpreted as false

All else is interpreted as true (equal to 1)

Example:

```
>> A=0
A =
     0
>> B=10
B =
    10
>> A&B
ans =
     0
>> A|B
ans =
     1
>> ~A
ans =
     1
>> ~B
ans =
     0
```

When logical operators are applied, A is false, B is true

Logical Operators – Examples

"Are you between 25 and 30 years old?"

(age>=25) & (age<=30)

```
>> age=23
age =
    23
>> (age>=25) & (age<=30)
ans =
     0
>> age=28
age =
    28
>> (age>=25) & (age<=30)
ans =
     1
```

Order of Precedence

1. Parentheses ( )
2. Transpose(') and exponentiation (.^)
3. Negation (-) and logical negation (~)
4. Multiplication (.*) and division (./),
5. Addition (+) and subtraction (-)
6. Relational operators (<, <=, >, >=, = =, ~=)
7. Logical AND (&)
8. Logical OR (|)

Order of Precedence - Examples

Consider:

$$3*5-6/3>3*3+\sim 0+4^2$$

$$15-2>9+1+16$$

$$13>26$$

$$0$$

```
>> 3*5-6/3>3*3+~0+4^2

ans =

     0
```

Branches and Conditional Statements

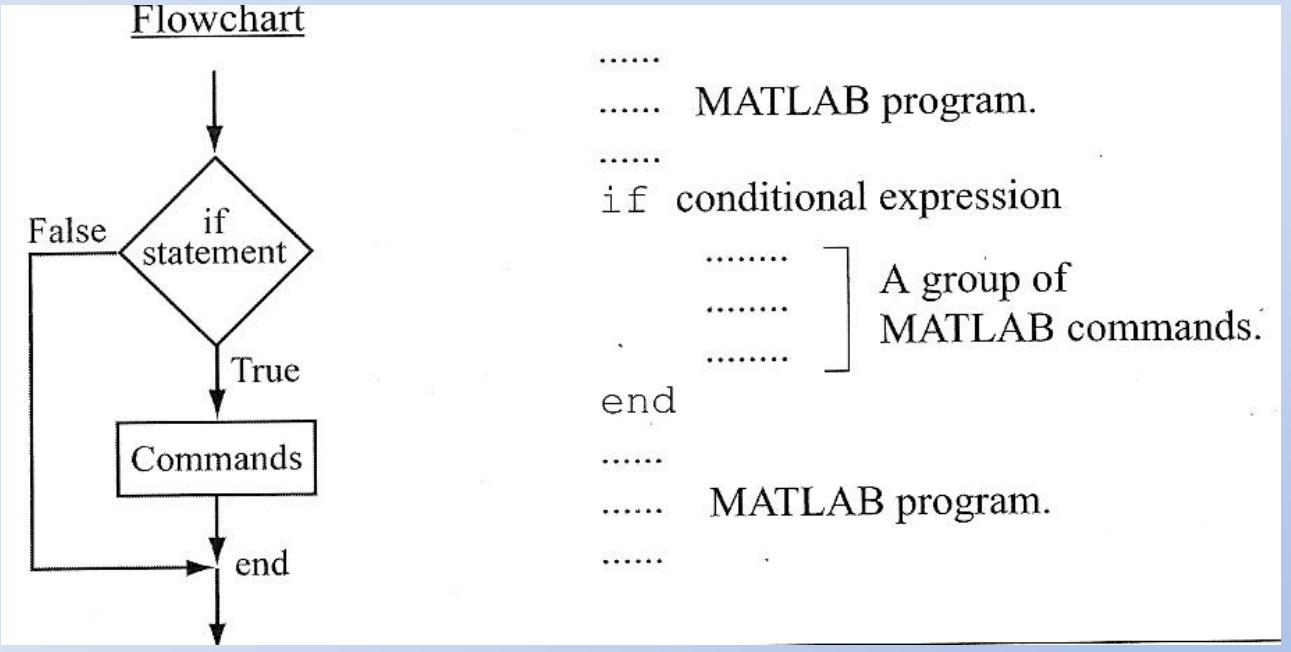
Commands to select and execute certain blocks of code, skipping other blocks.

Common types in MATLAB include:

- if / else
- switch
- try / catch

We will only discuss and use “if / else”

if – end command



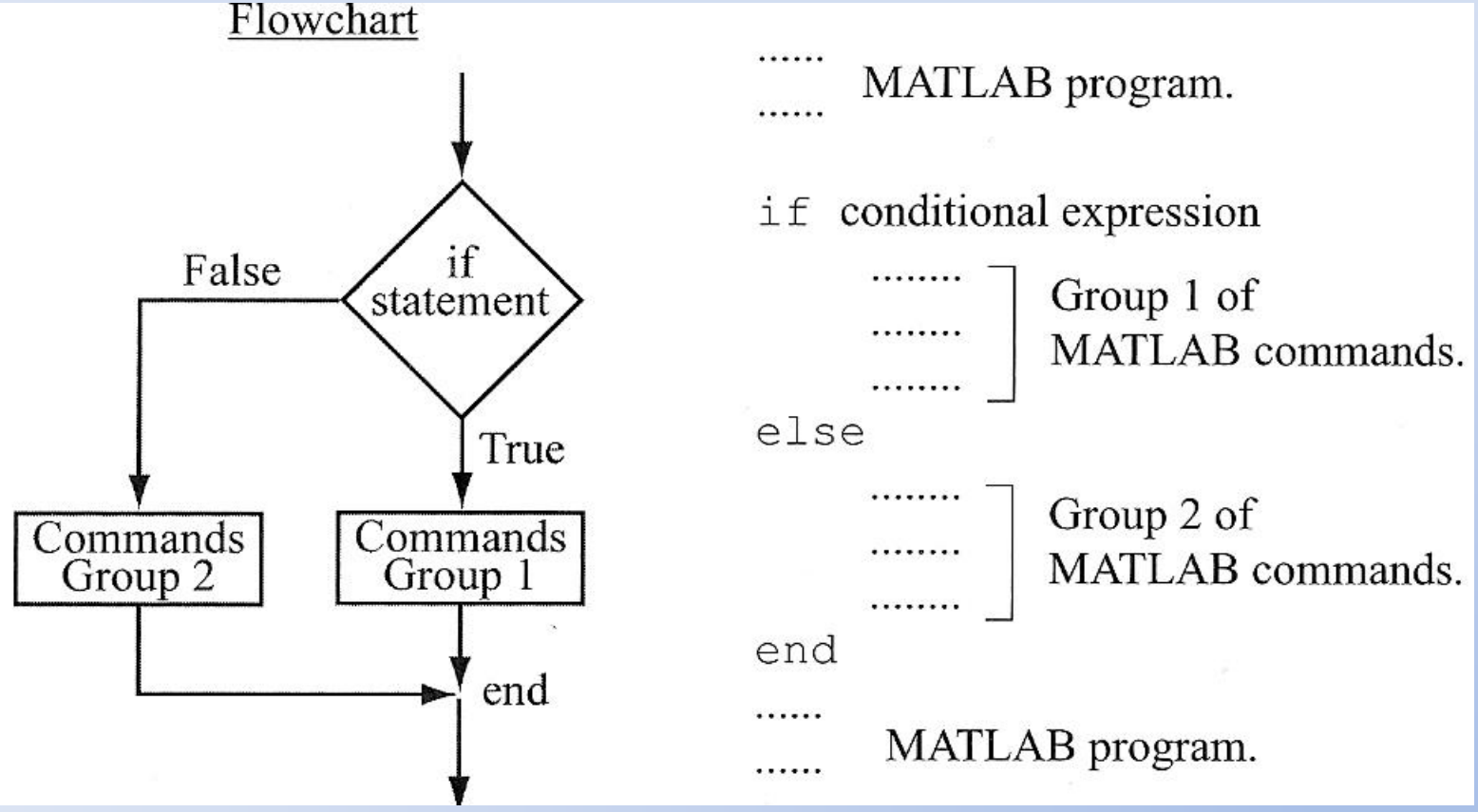
if – end (cont.)

Another example: Ask if a plot should be drawn

```
x = input('Enter 1 to plot now');
if  x==1
    plot(a,b)
end
```

in editor, blue color
& auto indent

“if – else – end” Command



“if – else – end” (cont.)

Example – output whether a variable x is positive or not:

```
% x =  { computed somehow };
%
if  x > 0
    disp('the value is positive')
else
    disp('the value is negative or zero')
end
```

“if – else – end” (cont.)

What if there are more than 2 pieces of code to make a choice on? For example:

Example - 4 choices:

convert a compass angle to a direction

0° → east

90° → north

180° → west

270° → south

“if – else – end” (cont.)

Example: Convert a compass angle to a direction:

0° → east

90° → north

180° → west

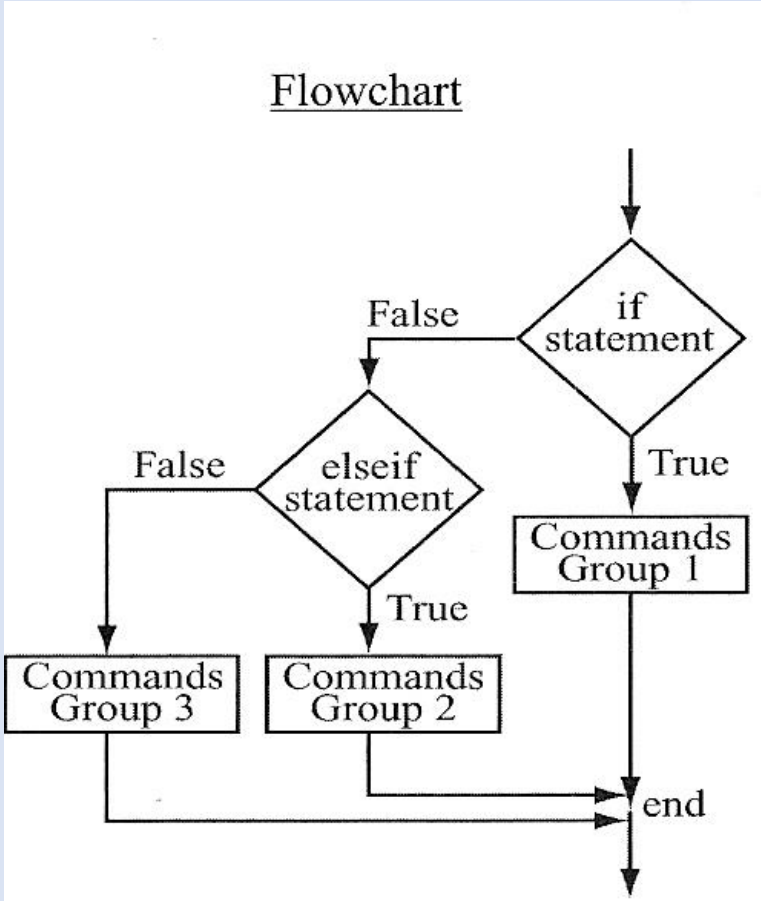
270° → south

Using “nested” if-else-end:

```
D = input('angle');
if D == 0
    disp('east');
else
    if D == 90
        disp('north');
    else
        if D == 180
            disp('west');
        else
            disp('south');
        end
    end
end
```

Better to use if-elseif-else-end

“if-elseif-else-end” Command




```
......
      MATLAB program.
......

if  conditional expression
        ........
        ........      Group 1 of
        ........      MATLAB commands.
elseif  conditional expression
        ........
        ........      Group 2 of
        ........      MATLAB commands.
else
        ........
        ........      Group 3 of
        ........      MATLAB commands.
end
......
      MATLAB program.
......
```

“if-elseif-else-end” (cont.)

Previous example

Note – many “elseifs” are allowed, but only 1 “else”

```
D = input('angle');
if D == 0
    disp('east');
elseif D == 90
    disp('north');
elseif D == 180
    disp('west');
else
    disp('south');
end
```

“if-elseif-else-end” (cont.)

Another example :

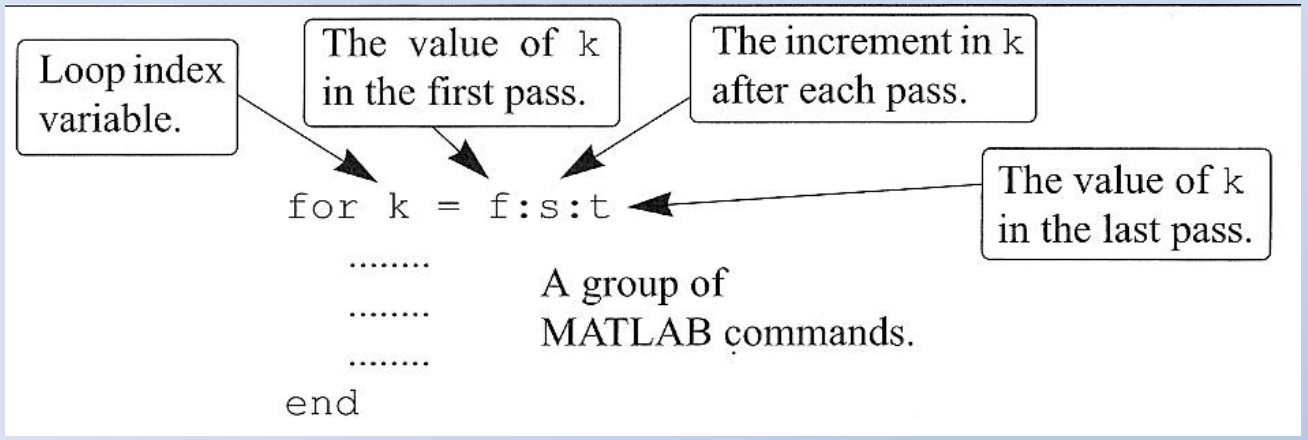
Output temperature in three different ranges:

```
%  T = ... { computed somehow };
if  T >= 100
    disp('temperature is above boiling')
elseif  (T>0) & (T<100)
    disp('temperature is moderate')
else
    disp('temperature is below freezing')
end
```

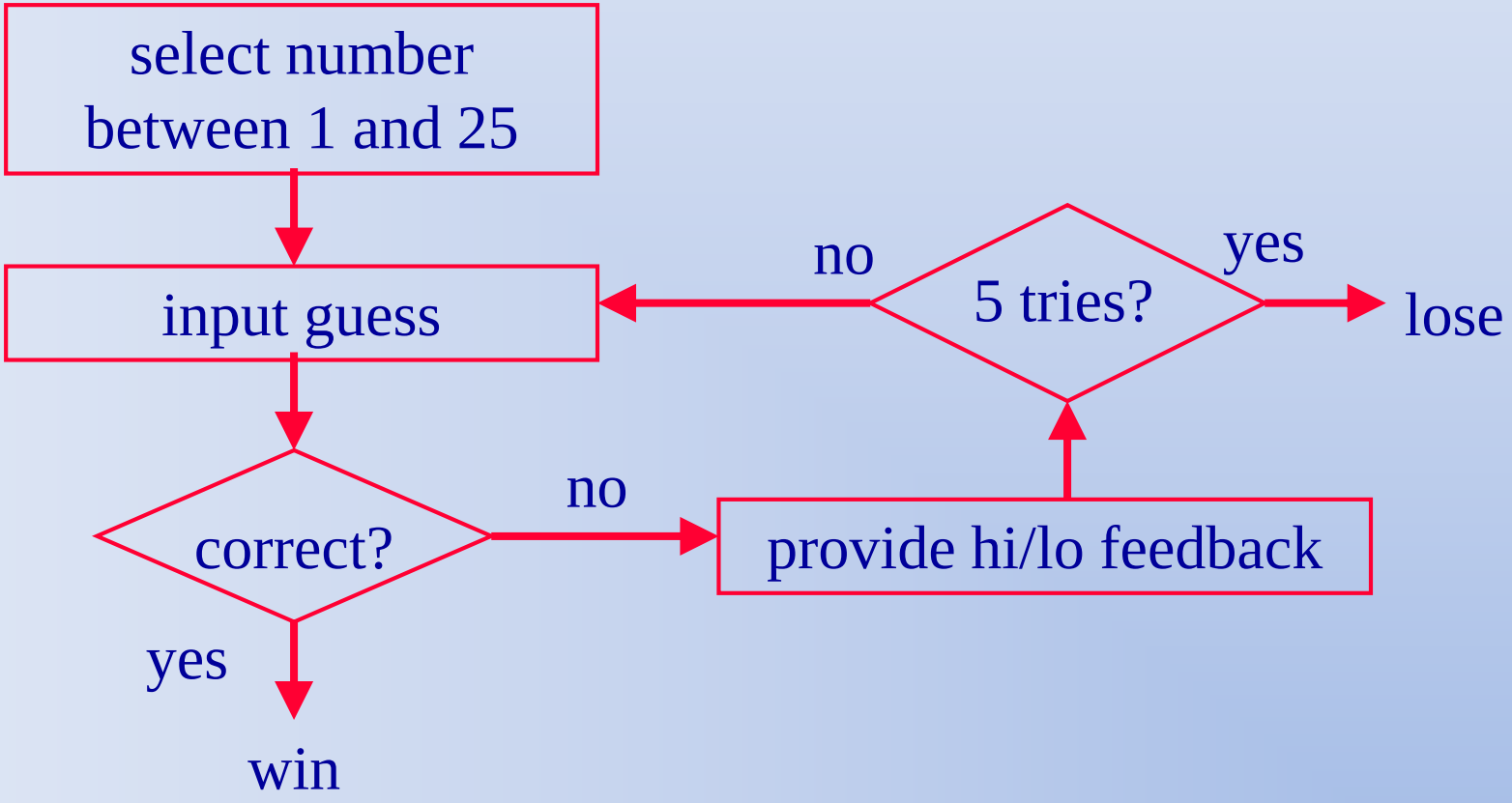
Example – Real Roots of Quadratic Equation (see Lecture_4.m – Example 1)

```
%
% Example 1 - Roots of a quadratic equation
%
%  test cases:
%   x^2+2x+3=0 => a,b,c = 1,2,3 => no real roots exist
%   x^2-4x+4=(x-2)^2=0 => a,b,c = 1,-4,4 => repeated root at x = 2
%   x^2=3x+2=(x-2)*(x-1)=0 => a,b,c = 1,-3,2 => two real roots x = 2, 1
%
disp('Compute real roots of the quadratic equation ax^2+bx+c=0');
a=input('Enter a: ');
b=input('Enter b: ');
c=input('Enter c: ');
%
discriminant=b^2-4*a*c;
if discriminant<0
    disp('no real roots exist')
elseif discriminant==0
    root=-b/(2*a);
    disp(['repeated root at x = ',num2str(root)])
else
    root=[(-b+sqrt(b^2-4*a*c))/(2*a) (-b-sqrt(b^2-4*a*c))/(2*a)];
    disp(['two real roots at x = ',num2str(root)])
end
```

The “for-end” Loop (more on this later)



Example – Hi-Lo guessing game



Example – Hi-Lo guessing game (see Lecture_4.m – Example 2)

```
clear all; clc;
numb=ceil(25*rand(1));
for count=1:5
    guess=input('Guess a number between 1 and 25: ');
    if guess==numb
        disp('Correct!')
        break
    elseif guess>numb
        disp('Too high')
    elseif guess<numb
        disp('Too low')
    end
    if count==5
        disp(['Sorry, you lose. My number was ',num2str(numb)])
    end
end
```

```
Guess a number between 1 and 25: 5
Too low
Guess a number between 1 and 25: 10
Too low
Guess a number between 1 and 25: 15
Too high
Guess a number between 1 and 25: 13
Too high
Guess a number between 1 and 25: 12
Too high
Sorry, you lose. My number was 11
```

Note: “break” will be discussed later