

MATLAB Introductory courses

Lecture 1

Siamak Ghorbani Faal
sghorbanifaal@wpi.edu



WPI
IT - Academic and Research Computing Center

Course outlines

- Lecture 1: Introduction to MATLAB
- Lecture 2: Using MATLAB
- Lecture 3: Visualization and Specialized Tools

Current lecture outlines

Getting Started

MATLAB help

Operators

Matrices

Operators

Common MATLAB functions

Flow control

Scripts

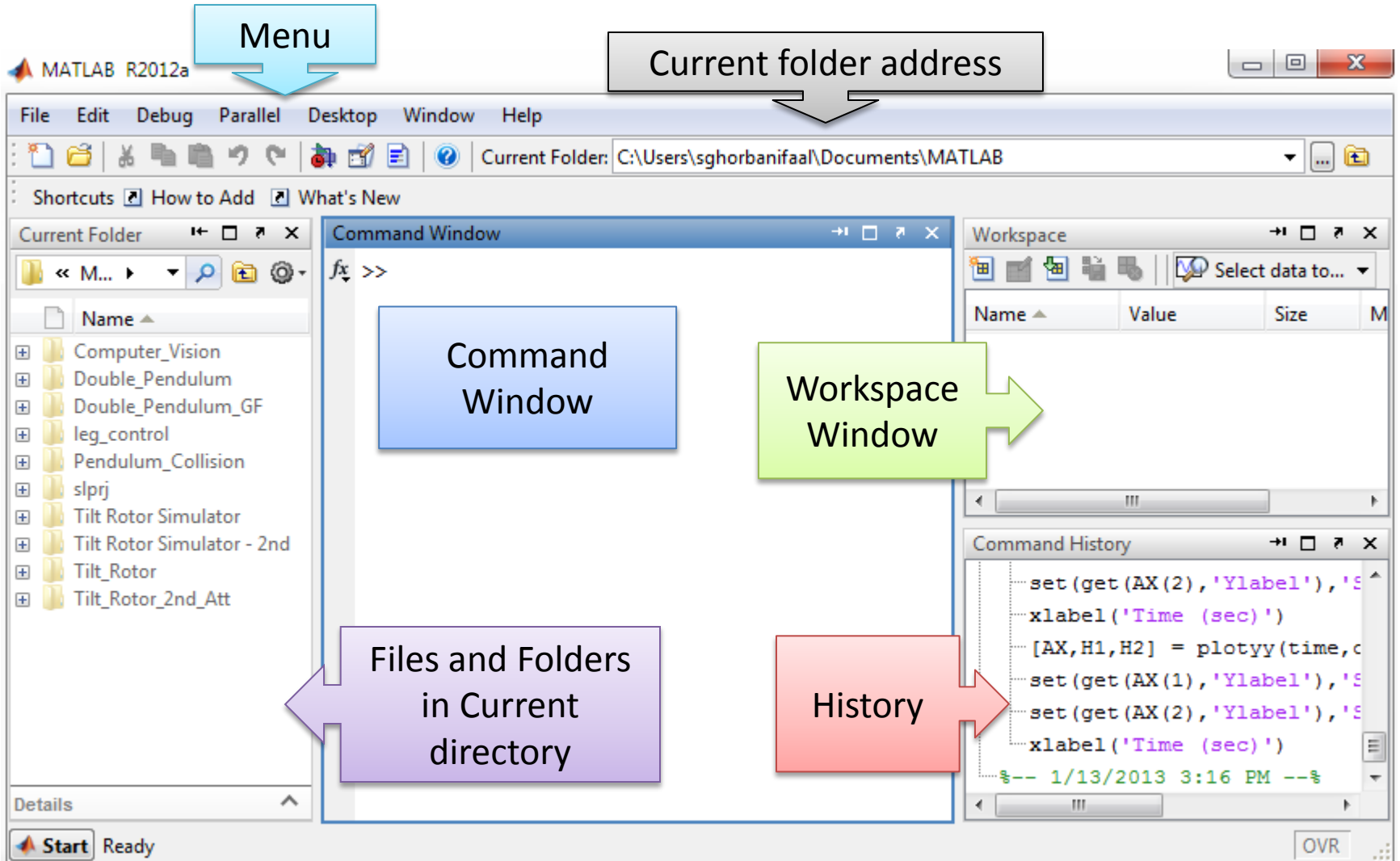
What is MATLAB



Applications

- Mathematics and computations
- Algorithm development
- Data acquisition
- Modeling, Simulation and prototyping
- Data analysis, exploration and visualization
- Scientific and engineering graphs
- Application development (GUI)
- Almost everything!

MATLAB Window



Before we start!

- Ctrl and Pause Emergency stop
- Ctrl and c Emergency stop
- clc Clear screen

MATLAB Help

- Help Menu → Product Help
- Type **helpdesk** or **doc** in **command window**
- Type: **help function** in **command window**
 - Example: >> **help sin**
- Type: **doc function** in **command window**
 - Example: >> **doc sin**
- Type **lookfor** keyword in **command window**
 - Example: >> **lookfor sin**

Predefined Values

- `eps` Epsilon
- `inf` Infinity
- `NaN` Not a Number
- `pi` π
- `i` $\sqrt{-1}$
- `j` Same as `i`

MATLAB Variables

- A MATLAB variable is essentially a **tag** that you assign to a **value in memory**.
- MATLAB does not require any type declarations or dimension statements.
- When MATLAB encounters a new variable name, it automatically creates the variable and allocates the predefined amount of storage.
- **If the variable already exists, MATLAB changes its contents.**
- Variable names consist of a letter, followed by any number of letters, digits, or underscores.
- MATLAB uses only the first 64 characters of a variable name.
- MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters.
- MATLAB stores variables in a part of memory called workspace.
- MATLAB provides three types of variables: Local, Global and persistent

MATLAB Variables

- Rules:
 - Make sure variable name is valid.
 - DON'T use function names as variable name.
 - Avoid using `i`, `j`, `pi`, `eps`, `inf` and `NaN` as variable names.
- Syntax:
 - `VariableName = Value;`
- Examples:

```
>> a = 5;
>> b = 7;
>> name = 'Max'
>> FirstLetter = 'S'
```

MATLAB Variables

- MATLAB automatically assigns `ans` to result of an operation.
 - `>> a = 5; b = 7;`
 - `>> c = a + b;`
 - `>> a*b;`
 - Clearing a variable:
 - `>> clear VariableName1 VariableName2`
 - `>> clear all`
- Or right click on specific variable in Workspace and select Delete

Formats

- Integers:

- MATLAB has 8, 16, 32 and 64 bit, signed and unsigned integer data types.

`int8, int16, int32, int64, uint8, uint16, uint32, uint64`

- Floating point numbers:

- Double precision (64 bits): `double`
- Single precision (32 bits): `single`

Imaginary numbers

- Use **i** or **j** to define imaginary numbers

```
>> a = 10 + 5i;
```

```
>> b = 6 + i*2;
```

```
>> c = 7 - 1i*7;
```

```
>> d = 5 + 1j;
```

```
>> e = 14*j;
```

Strings

```
>> Name = 'Max' ;
```

```
>> ErrorMessage = 'Divided by Zero' ;
```

Vectors

- Defining a row vector:
 - Syntax: `VectorName = [array of numbers or variables]`
 - Syntax: use `space ()` or `comma (,)` or their `combination` to indicate different entries
 - Example:

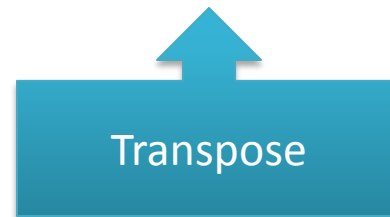
```
>> A = [1,2,3];  
          >> B = [1 5 6];  
          >> C = [0, 0, 10];
```
 - It is possible to include `imaginary numbers`, `pi`, `NaN`, `inf` and other constants in the vector:

```
>> E = [0, inf, NaN, pi, 1i]
```


Vectors

- Defining a column vector:
 - Syntax: `VectorName = [array of numbers or variables]`
 - Syntax: use semicolon (`;`) to indicate different entries
 - Example: `>> A = [1; 2; 3];`
 - It is also to use Transpose to define a column vector.

```
>> B = [4, 5, -3]';
```



Vectors

- Accessing to specific element in a Vector
 - Example:

```
>> A = [1 -15 2 9 8 0 0 5]
```

```
>> a = A(4)
```
- Accessing to range of elements in a Vector
 - Example:

```
>> b = A(2:5)
```
- create regularly spaced vector
 - Example:

```
>> time = 0:0.1:10
```

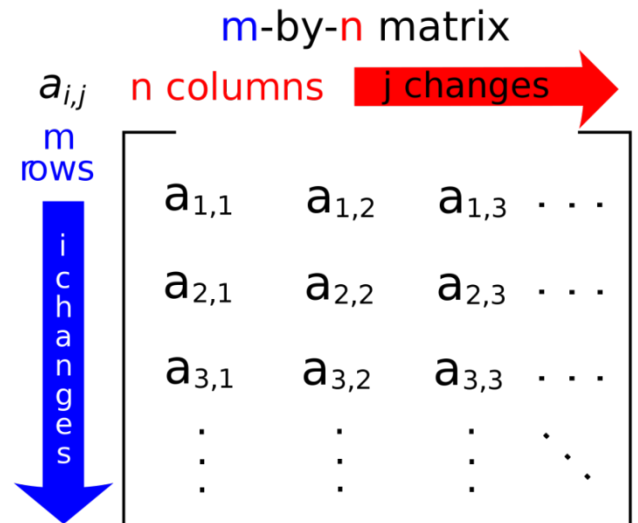
`:` is an operator

It means “all the way to” or “all the elements”



Matrices

- Defining a matrix
 - Use combination of commas (,) and semicolons (;) to define elements of each row and column.
 - Syntax: `MatrixName = [a1,1 , a1,2 , a1,3 ; a2,1 , a2,2 , a2,3 ; a3,1 , a3,2 , a3,3];`
 - Example: `>> A = [1, -5, 0 ; 6, 6, 2 ; -1, 9, 7];`



Matrices

- Accessing to specific element in a Vector

- Syntax: MatrixName(RowNumber, ColumnNumber)

- Example:

```
>> A = [1, 2, 3, 4 ; 5, 6, 7, 8 ; 9, 10, 11, 12]
```

```
>> a = A(2,3)
```

- Accessing to range of elements in a Vector

- Syntax: MatrixName(RowRange, ColumnRange)

```
>> b = A(2:3,2:4)
```

Operators

Operator	Description		
	Real Numbers	Complex Numbers	Matrices
+	Addition	Addition	Addition
−	Subtraction	Subtraction	Subtraction
*	Multiplication	Multiplication	Matrix Multiplication
/	Division: $a/b = a \div b$	Division: $a/b = a \div b$	$A/B = A \times B^{-1}$
\	Left division: $a \backslash b = b \div a$	Left division: $a \backslash b = b \div a$	$A \backslash B = A^{-1} \times B$
^	Power	Power	Power
'		Complex conjugate transpose	Transpose
()	Specify evaluation order	Specify evaluation order	Specify evaluation order
.			Convert to Element by element operation

Operators

- Real and Complex number operators examples:

Real numbers	Complex numbers
<pre>>> a = 9; >> b = 5; >> a+b >> a - b - 2 >> a*b >> a/(b^2) >> (a^b)/3 >> c = a^2 + b^2 >> d = c^0.5</pre>	<pre>>> a = 2 + 5i; >> b = 3; >> c = 4i; >> a + b + c >> a - b - c >> (a^2 + b^2)/c >> a/b/c >> a - b/c >> c = a*b</pre>

MATLAB does not give any warning for this operation

Operators

- Vectors and matrix operator examples:

```
>> H1 = [1, 2, 3];
```

```
>> H2 = [2, 0, 1];
```

```
>> H1 + H2
```

```
>> H1 - H2
```

```
>> H1 * H2
```

This operation will generate an error

```
>> H1 .* H2
```

```
>> H1 .^2
```

```
>> H1 ./ H2
```



• Converts the operation to Element-by-Element operation

Operators

- Vectors and matrix operator examples:

```
>> V1 = [1; 5; 0];
```

```
>> V2 = [3; -1; 9];
```

```
>> H1 = [2, 1, 3];
```

```
>> V1 + V2
```

```
>> V1.*V2
```

```
>> V1.^2
```

```
>> V1./V2
```

```
>> H1*V1
```



Dot product!

Operators

- Vectors and matrix operator examples:

```
>> A = [5, 2, 0; 3, 2, 1; 4, 3, 1];
```

```
>> B = [2, 1, 1; 2, 3, 5; 4, 0, 1];
```

```
>> V = [2; 2; 0];
```

```
>> H = [1, 3, 1];
```

```
>> A*V
```

```
>> H*A
```

```
>> A^2
```

```
>> A.^2
```

```
>> A/B      =  $A \cdot B^{-1}$ 
```

```
>> A\B      =  $A^{-1} \cdot B$ 
```

```
>> A(1, :) * B(:, 2)
```

```
>> A(:, 3) * B(2, :)
```

Relational Operators

- Relational operators perform element-by-element comparisons between two arrays.
- Operation results in Logical array with elements set to logical 1 (true) or to logical 0 (false)

Operator	Description	Syntax
<	Less than	A < B
<=	Less than or equal	A <= B
>	Greater than	A > B
>=	Greater than or equal	A >= B
==	Equal to	A == B
~=	Not equal to	A ~= B

Relational Operators

- Examples:

```
>> a = 3; b = 5; c = 2;
```

```
>> A = [1, 5, 1, 6, 7];
```

```
>> B = [2, 5, 1, 2, 0];
```

```
>> a < b
```

```
>> c > b
```

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

```
>> b == a + c
```

```
>> b - 2 == a
```

```
>> A == B
```

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

```
>> A == b
```

Logical Operators

- Logical operator work element by element on arrays
- Numbers 0 and 1 represent logical 0 and 1, respectively.

Operator	Description	Syntax
~	NOT	~A
&	Elementwise AND	A & B
	Elementwise OR	A B

Logical Operators

- Examples:

```
>> A = [1 1 0 0 1 0 1 1];
```

```
>> B = [1 0 1 1 0 0 1 1];
```


```
>> ~A
```

```
>> A & B
```

```
>> A | B
```

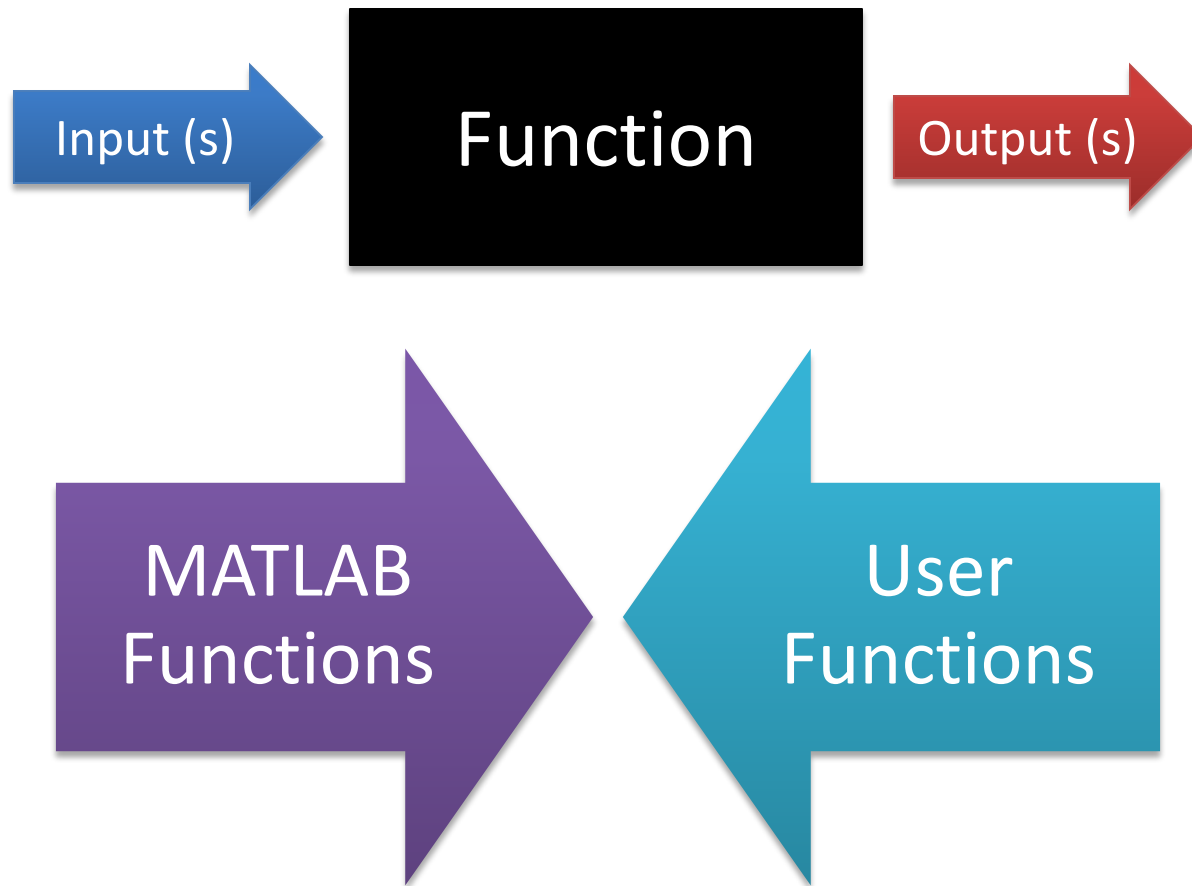
```
>> ~A | ~B
```

```
>> (A & ~B) | (~A & B)
```



XOR operator, you may also use
xor function: xor(A, B)

Functions



Functions

- List of elementary mathematical functions:

```
>> help elfun
```

- List of elementary matrix functions:

```
>> help elmat
```

- Specialized math functions:

```
>> help specfun
```

Common Functions: Numbers

- `TF = isnan(A)`
 `>> isnan (454)`
 `>> isnan (NaN)`
- `TF = isfinite(A)`
 `>> isfinite(100)`
 `>> isfinite(inf)`
- `TF = isinf(A)`
 `>> isinf(50)`
 `>> isinf(inf)`
- `TF = isreal(A)`
 `>> isreal (56)`
 `>> isreal (4 + 5i)`

Common Functions: Class

- **K = isa(obj, 'class_name')**

```
>> a = uint8(10)
>> isa(a, 'uint8')
>> isa(a, 'double')
```

- **A = exist('name','kind')**

```
>> my_number = 95;
>> exist('my_number', 'var')
```

Common Functions: Strings

- `combinedStr = strcat(s1, s2, ..., sN)`
`>> Sentence = strcat(' Hello', ' World', ' !')`
- `x = str2num('str')`
`>> a = str2num('123.3')`
- `str = num2str(A)`
`>> b = num2str(pi)`
- `str = num2str(A,precision)`
`>> c = num2str(pi,3)`

Common Functions: Trigonometric

`sin, sind, sinh`
`cos, cosd, cosh`
`tan, tand, tanh`
`cot, cotd, coth`

`sec, secd, sech,`
`csc, cscd, csch,`

`asin, asind, asinh`
`acos, acosd, acosh`
`atan, atand, atanh, atan2`
`acot, acotd, acoth,`

`asec, asecd, asech`
`acsc, acscd, acsch`

Common functions: Exponential

- `exp` - Exponential.
- `log` - Natural logarithm.
- `log10` - Common (base 10) logarithm.
- `Sqrt` - Square root.
- `nthroot` - Real n^{th} root of real numbers.

Common Functions: Complex

- `abs` - Absolute value.
- `angle` - Phase angle.
- `complex` - Construct complex data from real and imaginary parts.
`>> complex(1, 6)`
- `imag` - Complex imaginary part.
- `real` - Complex real part.

Common Functions: Rounding and remainder

- **fix** - Round towards zero.

```
>> fix(5.9)  
>> fix(-5.9)
```
- **floor** - Round towards minus infinity.

```
>> floor(4.7)  
>> floor(-4.7)
```
- **ceil** - Round towards plus infinity.
- **round** - Round towards nearest integer.
- **mod** - Modulus (signed remainder after division).

```
>> mod(13, 5)
```
- **rem** - Remainder after division.

```
>> rem(15, 5)
```
- **sign** - Signum.

Common Functions: Elementary matrices

- **zeros** - Zeros array.

```
>> zeros(1,10)  
>> zeros(4,4)
```
- **ones** - Ones array.

```
>> ones(5,1)  
>> ones(6,7)
```
- **eye** - Identity matrix.

```
>> eye(4)  
>> eye(3)
```
- **rand** - random matrix.

```
>> rand()  
>> rand(2,5)
```

Common Functions: Basic array information

- `size`
 - Size of array.
- `length`
 - Length of vector.
- `disp`
 - Display matrix or text.
- `isempty`
 - True for empty array.

```
>> a = [1, 2, 3]
>> b = []
>> isempty(a)
>> isempty(b)
```
- `isequal`
 - True if arrays are numerically equal.

```
>> a = [1, 5, 2]
>> b = a
>> isequal(a,b)
>> b(2) = 4
>> isequal(a,b)
```
- `isequaln`
 - True if arrays are numerically equal.
- `isequalwithequalnans`
 - True if arrays are numerically equal.

Common Functions: Array utility functions

- `isscalar` - True for scalar.
- `isvector` - True for vector.
- `isrow` - True for row vector.
- `iscolumn` - True for column vector.
- `ismatrix` - True for matrix.

Common Functions: Matrices

- `det` - Determinant
- `trace` - Sum of diagonal elements
- `linsolve` - Solve linear systems of equations (using LU factorization)
- `eig` - Find eigenvalues and eigenvectors
- `svd` - Singular value decomposition
- `balance` - Improve accuracy of computed eigenvalues
- `cond` - condition number
(Ratio of the largest singular value to the smallest)

Common Functions: Vectors

- Dot product
- Cross product

```
>> a = [1, 4, 1]
>> b = [2, 1, -1]
>> dot(a,b)
>> cross(a,b)
```

Function call

- It is important how you call a function:

- Example: `eig` function

```
>> doc eig
```

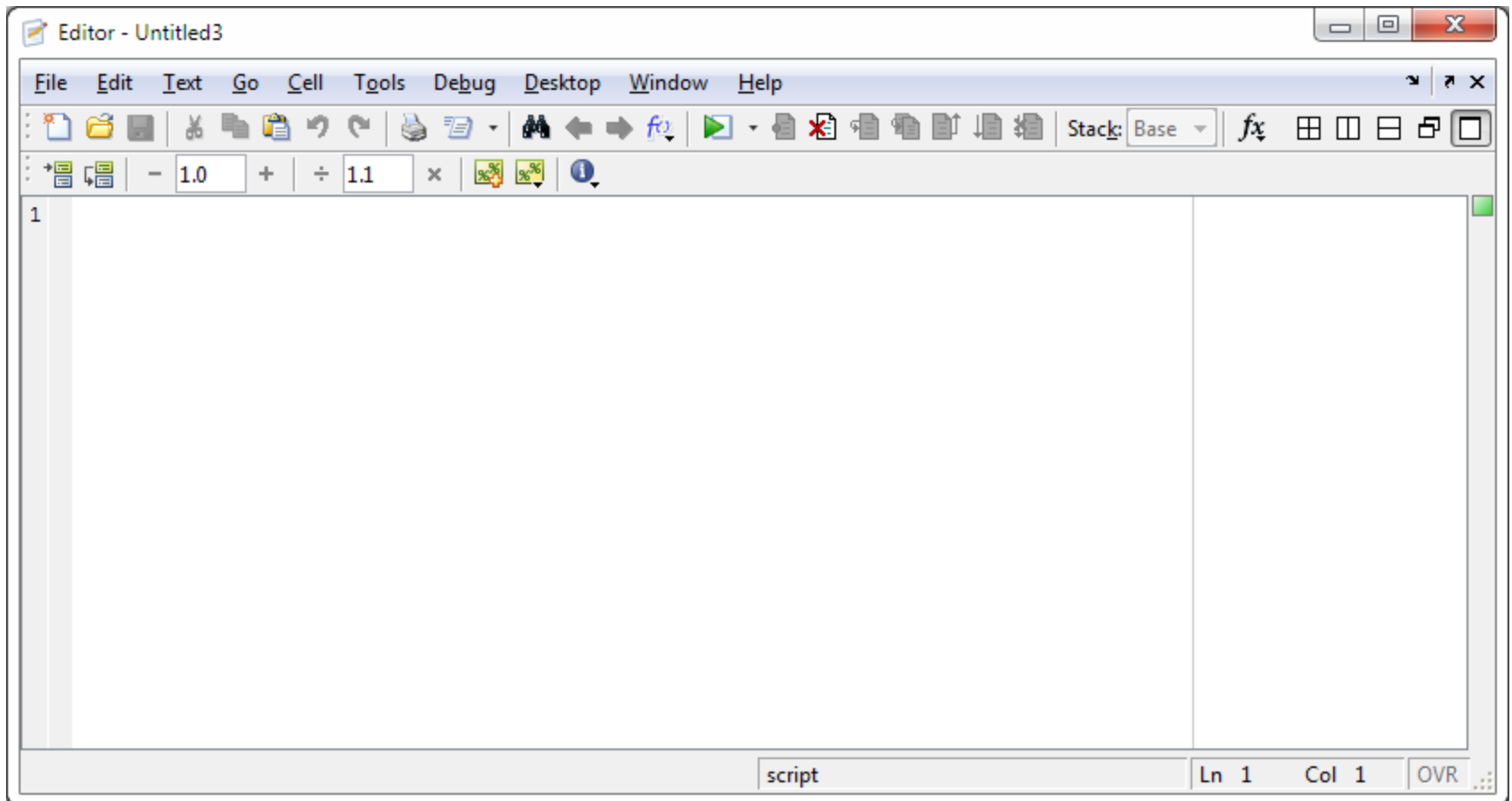
```
>> A = 10*rand(3,3)
```

```
>> d = eig(A)
```

```
>> [V,D] = eig(A)
```

Scripts

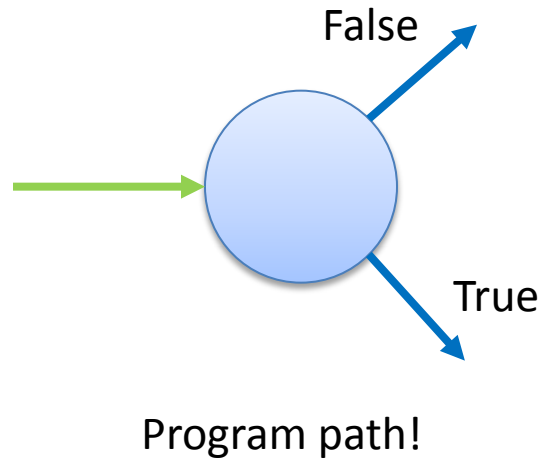
- Writing your code in file!



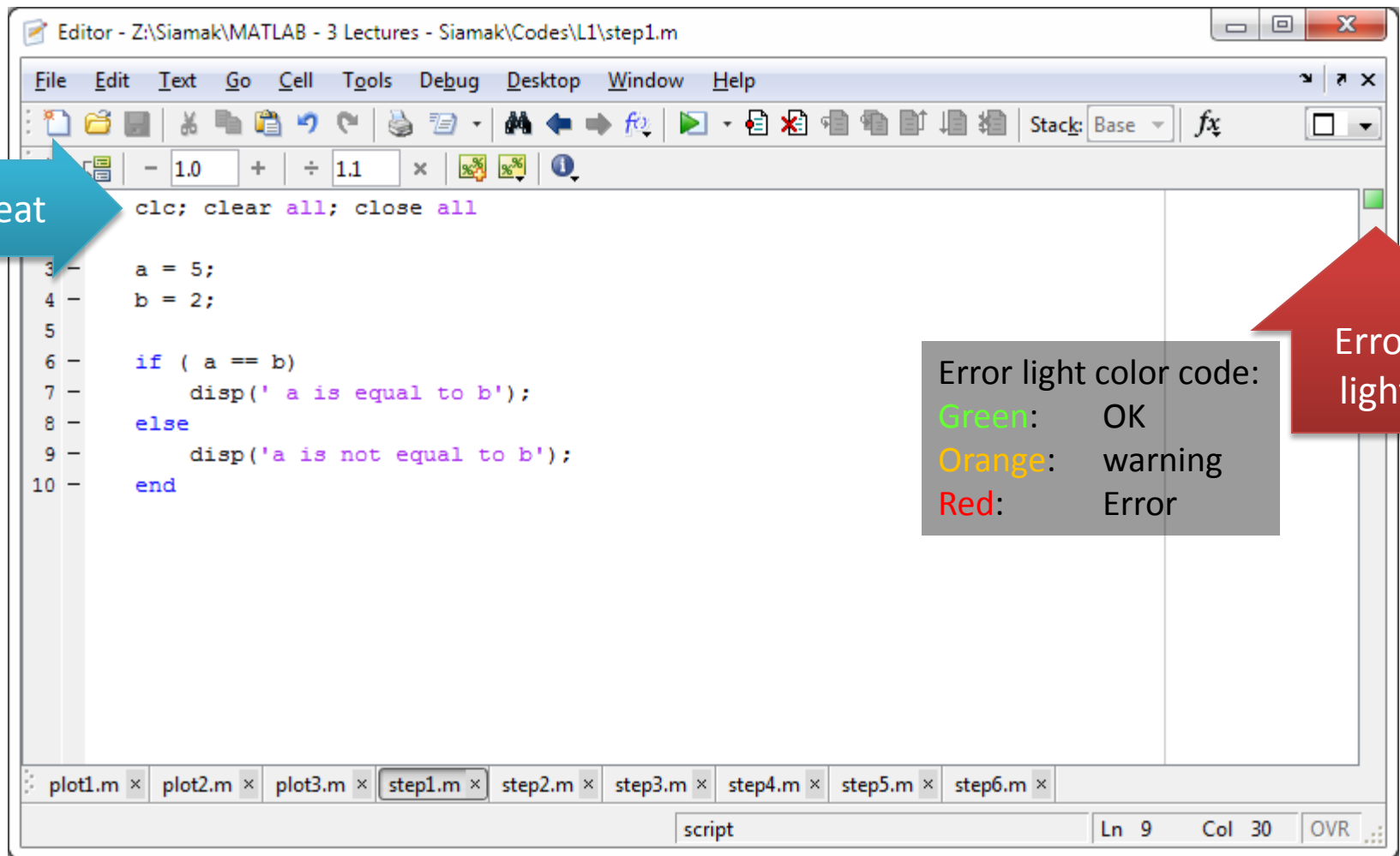
Flow control

- IF, ELSE and ELSEIF

```
if (This condition is true)  
    Execute this part  
else  
    Execute this part  
end
```



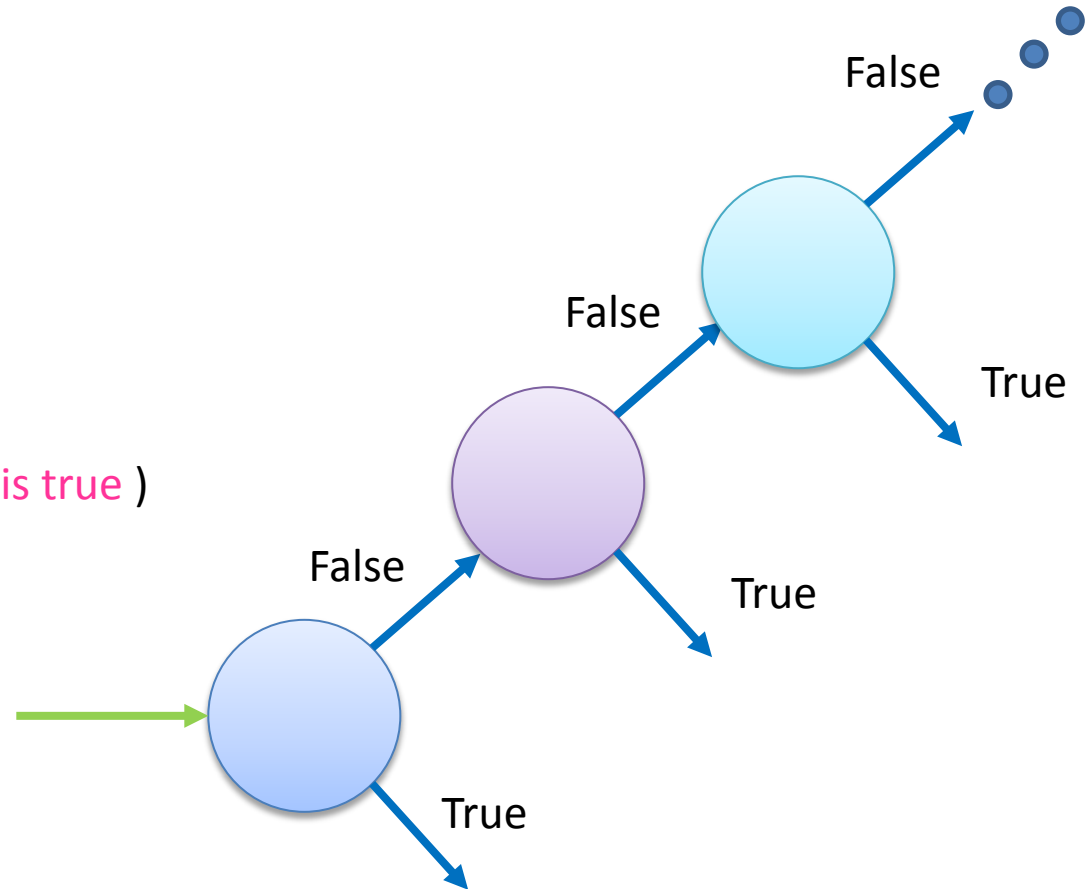
Be neat



Flow control

- IF, ELSE and ELSEIF

```
if (This condition is true)  
    Execute this part  
elseif ( Otherwise, if this condition is true )  
    Execute this part  
else  
    Execute this part  
end
```



Program path!

Editor - Z:\Siamak\MATLAB - 3 Lectures - Siamak\Codes\L1\step2.m

File Edit Text Go Cell Tools Debug Desktop Window Help

Stack: Base fx

```
1 - clc; clear all; close all
2
3 - a = input('Enter a: ');
4 - b = input('Enter b: ');
5
6 - if ( a == b)
7 -     disp(' a is equal to b');
8 - elseif (a > b)
9 -     disp(' a is greater than b');
10 - else
11 -     disp(' a is smaller than b');
12 - end
```

script Ln 12 Col 4 OVR

Flow control

- **FOR**

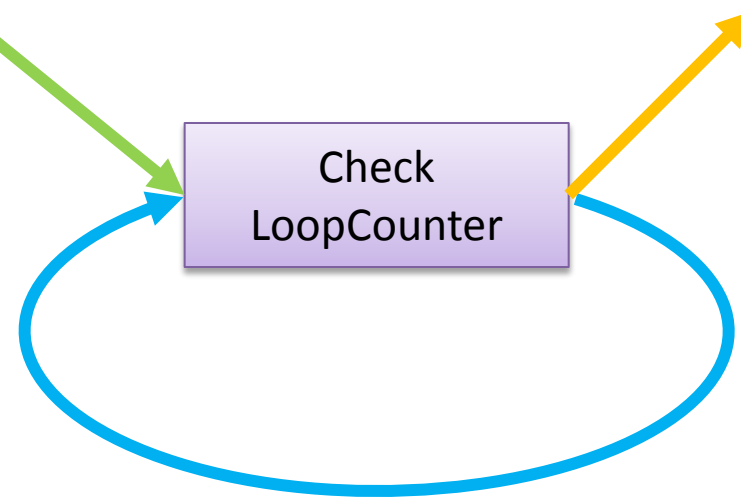
```
for (LC = IN : Step : DesireValue)
```

Execute this part

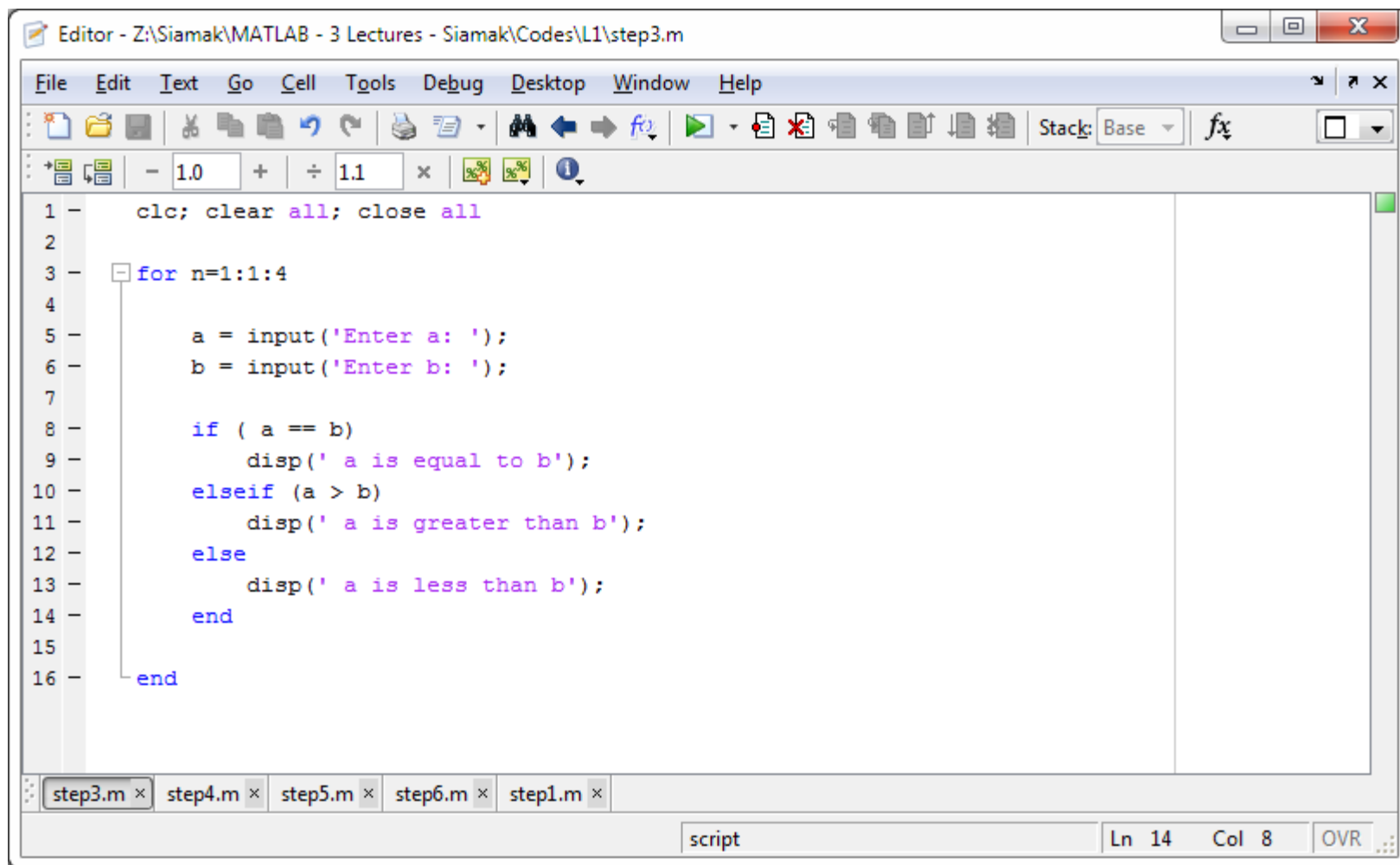
```
end
```

Set LoopCounter to
an InitialNumber

If it is larger than
the DesireValue



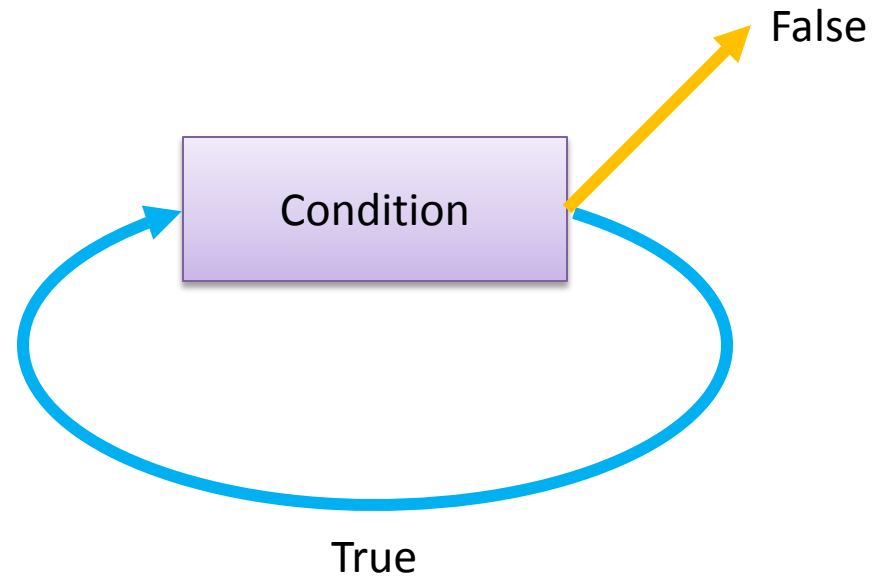
If it is less than the desired value:
 $\text{LoopCounter} = \text{LoopCounter} + \text{Step}$



Flow control

- While

```
while (Condition is valid)  
    Execute this part  
end
```



Editor - Z:\Siamak\MATLAB - 3 Lectures - Siamak\Codes\L1\step4.m

File Edit Text Go Cell Tools Debug Desktop Window Help

Stack: Base fx

```
1 - clc; clear all; close all
2
3 - e = 0;
4
5 - while (e == 0)
6
7 -     a = input('Enter a: ');
8 -     b = input('Enter b: ');
9
10 -    if ( a == b)
11 -        disp(' a is equal to b');
12 -    elseif (a > b)
13 -        disp(' a is greater than b');
14 -    else
15 -        disp(' a is less than b');
16 -        e = 1;
17 -    end
18
19 - end
```

script Ln 16 Col 15 OVR

Flow control

- `continue`
 - Pass control to next iteration of for or while loop
- `break`
 - Terminate execution of for or while loop
- `return`
 - Return to invoking function

Editor - Z:\Siamak\MATLAB - 3 Lectures - Siamak\Codes\L1\step5.m

File Edit Text Go Cell Tools Debug Desktop Window Help

Stack: Base fx

```
1 -   clc; clear all; close all
2
3 -   while (1)
4
5       a = input('Enter a: ');
6       b = input('Enter b: ');
7
8       if ( a == b)
9           disp(' a is equal to b');
10      elseif (a > b)
11          disp(' a is greater than b');
12      else
13          disp(' a is less than b');
14          break
15      end
16
17   end
18
19   disp('Execution completed successfully')
```

script Ln 16 Col 5 OVR

Editor - Z:\Siamak\MATLAB - 3 Lectures - Siamak\Codes\L1\step6.m

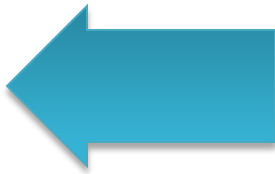
File Edit Text Go Cell Tools Debug Desktop Window Help

Stack: Base fx

```
1 - clc; clear all; close all
2
3 - while (1)
4
5 -     a = input('Enter a: ');
6 -     b = input('Enter b: ');
7
8 -     if ( a == b)
9 -         disp(' a is equal to b');
10 -    elseif (a > b)
11 -        disp(' a is greater than b');
12 -    else
13 -        disp(' a is less than b');
14 -        return
15 -    end
16
17 - end
18
19 - disp('Execution completed su
```

⚠ Line 19: This statement (and possibly following ones) cannot be reached. Details

script Ln 17 Col 4 OVR



Flow control

- SWITCH, CASE, OTHERWISE

switch (Variable)

case 1

Execute this part

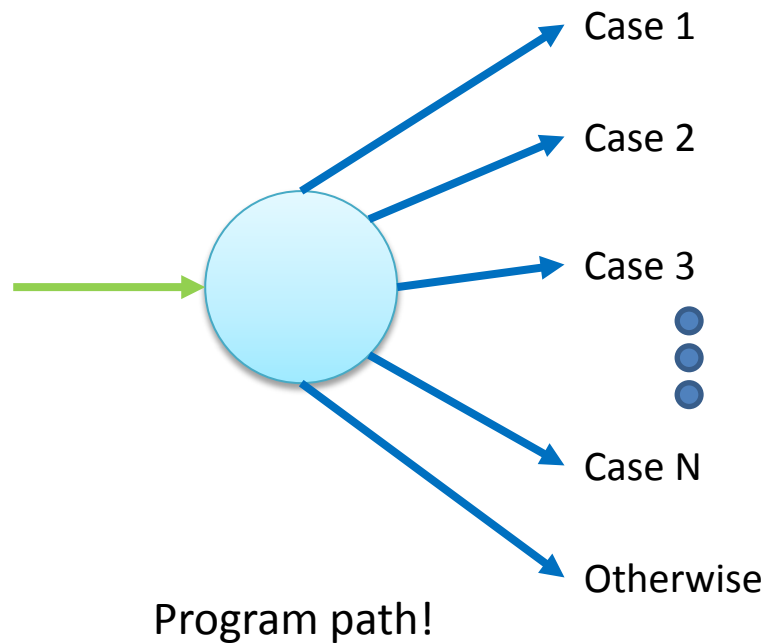
case 2

Execute this part

otherwise

Execute this part

end





```
Enter a: 9
Enter operator: '^'
Enter b: 2
c =
    81
```

```
Enter a: 8
Enter operator: '+'
Enter b: 7
c =
    15
```

```
Enter a: 6
Enter operator: '&'
Enter b: 2
Unknown operator -> Execution will be terminated.
fx >>
```

script