

Programming in MATLAB

RELATIONAL OPERATORS

- A relational operator compares two numbers by determining whether a comparison statement (e.g. $5 < 8$) is true or false.

✓ If the statement is true, it is assigned a value of 1.

☒ If the statement is false, it is assigned a value of 0.

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

- The “equal to” relational operator consists of two = signs (with no space between them), since one = sign is the assignment operator.
- Relational operators are used as arithmetic operators within a mathematical expression.

Examples:

```
>> 5>8
ans =
    0
```

Checks if 5 is larger than 8.
Since the comparison is false (5 is not larger than 8) the answer is 0.

```
>> a=5<10
a =
    1
```

Checks if 5 is smaller than 10, and assigns the answer to a.
Since the comparison is true (5 is smaller than 10) the number 1 is assigned to a.

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

Using relational operators in math expression.

Equal to 1 since 6 is smaller than 10.

Equal to 0 since 7 is not larger than 8.

Equal to 1 since $5*3$ is equal to $60/4$.

```
y =
     2
```

```
>> b=[15 6 9 4 11 7 14]; c=[8 20 9 2 19 7 10];
```

Define vectors b and c.

```
>> d=c>=b
```

Checks which c elements are larger or equal to b elements.

```
d =
     0     1     1     0
```

Assigns 1 where an element of c is larger than or equal to an element of b.

```
>> b == c
```

Checks which b elements are equal to c elements.

```
ans =
     0     0     1     0     0     1     0
```

```
>> b~=c
```

Checks which b elements are not equal to c elements.

```
ans =
     1     1     0     1     1     0     1
```

```
>> f=b-c>0
```

Subtracts c from b and then checks which elements are larger than zero.

```
f =
     1     0     0     1     0     0     1
```

```
>> A=[2 9 4; -3 5 2; 6 7 -1]
```

Define a 3×3 matrix A.

```
A =
     2     9     4
    -3     5     2
     6     7    -1
```

Checks which elements in A are smaller or equal to 2. Assigns the results to matrix B.

```
>> B=A<=2
```

```
B =
     1     0     0
     1     0     1
     0     0     1
```

- When a logical vector is used for addressing another vector, it extracts from that vector the elements in the positions where the logical vector has 1's.

Example:

```
>> r = [8 12 9 4 23 19 10]
r =
     8     12     9     4    23    19    10
>> s=r<=10
s =
     1     0     1     1
>> t=r(s)
t =
     8     9     4    10
>> w=r(r<=10)
w =
     8     9     4    10
```

Define a vector r .

Checks which r elements are smaller than or equal to 10.

A logical vector s with 1's at positions where elements of r are smaller than or equal to 10.

Use s for addresses in vector r to create vector t .

Vector t consists of elements of r in positions where s has 1's.

The same can be done in one step.

- The arithmetic operations (+, −, *, /, \) have precedence over relational operations.
- The relational operators themselves have equal precedence and are evaluated from left to right.
- Parentheses can be used to alter the order of precedence.

Examples:

```
>> 3+4<16/2
ans =
     1
>> 3+(4<16)/2
ans =
    3.5000
```

+ and / are executed first.

The answer is 1 since $7 < 8$ is true.

$4 < 16$ is executed first, and is equal to 1, since it is true.

3.5 is obtained from $3 + 1/2$.

LOGICAL OPERATORS:

<u>Logical Operator</u>	<u>Name</u>	<u>Description</u>
& Example: A&B	AND	Operates on two operands (A and B). If both are true, the result is true (1), otherwise the result is false (0).
 Example: A B	OR	Operates on two operands (A and B). If either one, or both are true, the result is true (1), otherwise (both are false) the result is false (0).
~ Example: ~A	NOT	Operates on one operand (A). Gives the opposite of the operand. True (1) if the operand is false, and false (0) if the operand is true.

- Logical operators have numbers as operands.

**Nonzero number is true,
Zero number is false.**

- Logical operators (like relational operators) are used as arithmetic operators within a mathematical expression.

Examples:

[illegible]

```
>> t=25*((12&0)+(~0)+(0|5))
```

Using logical operators in math expression.

```
t =
    50
```

```
>> x=[9 3 0 11 0 15]; y=[2 0 13 -11 0 4];
```

Define two vectors x and y.

```
>> x&y
```

```
ans =
     1     0     0     1
```

The outcome is a vector with 1 in every position that both x and y are true (nonzero elements), and 0's otherwise.

```
>> z=x|y
```

```
z =
     1     1     1     1
```

The outcome is a vector with 1 in every position that either or both x and y are true (nonzero elements), and 0's otherwise.

```
>> ~(x+y)
```

```
ans =
     0     0     0     1
```

The outcome is a vector with 0 in every position that the vector x + y is true (non-zero elements), and 1 in every position that x + y is false (zero element).

Order of precedence:

The following is the order used by MATLAB:

<u>Precedence</u>	<u>Operation</u>
1 (highest)	Parentheses (If nested parentheses exist, inner have precedence).
2	Exponentiation.
3	Logical NOT (~).
4	Multiplication, division.
5	Addition, subtraction.
6	Relational operators (>, <, >=, <=, ==, ~=).
7	Logical AND (&).
8 (lowest)	Logical OR ().

- If two or more operations have the same precedence, the expression is executed in order from left to right.

The following are examples of expressions that include arithmetic, relational, and logical operators:

```
>> x=-2; y=5;
```

Define variables x and y.

```
>> -5<x<-1
```

```
ans =  
      0
```

This inequality is correct mathematically. The answer, however, is false since MATLAB executes from left to right. $-5 < x$ is true (=1) and then $1 < -1$ is false (0).

```
>> -5<x & x<-1
```

```
ans =  
      1
```

The mathematically correct statement is obtained by using the logical operator &. The inequalities are executed first. Since both are true (1), the answer is 1.

```
>> ~(y<7)
```

```
ans =  
      0
```

$y < 7$ is executed first, is true (1), ~ 1 is 0.

```
>> ~y<7
```

```
ans =  
      1
```

$\sim y$ is executed first. y is true (1) (since y is nonzero), ~ 1 is 0, and $0 < 7$ is true (1).

```
>> ~( (y>=8) | (x<-1) )
```

```
ans =  
      0
```

$y \geq 8$ (false), and $x < -1$ (true) are executed first. OR executed next (true). $\sim$ executed last, gives false (0).

```
>> ~(y>=8) | (x<-1)
```

```
ans =  
      1
```

$y \geq 8$ (false), and $x < -1$ (true) are executed first. NOT of $(y \geq 8)$ executed next (true). OR executed last, gives true (1).

CONDITIONAL STATEMENTS

- A conditional statement is a command that allows MATLAB to make a decision of whether to execute a group of commands that follow the conditional statement, or to skip these commands.

`if` conditional expression consisting of relational and/or logical operators.

Examples:

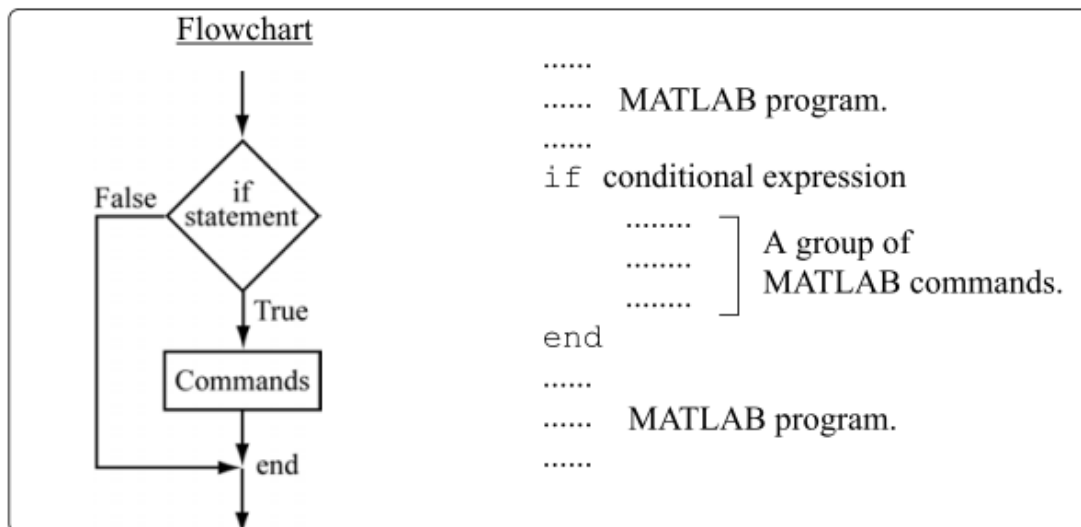
```
if a < b
if c >= 5
if a == b
if a ~= 0
if (d<h) & (x>7)
if (x~=13) | (y<0)
```

All the variables must
have assigned values.

- For every `if` statement there is an `end` statement.
- The if statement is commonly used in three structures:
 - if-end,
 - if-else-end,
 - if-elseif-else-end,

❖ The `if-end` structure

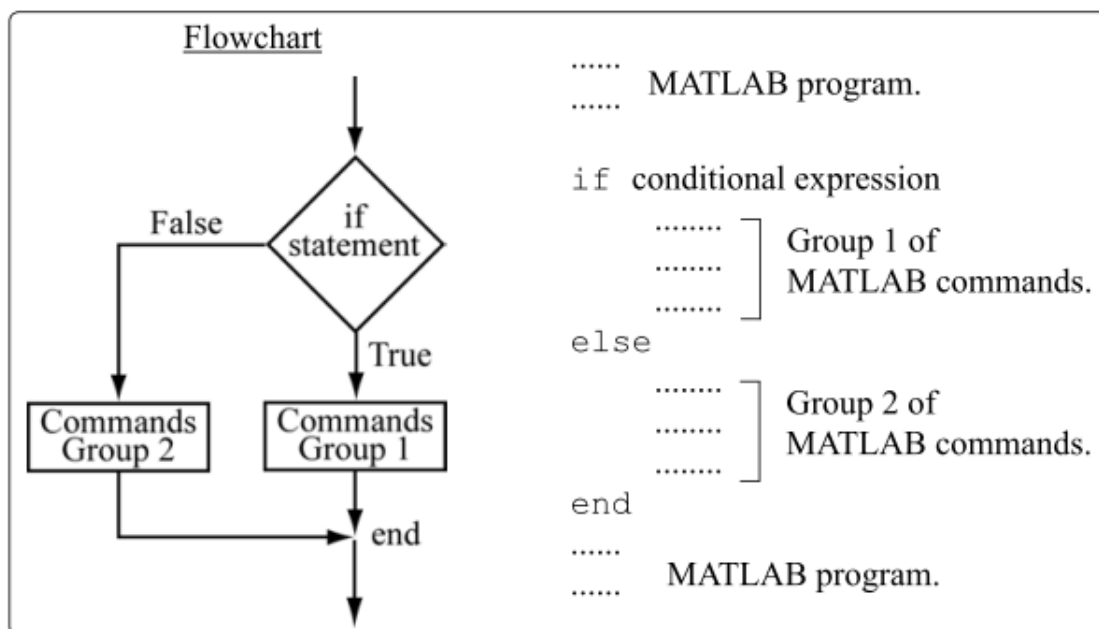
- The if-end structure of the conditional statement is shown schematically in Figure below.



- If the conditional expression in the if statement is true (1), the program continues to execute the commands that follow the if statement all the way down to the end statement.
- If the conditional expression is false (0), the program skips the group of commands between the if and the end, and continues with the commands that follow the end.
- if and end appear on the screen in blue color.

❖ The if-else-end Structure

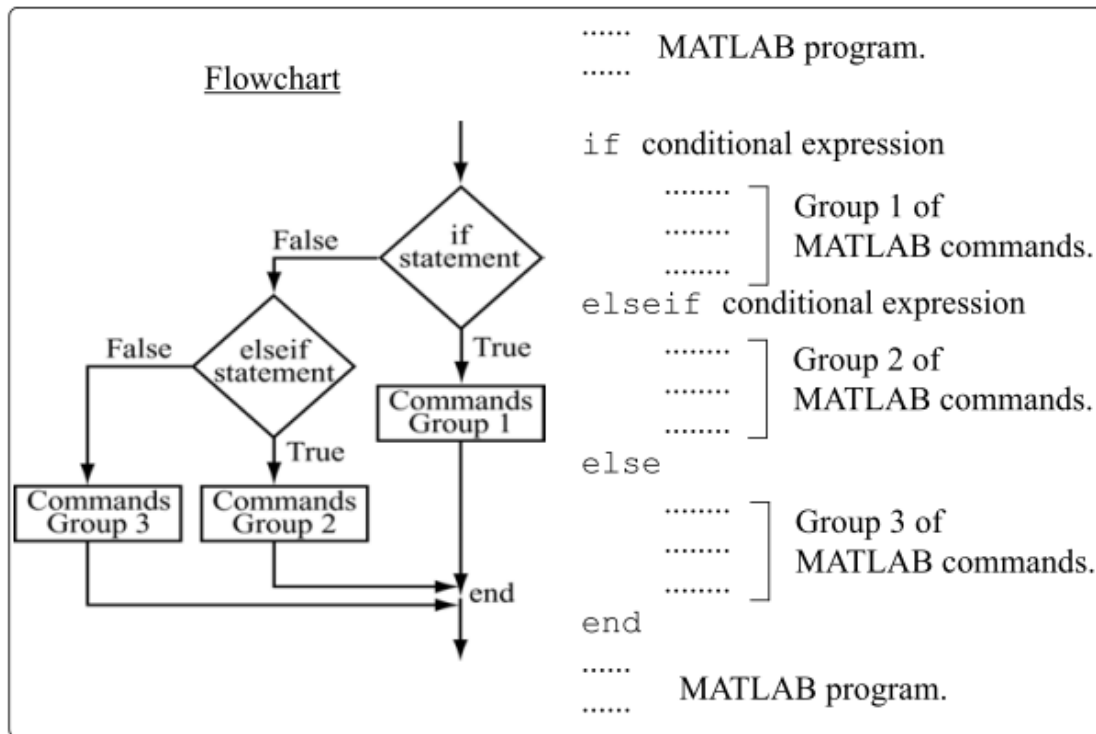
- The if-else-end structure is shown in Figure below



- If the conditional expression is true, the program executes group 1 of commands between the if and the else statements and then skips to the end.
- If the conditional expression is false, the program skips to the else, and then executes group 2 of commands between the else and the end.

The if-elseif-else-end structure

- The if-elseif-else-end structure is shown in the figure below.



- If the conditional expression is true, the program executes group 1 of commands between the if and the elseif statement and then skips to the end.
 - If the conditional expression in the if statement is false, the program skips to the elseif statement.
 - If the conditional expression in the elseif statement is true the program executes group 2 of commands between the elseif and the else and then skips to the end.
 - If the conditional expression in the elseif statement is false the program skips to the else and executes group 3 of commands between the else and the end.
- Several elseif statements and associated groups of commands can be added. In this way more conditions can be included.

The switch-case statement

The structure of the statement is shown in the figure below.

```
..... MATLAB program.

switch switch expression
    case value1
        ..... ] Group 1 of commands.
    case value2
        ..... ] Group 2 of commands.
    case value3
        ..... ] Group 3 of commands.
    otherwise
        ..... ] Group 4 of commands.
end
..... MATLAB program.
```

- The value of the switch expression in the switch command is compared with the values that are next to each of the case statements.
- If a match is found, the group of commands that follow the case statement with the match are executed.
- Only one group of commands, the one between the case that matches and either the case, otherwise or end statement that is next, is executed.
- If there is more than one match, only the first matching case is executed.
- If no match is found and the otherwise statement (which is optional) exists, the group of commands between otherwise and end is executed.
- If no match is found and the otherwise statement does not exist, none of the command groups is executed.
- A case statement can have more than one value. This is done by typing the values in the form: {value1, value2, value3,...}.
- The case is executed if at least one of the values matches the value of switch expression.
- In MATLAB only the first matching case is executed.

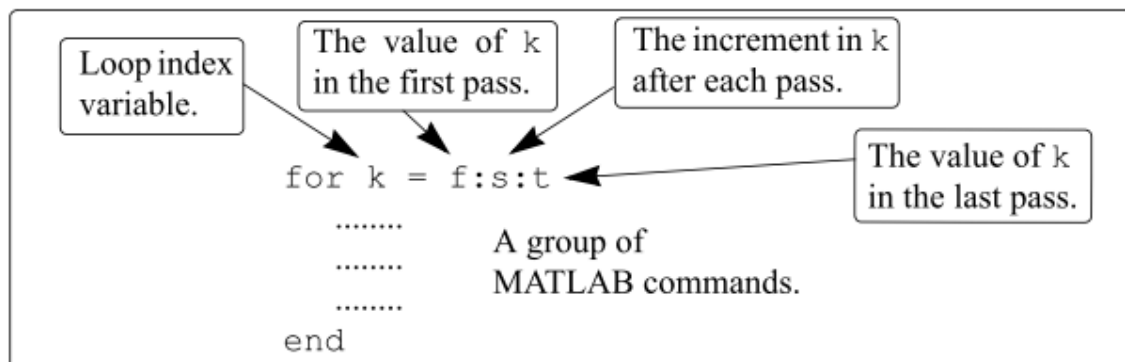
- After the group of commands that are associated with the first matching are executed, the program skips to the end statement.
- This is different than the C language where break statements are required.

LOOPS

- A loop is another method to alter the flow of a computer program.
- In a loop, the execution of a command, or a group of commands, is repeated several times consecutively.
- Each round of execution is called a pass.
- MATLAB has two kinds of loops.
 - 1) `for-end` loops: the number of passes is specified when the loop starts.
 - 2) `while-end` loops: the number of passes is not known ahead of time, and the looping process continues until a specified condition is satisfied.
- Both kinds of loops can be terminated at any time with the `break` command (see Section 7.6).

for-end Loops

- the execution of a command, or a group of commands, is repeated a predetermined number of times.



- The loop index variable can have any variable name (usually `i`, `j`, `k`, `m`, and `n` are used).
- `i` and `j` should not be used if MATLAB is used with complex numbers.
- ❖ In the first pass $k=f$ and the computer executes the commands between the `for` and the `end` commands.
- ❖ Then, the program 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` commands are executed with the new value of k .
- ❖ The process repeats itself until the last pass where $k=t$.
- ❖ Then the program does not go back to the `for`, but continues with the commands that follow the `end` command.
For example, if $k=1:2:9$, there are five loops, and the value of k in each loop is 1, 3, 5, 7, and 9.
- ❖ The increment s can be negative (i.e. $k=25:-5:10$ produces four passes with $k=25, 20, 15, 10$).
- ❖ If the increment value s is omitted, the value is 1 (default) (i.e. $k=3:7$ produces five passes with $k=3, 4, 5, 6, 7$).
- ❖ If $f=t$, the loop is executed once.
- ❖ If $f>t$ and $s>0$, or if $f<t$ and $s<0$ the loop is not executed.
- ❖ If the values of k , s , and t are such that k cannot be equal to t , then:
 - if s is positive, the last pass is the one where k has the largest value that is smaller than t (i.e. $k=8:10:50$ produces five passes with $k=8, 18, 28, 38, 48$).
 - If s is negative the last pass is the one where k has the smallest value that is larger than t .
- In the `for` command k can also be assigned specific value (typed in as a vector). For example: `for k=[7 9 -1 3 3 5]`.
- The value of k should not be redefined within the loop.
- Each `for` command in a program must have an `end` command.

- The value of the loop index variable (k) is not displayed automatically. It is possible to display the value in each pass (sometimes useful for debugging) by typing k as one of the commands in the loop.
- When the loop ends, the loop index variable (k) has the value that was last assigned to it.

Example:

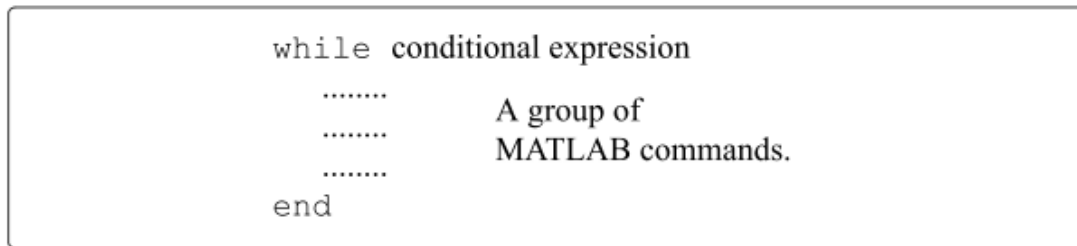
```
for k=1:3:10
    x = k^2
end
```

- When this program is executed, the loop is executed four times.
- The value of k in the four passes is $k=1, 4, 7$, and 10 , which means that the values that are assigned to x in each of the passes are $x=1, 16, 49$, and 100 , respectively.
- Since a semicolon is not typed at the end of the second line, the value of x is displayed in the Command Window at each pass.
- When the script file is executed, the display in the Command Window is:

```
>> x =
     1
x =
    16
x =
    49
x =
   100
```

while-end Loops

- `while-end` loops are used in situations when looping is needed but the number of passes is not known ahead of time.
- In `while-end` loops the number of passes is not specified when the looping process starts. Instead, the looping process continues until a stated condition is satisfied.

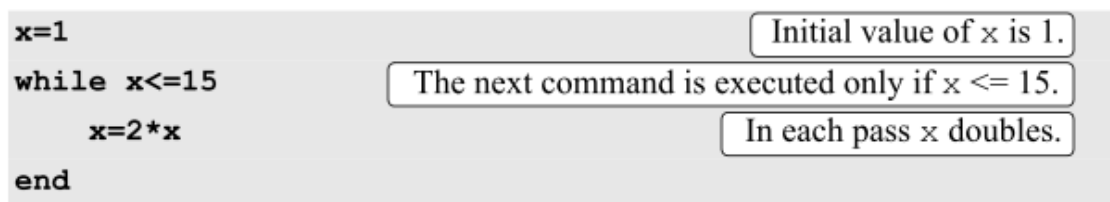


- ❖ The first line is a `while` statement that includes a conditional expression. When the program reaches this line the conditional expression is checked.
 - If it is false (0), MATLAB skips to the `end` statement and continues with the program.
 - If the conditional expression is true (1), MATLAB executes the group of commands that follow between the `while` and `end` command. Then MATLAB jumps back to the `while` command and checks the conditional expression.

This looping process continues until the conditional expression is false.

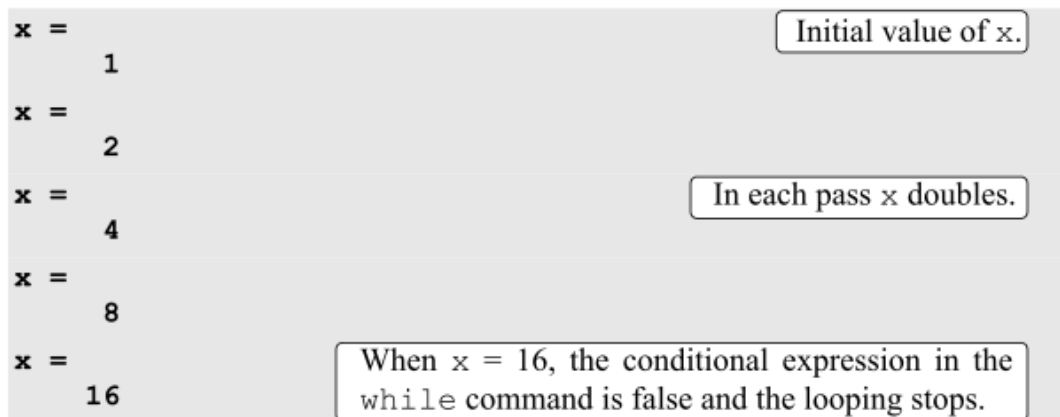
- The conditional expression in the `while` command must include at least one variable.
- At least one of the variables in the conditional expression must be assigned a new value in the commands that are between the `while` and the `end`. Otherwise, once the looping starts it will never stop since the conditional expression will remain true.

Example:



- In this program a variable `x` with an initial value of 1 is doubled in each pass as long as its value is equal to or smaller than 15.

When this program is executed the display in the Command Window is:



```
x =  
    1  
x =  
    2  
x =  
    4  
x =  
    8  
x =  
   16
```

Initial value of x.

In each pass x doubles.

When x = 16, the conditional expression in the while command is false and the looping stops.

- When writing a `while-end` loop, the programmer has to be sure that the variable (or variables) that are in the conditional expression and are assigned new values during the looping process will eventually be assigned values that make the conditional expression in the `while` command false. Otherwise the looping will continue indefinitely (indefinite loop).

For example, in the example above if the conditional expression is changed to be `x >= 0.5`, the looping will continue indefinitely. Such a situation can be avoided by counting the passes and stopping the looping if the number of passes exceeds some large value. This can be done by adding the maximum number of passes to the conditional expression, or by using the `break` command.