Introduction to Computing for Engineers 050IU

Logical Operators and Conditional Statements

Dr. Nguyen Ngoc Truong Minh

Relational Operators

Relational operator	Meaning
<	Less than.
<=	Less than or equal to.
>	Greater than.
>=	Greater then or equal to.
==	Equal to.
~=	Not equal to.

- Relational operators compare two numbers in a comparison statement.
- If the statement is **TRUE**, it is assigned a value of 1.
- If the statement is **FALSE**, it is assigned a value of 0.

Relational Operators

```
>> 5>8

ans =

     0

>> a=5<10

a =

     1

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

y =

     2
```

Since 5 is not larger than 8 the answer is 0.

Checks if 5 is smaller than 10, and assigns the answer to **a**.

Since 5 is smaller than 10 the number 1 is assigned to **a**.

=1

=0

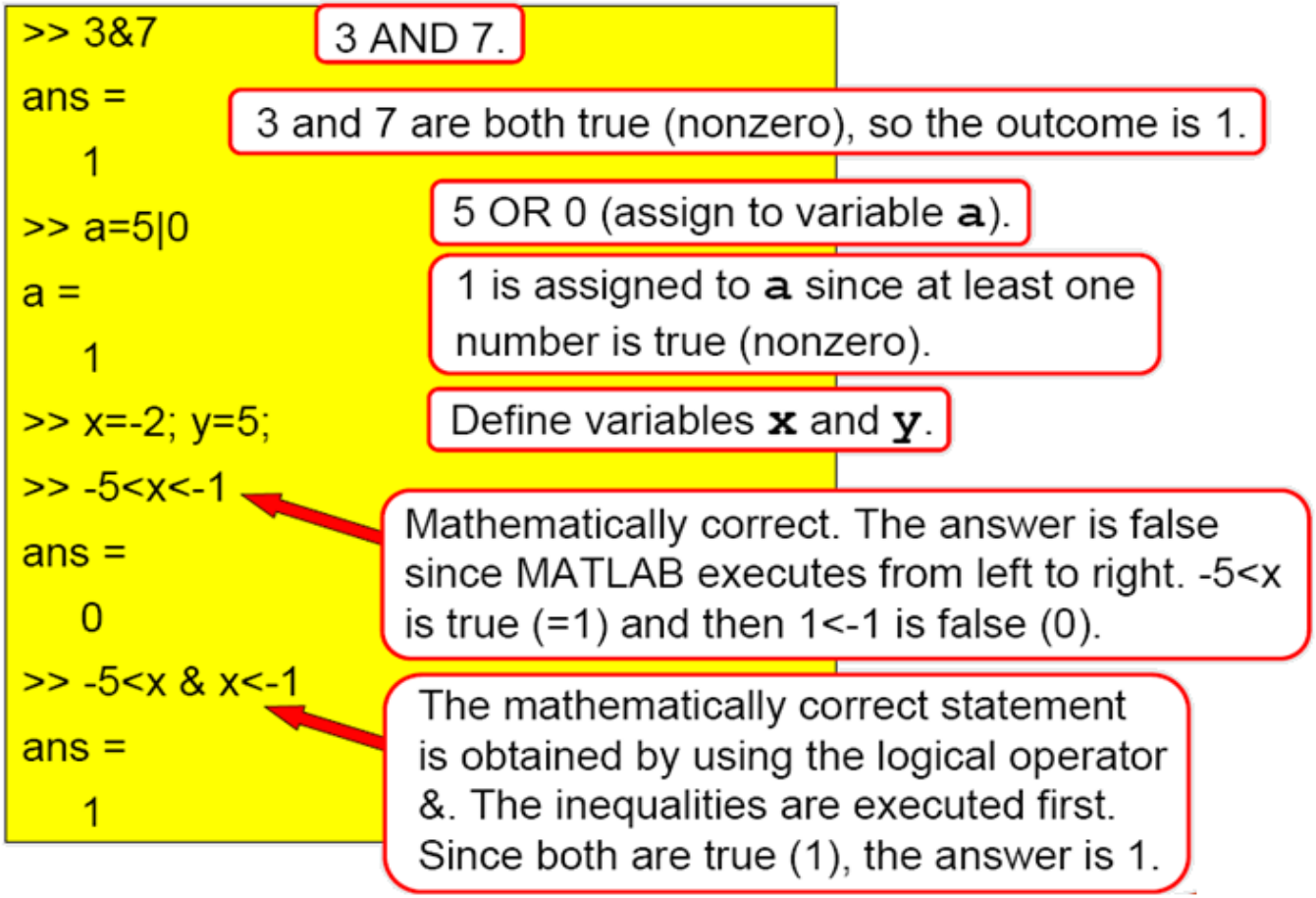
=1

Logical Operators

- Logical operators have numbers as operands.
- A nonzero number is **TRUE**.
- A zero number is **FALSE**.

Logical Operator	Name	Meaning
& Example: *A* & *B*	AND	**TRUE** if both operands (*A* and *B*) are true.
| Example: *A* | *B*	OR	**TRUE** if either or both operands (*A* and *B*) are true.
~ Example: ~ *A*	NOT	**TRUE** if the operand (*A*) is false **FALSE** if the operand (*A*) is true

Logical Operators



Conditional Statements

- Conditional statements enable MATLAB to make decisions.
- The process is similar to the way we (humans) make decisions.
- A condition stated. If the condition is met, one set of actions is taken. If the condition is not met, either nothing is done, or a second set of actions is taken.

Example:

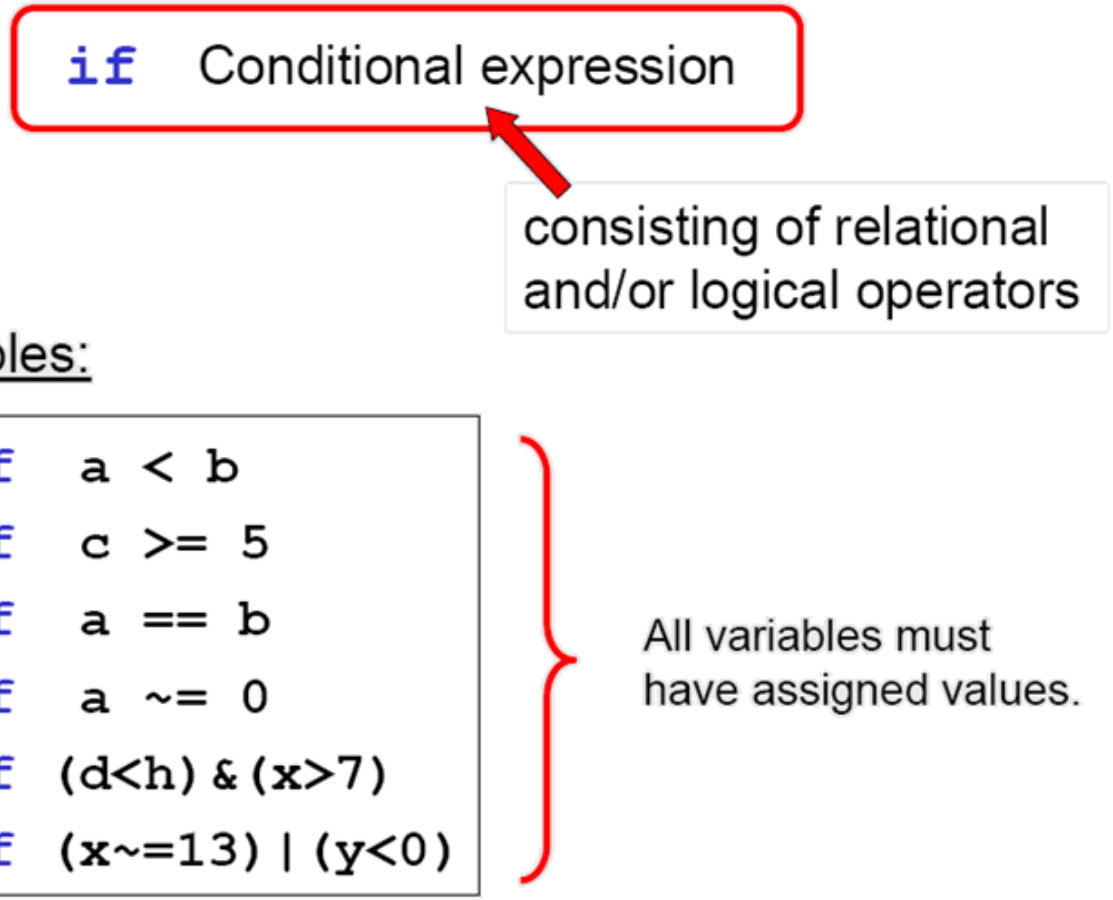
If I win the Lottery,

I will quit college, buy a new car, and go fishing.

If I do not win the Lottery,

I will study harder so that I can get a better job.

The Form of a Conditional Statement



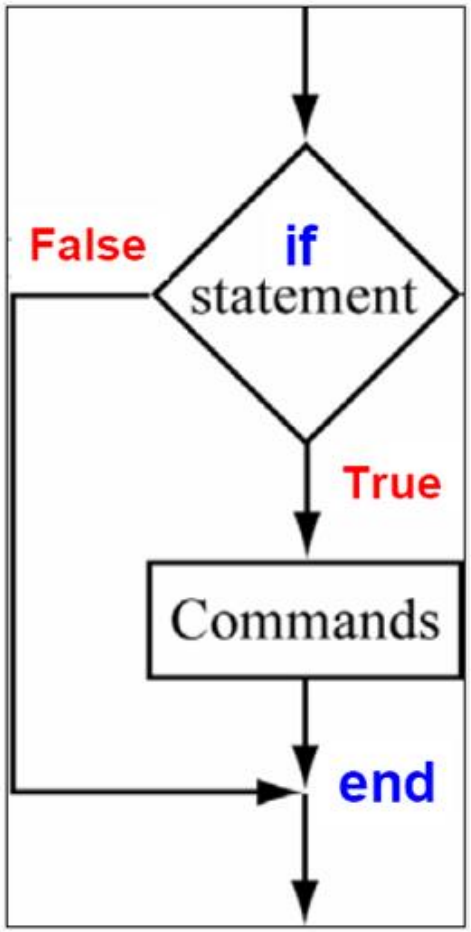
Three Forms of The if Statement

```
If  conditional statement
        commands
end
```

```
if  conditional statement
      command group 1
else
      command group 2
end
```

```
if  conditional statement 1
        command group 1
elseif  conditional statement 2
        command group 2
else
        command group 3
end
```

The if-end Statement




```
.....
.....          MATLAB program.
.....
if  conditional expression
        .....
        .....      A group of MATLAB
        .....      commands.
end
.....
.....          MATLAB program.
.....
```

Programming Example #1

1. Problem Definition

Write a MATLAB function file that computes the average grade based on a vector of test grades entered by the user.
If the student did not pass the class (i.e., average grade < 50) then display the message “The student did not pass the course.”

2. Problem Analysis

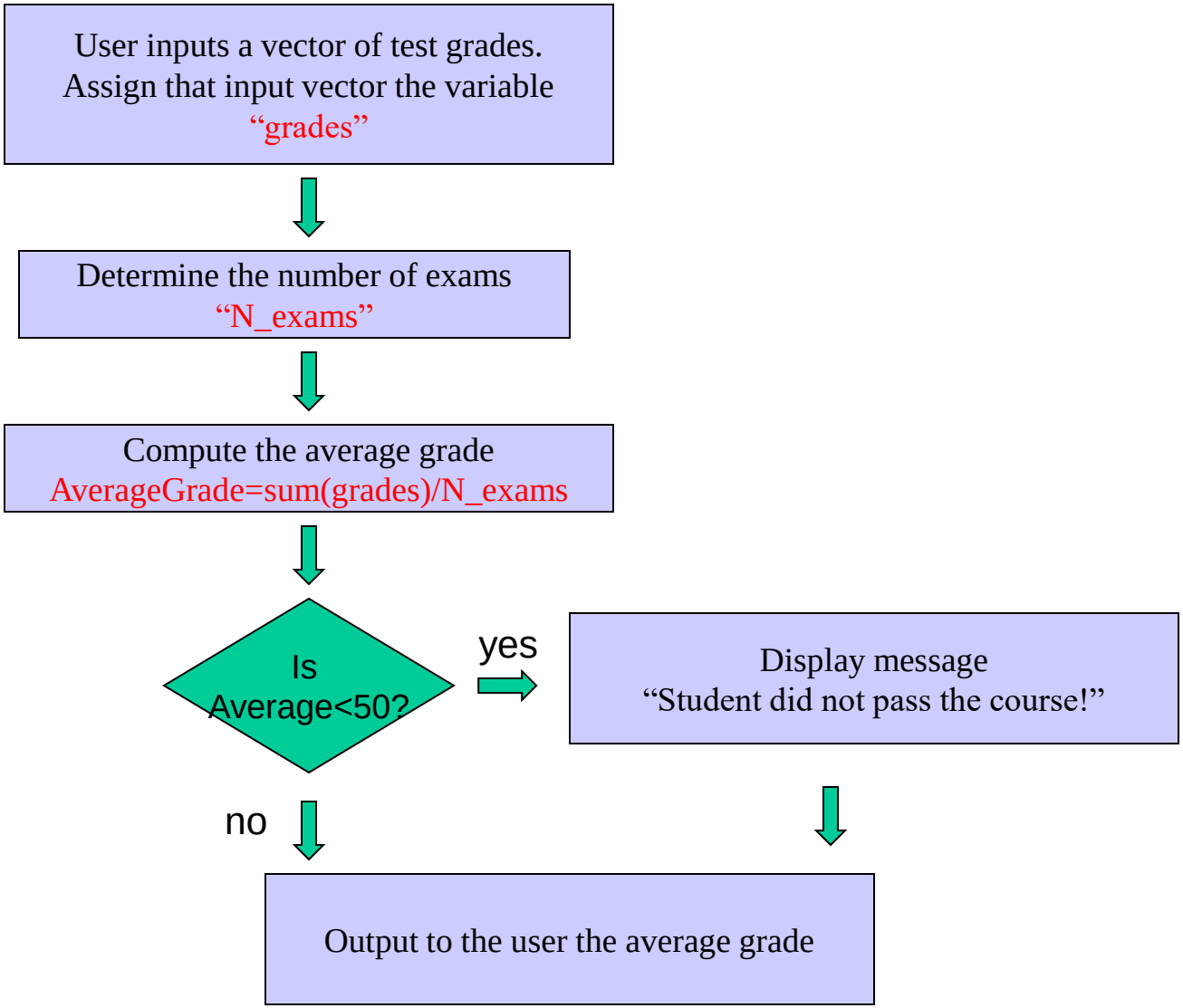
User Input: a vector of numbers that represent test grades

User Output: average test grade and message if the student did not pass the course

Equation: **Average = (sum of test grades) / (number of exams)**

Programming Example

3. Develop Algorithm (processing steps to solve problem)



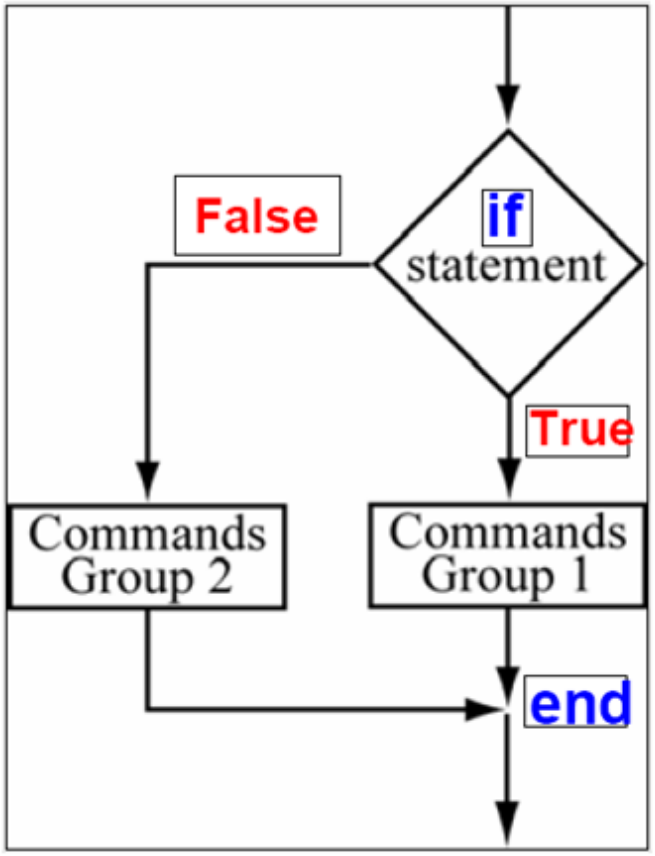
Programming Example #1

```
function [ave_grade] = calc_grade(grades)
% function [ave_grade] = calc_grade(grades)
% This function computes the average grade for a input vector of exam grades
% Inputs: grades = vector of exam grades
% Outputs: ave_grade = scalar of the average exam grade
%

N_exams = length(grades); % determine the number of exams
ave_grade = sum(grades)/N_exams; % determine the average grade
disp('The average grade is:')
disp(ave_grade)

if (ave_grade<50)
        disp('The student failed the class!')
end
```

The if-else-end Statement




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

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

Programming Example #2

1. Problem Definition

Write a MATLAB function file that computes the average grade based on a vector of test grades entered by the user.
If the student did not pass the class (i.e., average grade < 50) then display the message “The student failed the course.”, otherwise, if the exam grade is >= 50 then display the message “The student passed the course”.

2. Problem Analysis

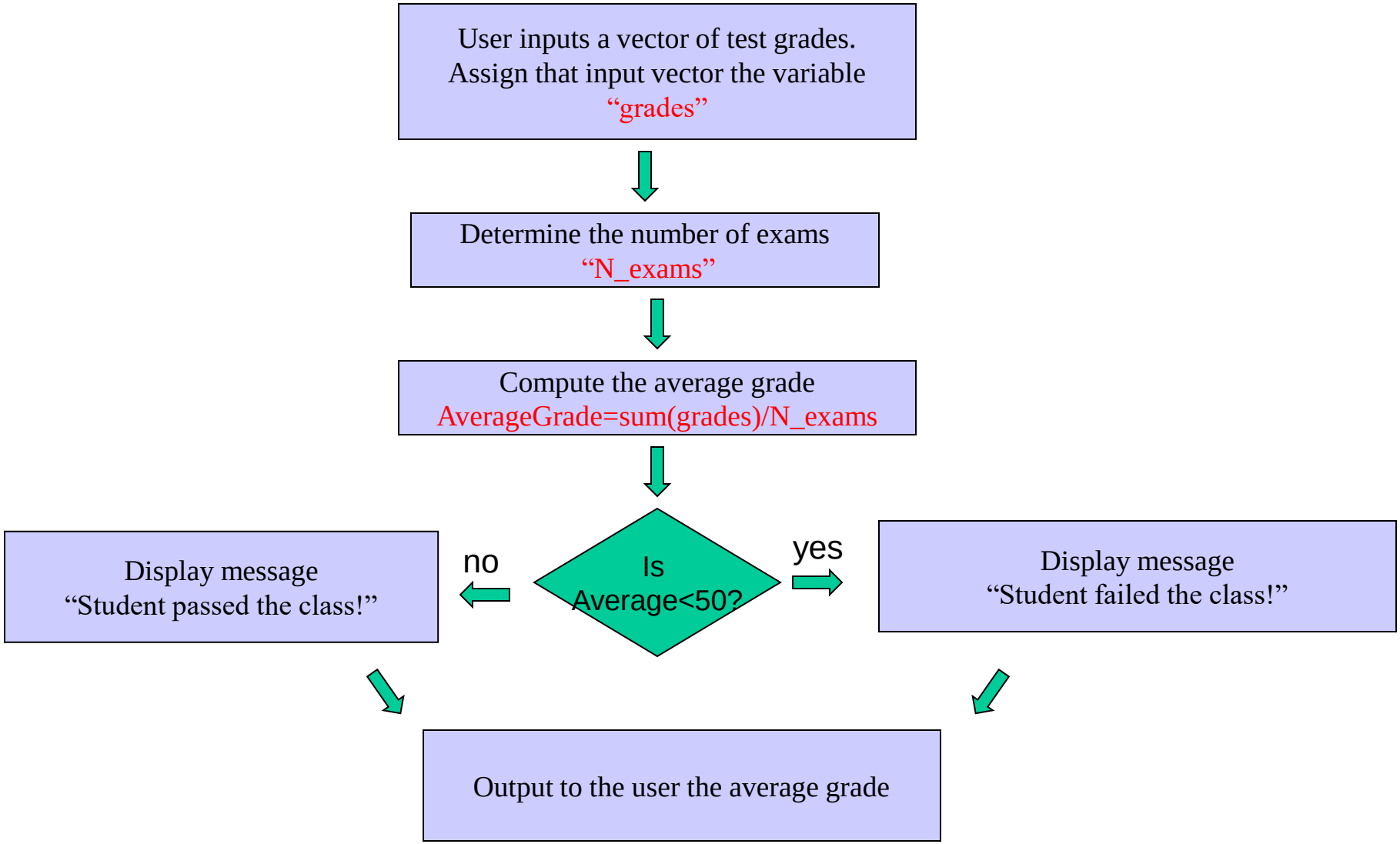
User Input: a vector of numbers that represent test grades

User Output: average test grade and message if the student passed/failed the course

Equation: **Average = (sum of test grades) / (number of exams)**

Programming Example

3. Develop Algorithm (processing steps to solve problem)



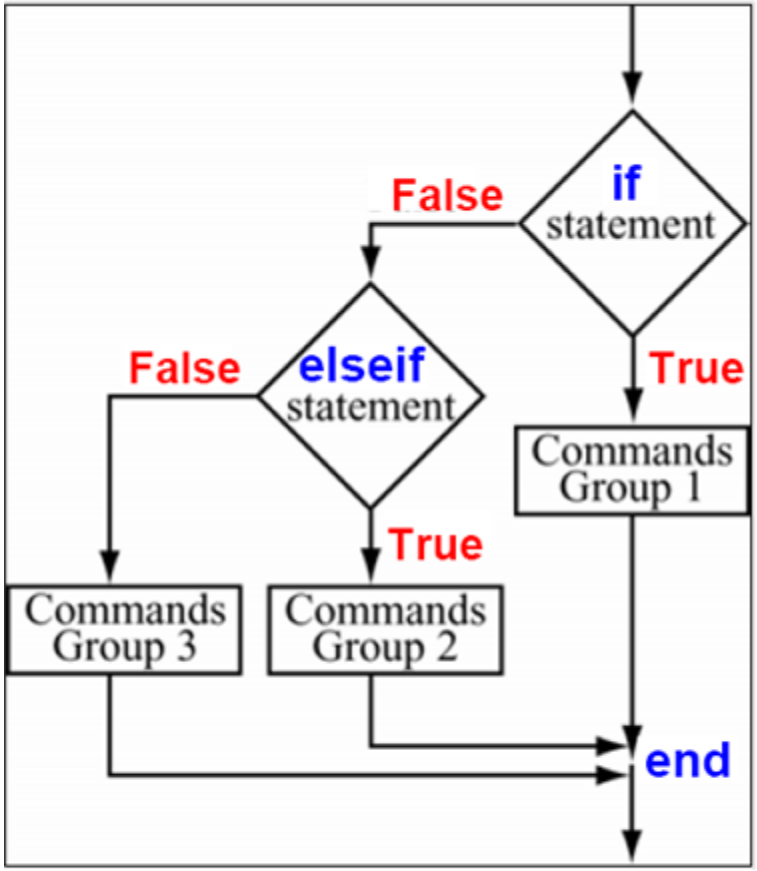
Programming Example #2

```
function [ave_grade] = calc_grade(grades)
% function [ave_grade] = calc_grade(grades)
% This function computes the average grade for a input vector of exam grades
% Inputs:  grades = vector of exam grades
% Outputs:  ave_grade = scalar of the average exam grade
%

N_exams = length(grades); % determine the number of exams
ave_grade = sum(grades)/N_exams; % determine the average grade
disp('The average grade is: ')
disp(ave_grade)

if (ave_grade < 50)
        disp('The student failed the class!')
else
        disp('The student passed the class!')
end
```

The if-elseif-else-end Statement




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

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

Programming Example #3

1. Problem Definition

Write a MATLAB function file that computes the tip on a restaurant bill.

If the bill is less than or equal to $10, give a flat tip of $1.80. If the bill is greater than $10 but less than or equal to $60 give a 15% tip. If the bill is greater than $60 then give a 20% tip.

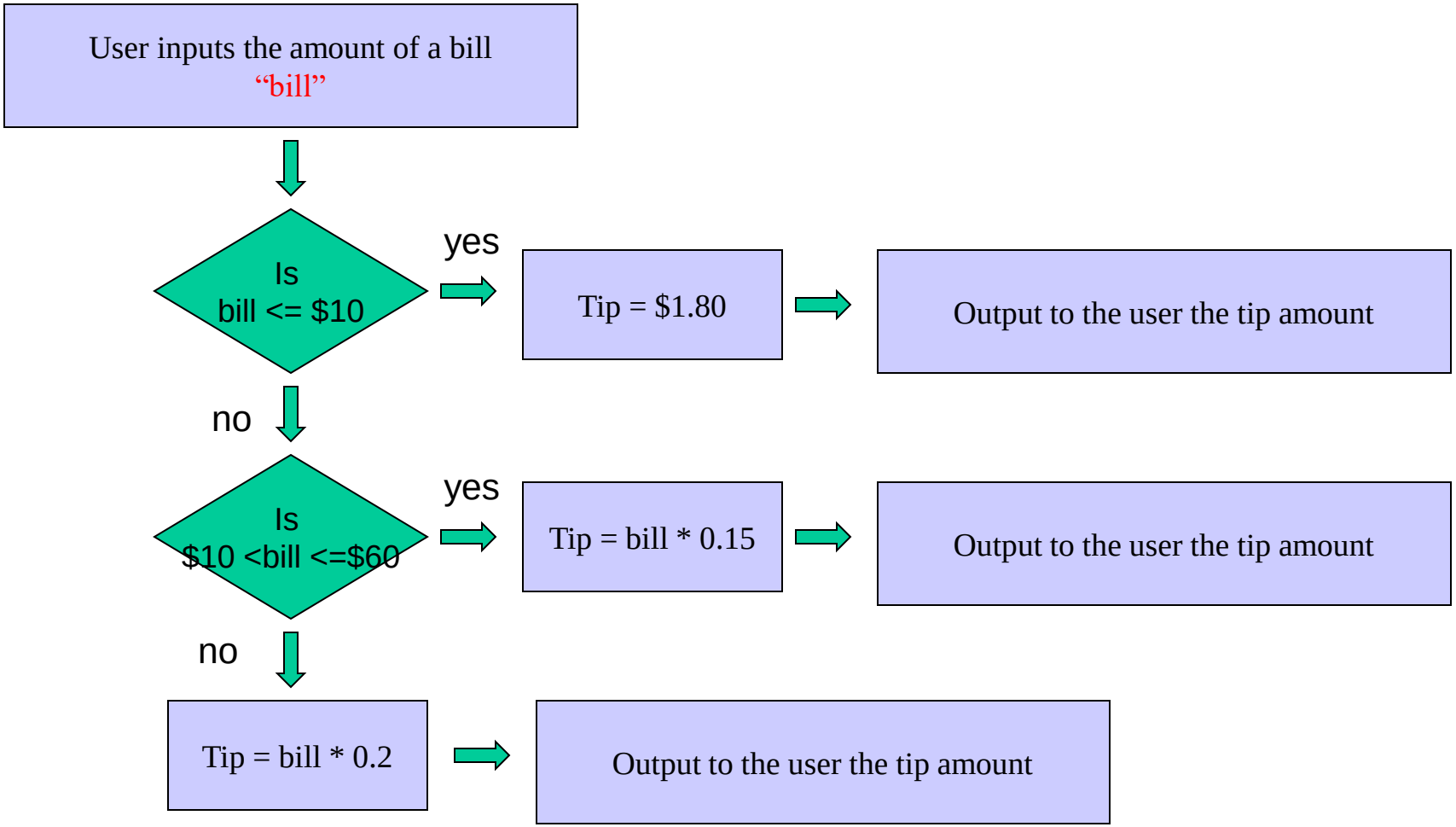
2. Problem Analysis

User Input: A scalar that represents the amount of a restaurant bill

User Output: A scalar that represents the tip amount

Programming Example

3. Develop Algorithm (processing steps to solve problem)



Programming Example #3

```
function [tip] = calc_tip(bill)
% function [tip] = calc_tip(bill)
% This function computes the tip on a restaurant bill
% Inputs:  bill = scalar bill value
% Outputs:  tip = scalar tip value
%
%format bank  % display numbers in a bank format

if (bill<=10)
        tip = 1.80;
elseif (bill <= 60) & (bill > 10)  %is it good here?
        tip = bill*0.15;
else
        tip = bill*0.20;
end

disp('The tip amount is: ')
disp(tip)
```

Comments about if-end Statements

- For every **if** command a computer program must have an **end** command.
- A program can have many **if ….. end** statements following each other.
- A computer program can perform the same task using different combinations of **if - end**, **if – else – end**, and **if– elseif – else – end** statements.
- Multiple **elseif** conditions are allowed within an **if– elseif – else – end** statement.
- An **else** condition is not required.

The switch Statements

Syntax

switch *switch_expression*
case *case_expression 1*
    statements
case *case_expression 2*
    statements
  ...
otherwise
    statements
end

The switch Statements example

```
n = input('Enter a number: ');
switch n
case -1
        disp('negative one')
case 0
        disp('zero')
case 1
        disp('positive one')
otherwise
        disp('other value')
end
```

The switch Statements example

```
grade = input('Enter your grade: ', 's');

switch(grade)
   case 'A'
       fprintf('Excellent!\n');
   case 'B'
       fprintf('Well done\n');
    case 'C'
       fprintf('Good\n');
    case 'D'
       fprintf('You passed\n');
    case 'F'
       fprintf('Better try again\n');
    otherwise
       fprintf('Invalid grade\n');
end
```

Introduction to Computing for Engineers 050IU

Iteration

Dr. Nguyen Ngoc Truong Minh

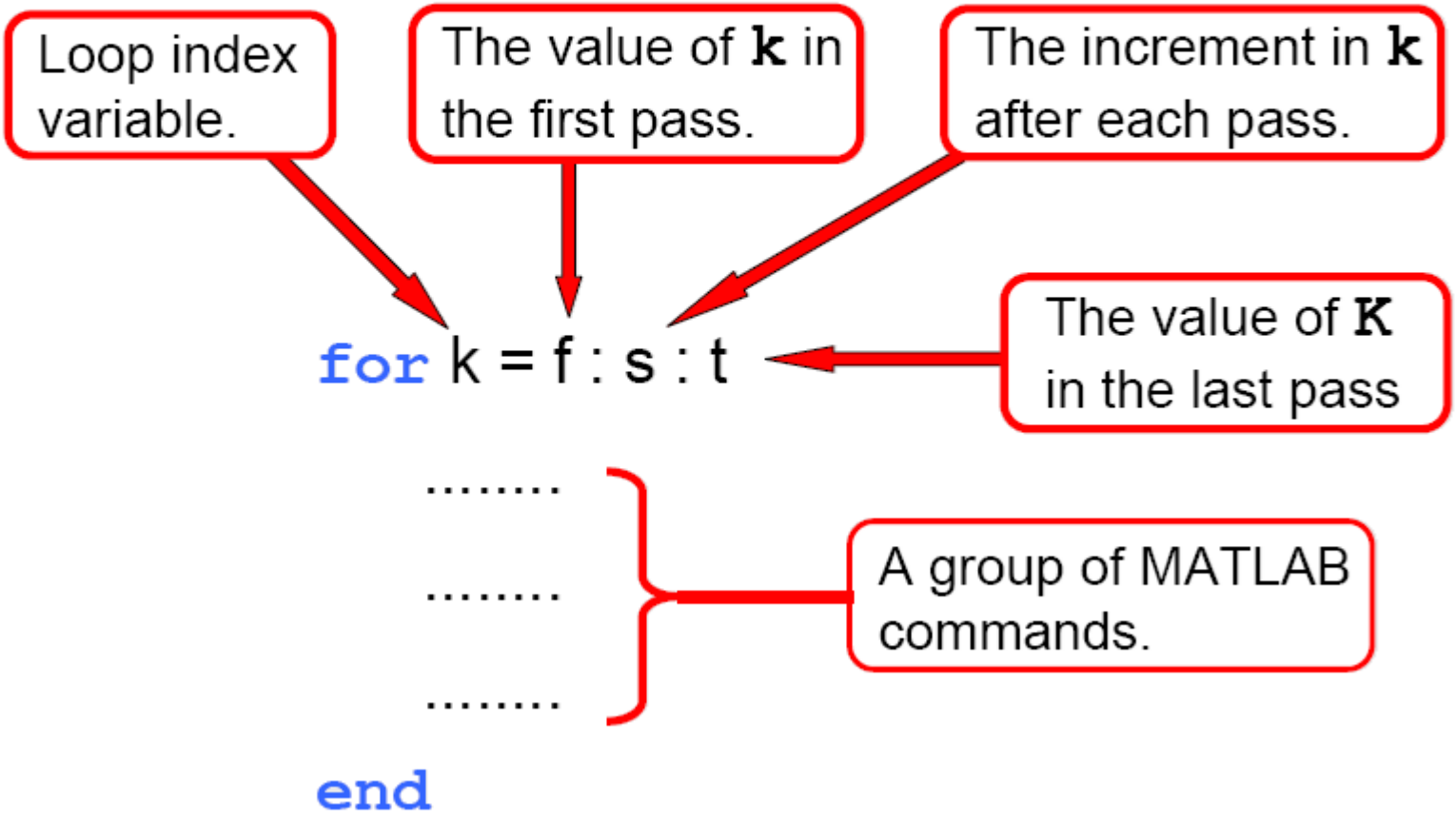
Definition of a Loop

- A loop is a group of commands in a computer program that are being repeated.
- Each repetition of the loop is called a pass.
- The number of passes can be set to be fixed, or the looping process is terminated when a specified condition is satisfied.
- In each pass some or all of the variables that are defined in the loop obtain new values.

Introducing the `for-end` Command

`for-end` loops are used when the number of passes is known in advance. A variable is used to control the looping process.

The general structure of a **`for-end`** loop is:



Introducing the `for-end` Command

- In the first pass, **k = f**, and the computer executes the commands between the **for** and the **end**.
- The computer goes back to the **for** command for the second pass. **k** obtains a new value equal to **k = f+s**, and the commands between the **for** and the **end** are executed with the new value of **k**.
- The process repeats itself until the last pass where **k = t**.

```
for  k = f : s : t
     ........
     ........
     ........
end
```

Example:

If **k** = 1:2:9 there are five loops. The values of **k** are:

1 3 5 7 9

Rules of the `for-end` Command

- The increment value **`s`** can be negative (i.e. **`k`** = 25:-5:10 produces four loops with: **`k`** = 25, 20, 15, 10).
- If **`s`** is omitted, the increment value is 1 (default).
- If **`f`** equals to **`t`**, the loop is executed once.
- The value of **`k`** should not be redefined within the loop.
- The looping continues until the value of **`k`** exceeds the value of **`t`** (i.e. **`k`** = 8:10:50 produces five loops with: **`k`** = 8, 18, 28, 38, 48).

Example of `for-end` loops

Type in the command window:

```
>> for k = 1:3:10
x = k^2
end
x =
     1
x =
    16
x =
    49
x =
   100
```

```
>> for k = 5:9
b = 2*k
end
b =
    10
b =
    12
b =
    14
b =
    16
b =
    18
```

If a step value is not entered, the default is 1.

```
>> for k = 10:2:20
a = k/3;
end
>> a
a =
    6.6667
>> k
k =
    20
```

Semicolon, so **a** is not printed after each pass.

a = 6.667 because in the last pass **k** = 20.

Comments about for-end Loops

- For every **for** command a computer program **MUST** have an **end** command.
- **for** loops can be used in the command window and in script and function files.
- A semicolon is not needed after the **for k = m:s:p** command to suppress printing of **k**.
- To display the value of **k** in each pass (sometimes useful for debugging) type **k** as one of the commands in the loop.
- Loops can include conditional statements and any other MATLAB commands (functions, plots, etc.)

Examples of using a for-end Loops

V = [5, 17, -3, 8, 0, -1, 12, 15, 20, -6, 6, 4, -7, 16]

Write a program in a script file that doubles the elements that are positive and are divisible by 3 and/or 5, and raise to the power of 3 the elements that are negative but greater than -5.

Examples of using a for-end Loops

```
V=[5, 17, -3, 8, 0, -7, 12, 15, 20 -6, 6, 4, -2, 16];
n=length(V);
for k = 1: n
    if  V(k)>0 & (rem(V(k),3)= =0 | rem(V(k),5)= =0)
        V(k)=2*V(k);
    elseif V(k)<0 & V(k)>-5
        V(k)=V(k)^3;
    end
end
V
```

Setting n to be equal to the number of elements in **V**.

for-end loop.

if-elseif -end statement.

When the file is executed, the display in the Command Window is:

```
V =
    10    17   -27     8     0    -7    24    30    40    -6    12     4    -8    16
```

Nested for-end Loops

A **for-end** loop can be nested within another **for-end** loop.

```
for k = 1 : 3
   for n = 1 : 5
   .
   commands
   .
   end
end
```

Every time **k** is increased by 1 the nested loop loops five times with the value of **n** ranging from 1 through 5.

Overall the commands will be executed 15 times with the values of:

k = 1	**n** = 1	**k** = 2	**n** = 1	**k** = 3	**n** = 1
k = 1	**n** = 2	**k** = 2	**n** = 2	**k** = 3	**n** = 2
k = 1	**n** = 3	**k** = 2	**n** = 3	**k** = 3	**n** = 3
k = 1	**n** = 4	**k** = 2	**n** = 4	**k** = 3	**n** = 4
k = 1	**n** = 5	**k** = 2	**n** = 5	**k** = 3	**n** = 5

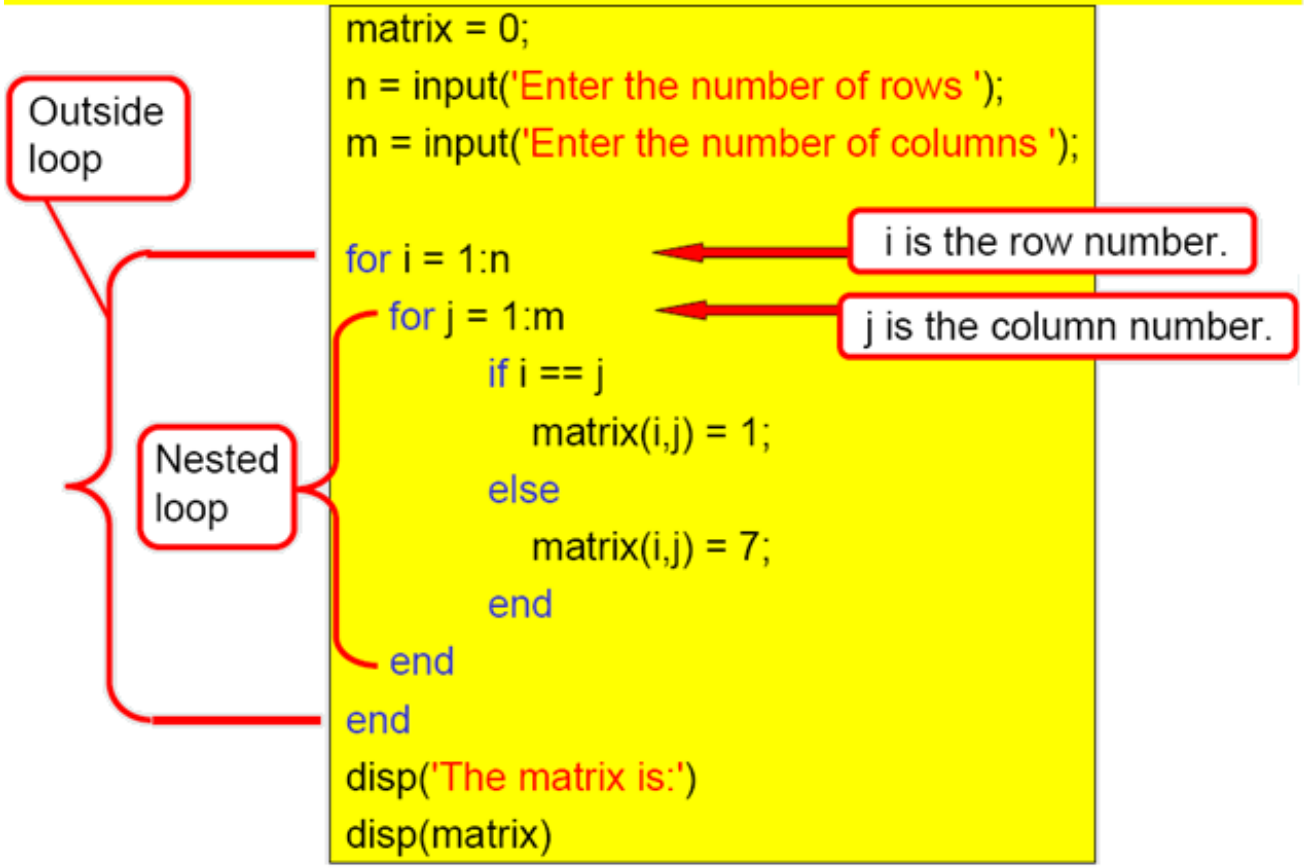
Example of Nested for-end Loops

- **Develop a following matrix of the size N by M**

1	7	7	7
7	1	7	7
7	7	1	7
7	7	7	1

Example of Nested for-end Loops

```
% A script file that demonstrates the use of nested for - end loop.
% The file creates a (nxm) matrix in which all the terms are 7 except
% the terms whose row number is equal to the column number.
% These terms are equal to 1.
```



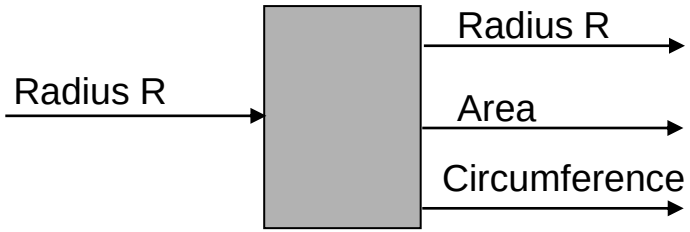
Example 1: The for-end Structure

Step 1: Describe problem: **Write a MATLAB program to**

- ***calculate the area and the circumference of TEN circles***
- ***enter the radius as an input variable***
- ***output radius, area and circumference IF the area is greater than 20 square units***
- ***output the number of circles with area < 20***

Step 2: Describe input and output

INPUT **CALCULATE** **OUTPUT**

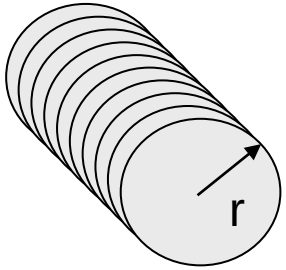


Example 1 … cont'd.

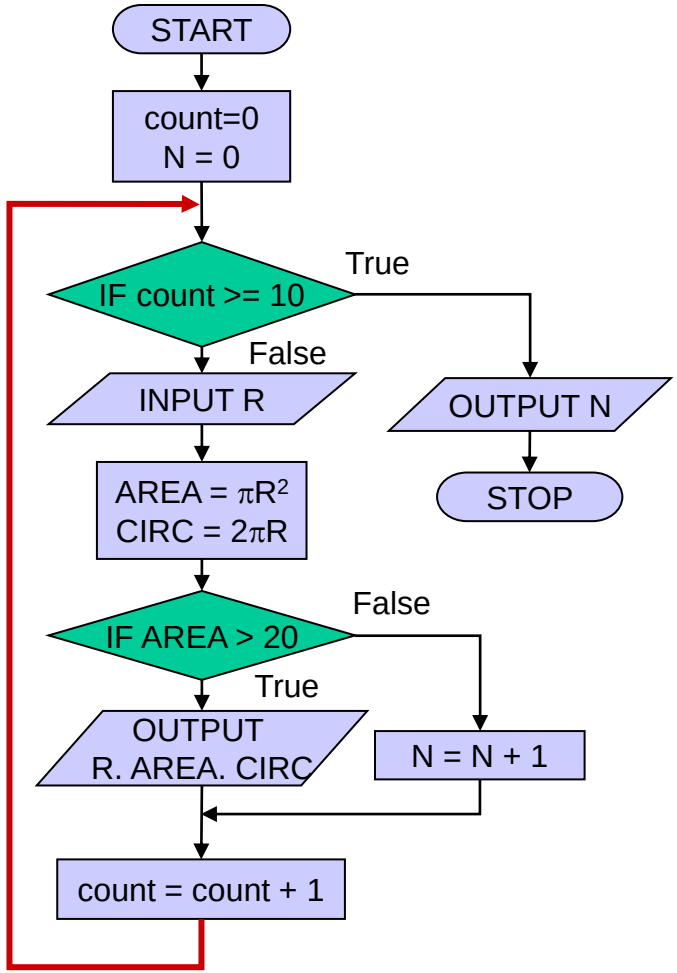
- **Step 3: Develop the solution**
  - ***Describe the algorithm:***

Area = πr^2

Circum = $2\pi r$



  - ***Develop the process:***
    - Calculate the area
    - Calculate the circumference
    - If the area is big enough,
      - ***Display radius, area and circumference***
      - ***otherwise, display nothing***
    - Repeat 10 times



Example 1: … cont'd

- **Step 4: Develop the solution**
  - ***Write Matlab code***

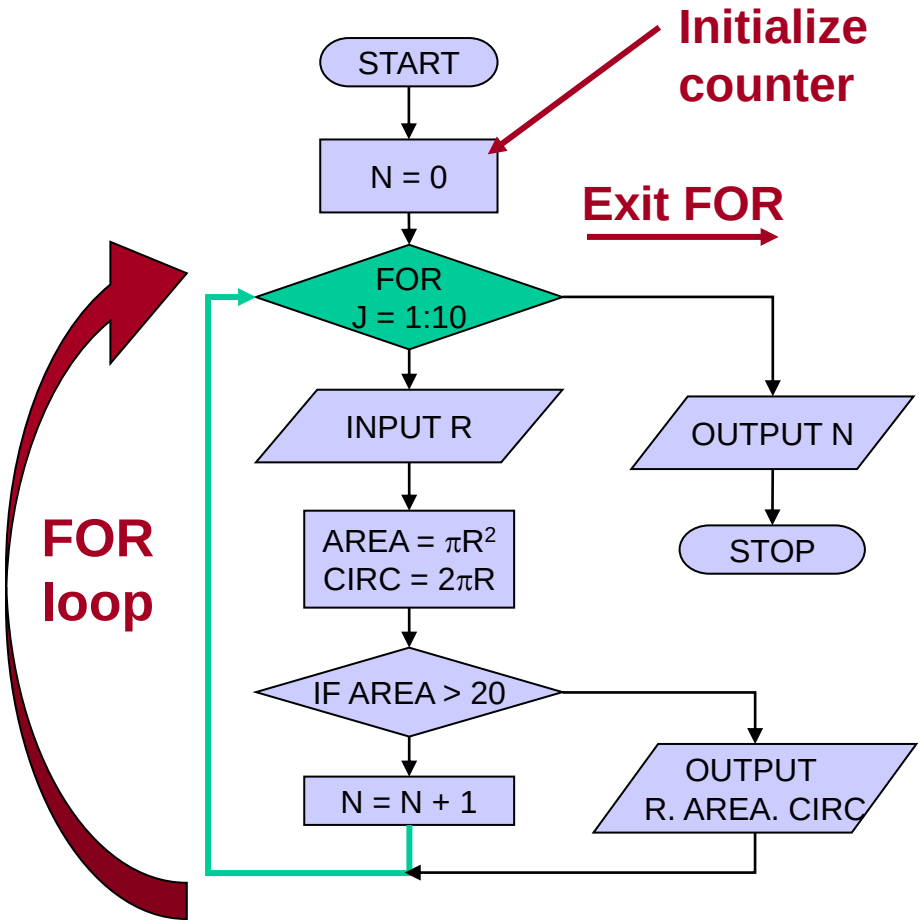
Use a `FOR` loop:

calculate the area and the circumference of TEN circles

allow the radius to be an input variable

output radius, area and circumference IF the area is greater than 20 square units.

output the number of circles with area $\leq$ 20.



Example 1: … cont'd

```
% calculate the area and circumference of 10 circles
% print it out if the area is greater than 20
N = 0;
for i = 1:10
    radius = input( '\nPlease enter a radius:' );
    radius = abs(radius);  % make sure always is positive!
    area = pi * radius ^2;
    circumference = 2 * pi * radius;
    if area > 20
        fprintf('\n Radius = %f units', radius);
        fprintf('\n Area = %f units squared', area);
        fprintf('\n Circumference = %f units',
circumference);
    else
        N = N + 1;
    end
end
fprintf('\n %f circles with area less than 20', N);
```

Loop for i=1…10

Print only if area>20

End of FOR loop

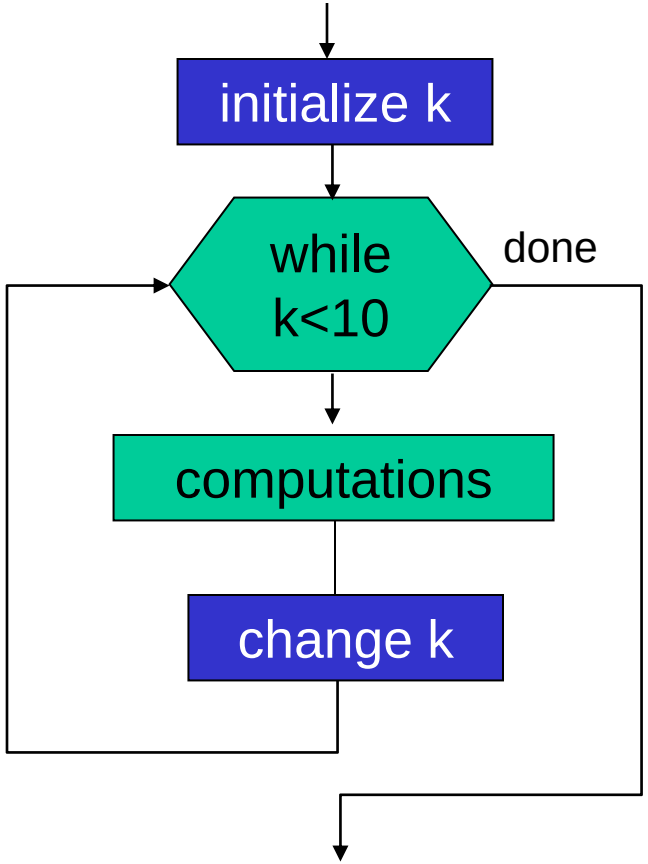
Example 1: … cont'd

- **Step 5: Test the results**

```
>> basicFOR
Please enter a radius:0
Please enter a radius:1
Please enter a radius:2
Please enter a radius:3
 Radius = 3.000000 units
 Area = 28.274334 units squared
 Circumference = 18.849556 units
Please enter a radius:-4
 Radius = 4.000000 units
 Area = 50.265482 units squared
 Circumference = 25.132741 uni
Please enter a radius:5
 Radius = 5.000000 units
 Area = 78.539816 units squared
 Circumference = 31.415927 units
Please enter a radius:6
 Radius = 6.000000 units
 Area = 113.097336 units squared
 Circumference = 37.699112 units
Please enter a radius:7
 Radius = 7.000000 units
 Area = 153.938040 units squared
 Circumference = 43.982297 units
Please enter a radius:8
 Radius = 8.000000 units
 Area = 201.061930 units squared
 Circumference = 50.265482 units
Please enter a radius:9
 Radius = 9.000000 units
 Area = 254.469005 units squared
 Circumference = 56.548668 units
 3.000000 circles with area less than 20
>>
```

Handled negative radius

`while` Loop




```
k=0;
while k<10
  % computations;
  k=k+1;
end
```

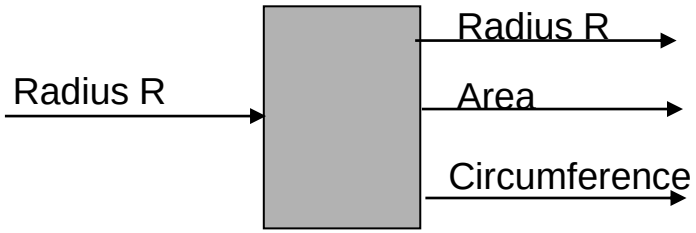
- **Will do computational loop ONLY if WHILE condition is met**
- **Be careful to initialize WHILE variable**
- **Can loop forever if WHILE variable is not updated within loop!!!**

Problem 2: Introducing the `while` Structure

- **Step 1: State the problem:** **Write a MATLAB program to**
  - ***calculate the area and the circumference of ANY NUMBER of circles so long as the radius is greater than zero***
  - ***take the radius as an input variable***
  - ***output radius, area and circumference IF the area is greater than 20 square units.***
  - ***output the number of circles with area <=20.***

Step 2: Describe input and output

INPUT **CALCULATE** **OUTPUT**



How will we be able to stop the calculation process? ANSWER: we'll stop when a zero radius is entered.

Step 3: Define test cases

R=0 → Area=0 and Circumference=0

R=1 → Area=π and Circumference=2 π

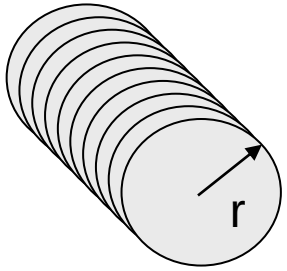
Create case with 0 areas > 20 and with specified # > 20 to test

Problem 2: … cont'd

- **Step 4: Develop the solution**
  - ***Describe the algorithm:***

Area = πr^2

Circum = $2\pi r$



  - ***Develop the process:***
    - Repeat until radius = 0
      - ***calculate the circumference***
      - ***calculate the area***
      - ***If the area is big enough,***
        - » Display radius, area and circumference
        - » otherwise, display nothing

Problem 2: … cont'd

- **Step 4: Develop the solution**
  - ***Develop a Matlab solution***

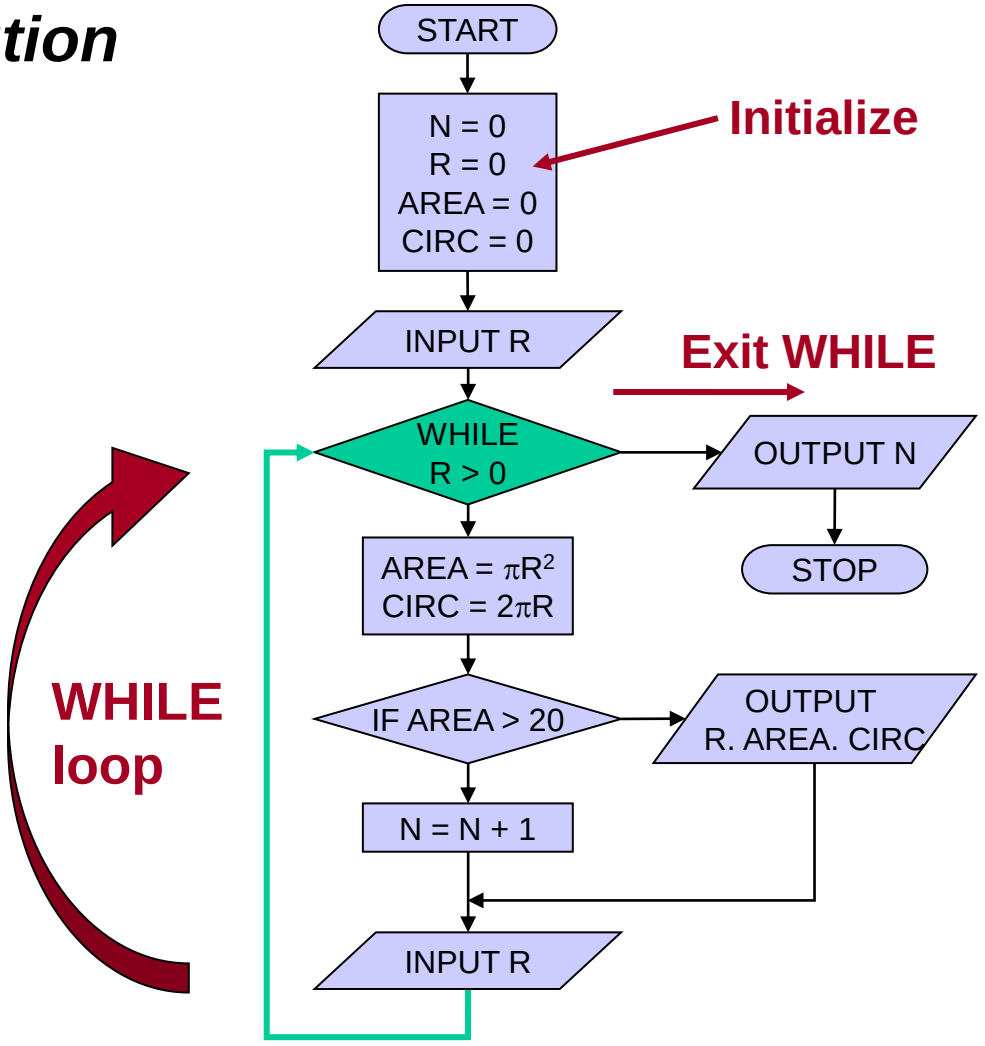
Use a `WHILE` loop:

calculate the area and the circumference of circles while the radius is > 0

allow the radius to be an input variable

output radius, area and circumference IF the area is greater than 20 square units.

output the number of circles with area <= 20.



Problem 2 - Programming the `while` Loop

```
% calculate the area and circumference of circles
% print results if the area is greater than 20; print the
% number of circules with area less than 20; terminate on area<=0
N = 0;
radius = input('\nPlease enter a radius: ');
while radius > 0
    area = pi * radius ^2;
    circumference = 2 * pi * radius;
    if area > 20
        fprintf('\n Radius = %f units', radius);
        fprintf('\n Area = %f units squared', area);
        fprintf('\n Circumference = %f units', circumfere    ,;
    else
        N = N + 1;
    end
    radius = input('\nPlease enter a radius: ');
end
fprintf('\n %f circles with area less than 20', N);
```

Loop while radius > 0

Note the need to repeat the user input here

While end

Problem 2: … cont'd

- **Step 5: Test the result**

```
Please enter a radius: 5

 Radius = 5.000000 units
 Area = 78.539816 units
squared
 Circumference = 31.415927
units
Please enter a radius: 4

 Radius = 4.000000 units
 Area = 50.265482 units
squared
 Circumference = 25.132741
units
Please enter a radius: 3

 Radius = 3.000000 units
 Area = 28.274334 units
squared
 Circumference = 18.849556
units
Please enter a radius: 2

Please enter a radius: 1

Please enter a radius: 0

 2.000000 circles with
area less than 20
>>
```

Break Command

MATLAB encounters a **break** command within a loop, MATLAB jumps to the **end** command of the loop and continues to execute the commands that follow.

The **break** command is typically used within a conditional statement (**if** statement) to terminate the execution of a loop if some condition is satisfied.

Break Command Example

```
% A script file that demonstrate the use of the break command.
% Given an initial investment, saving goal and expected return,
% the file calculates the number of years it will take to reach the goal.
```

(The file continues on the next slide)

$$B = A\left(1+\frac{r}{100}\right)^{n}$$

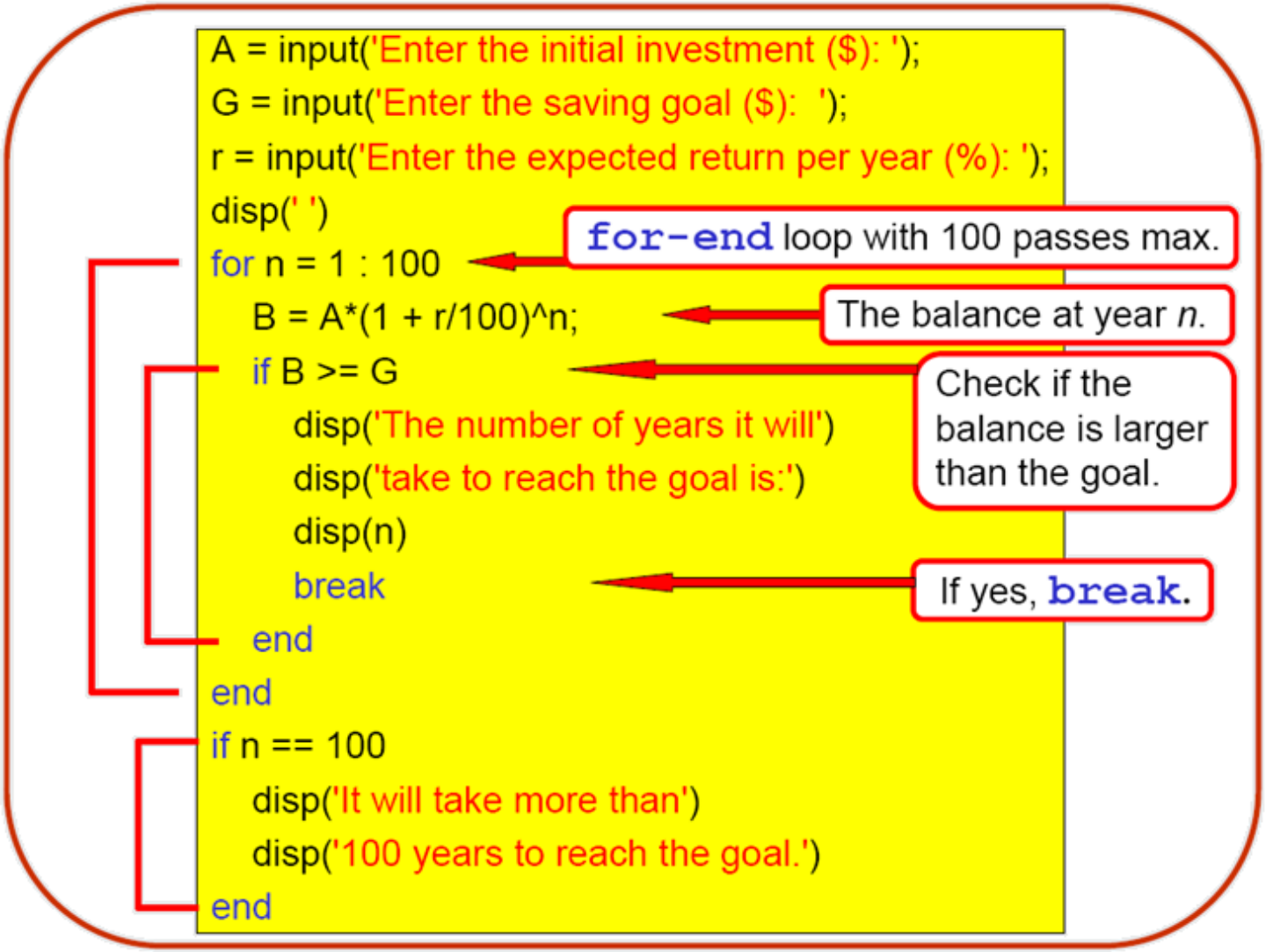
B Balance.

A Initial investment.

r Return (%) per year.

n Number of years.

Break Command Example



Break Command Example

```
Enter the initial investment ($): 2000
Enter the saving goal ($):  5000
Enter the expected return per year (%): 6

The number of years it will
take to reach the goal is:
    16
```

```
>> Lecture9Example2
Enter the initial investment ($): 1500
Enter the saving goal ($):  100000
Enter the expected return per year (%): 4

It will take more than
100 years to reach the goal.
```

Questions?

Programming in MATLAB

- In a simple program the commands are executed in the order they are typed.
- Many situations may require that:
  - Commands will not be executed in order.
  - Different commands are executed in different runs.
  - The execution of a group of commands is repeated many times.