# Introduction to MATLAB

## 4: Programming

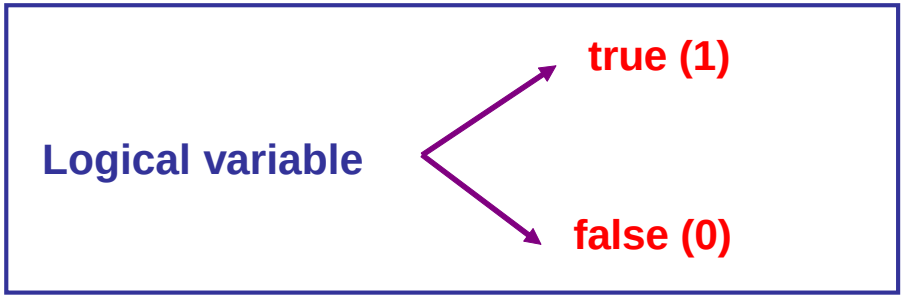
**Georgios Georgiou**
*Department of Mathematics and Statistics*
*University of Cyprus*

Πανεπιστήμιο Κύπρου
Τμήμα Μαθηματικών
και Στατιστικής

# Flow control structures

- for loops
- while loops
- if statement
- switch statement

# Logical variables




```
>> x=1
x =
     1
>> y= 1>2
y =
     0
>> z=logical(0)
z =
     0
>> A= ones(2)==eye(2)
A =
     1     0
     0     1

>> whos
  Name      Size            Bytes  Class     Attributes

  A         2x2                 4  logical
  x         1x1                 8  double
  y         1x1                 1  logical
  z         1x1                 1  logical
```

# Relational operators

| | |
|---|---|
| < | less than |
| > | greater than |
| < = | less than or equal |
| > = | greater than or equal |
| = = | equal |
| ~ = | not equal |

```
>> 1+1==2
ans =
     1
>> 3>2
ans =
     1
>> 4<=3
ans =
     0
>> [1 2 3 4] <= [ 3 -2 1 0]
ans =
     1     0     0     0
>> [1 2 3 4] ~= [ 3 -2 1 0]
ans =
     1     1     1     1
```

# Comparing matrices

```
>> A=rand(3)
A =
    0.4218    0.9595    0.8491
    0.9157    0.6557    0.9340
    0.7922    0.0357    0.6787
>> B=rand(3)
B =
    0.7577    0.6555    0.0318
    0.7431    0.1712    0.2769
    0.3922    0.7060    0.0462
>> C= A == B
C =
     0     0     0
     0     0     0
     0     0     0
>> D= A>B
D =
     0     1     1
     1     1     1
     1     0     1
>>
```

**The resulting matrices are logical matrices!**

# Relational functions

**Relational functions are alternatives to relational operators.**

**For example, A==B and eq(A,B) are equivalent.**

| | | | |
|---|---|---|---|
| **lt** | < | less than | μικρότερος |
| **gt** | > | greater than | μεγαλύτερος |
| **le** | < = | less than or equal | μικρότερος ή ίσος |
| **ge** | > = | greater than or equal | μεγαλύτερος ή ίσος |
| **eq** | = = | equal | ίσος |
| **ne** | ~ = | not equal | άνισος |

# Equivalent expressions

| | |
|---|---|
| x==y | eq(x,y) |
| 1>= 5 | ge(1,5) |
| A~=B | ne(A,B) |
| (a+b) < c/d | lt(a=b, c/d) |

# is functions

ischar
isempty
isequal
isfinite
isinf
isinteger
iskeyword
isletter
islogical
isnan
isreal
isscalar
issorted
isvarname
isvector

# is functions

```
>> A=[ 1 2 inf; nan -inf 0.5; nan inf 1]
A =
    1.0000    2.0000       Inf
       NaN      -Inf    0.5000
       NaN       Inf    1.0000
>> isreal(A)
ans =
     1
>> isfinite(A)
ans =
     1     1     0
     0     0     1
     0     0     1
>> isinf(A)
ans =
     0     0     1
     0     1     0
     0     1     0
>> isnan(A)
ans =
     0     0     0
     1     0     0
     1     0     0
```

# A useful function: find

**The most important logical function is find.**

**For example for**

```
>> u=[1 -3 4 0 -2 5 3 2 -4 6]
u =
     1    -3     4     0    -2     5     3     2    -
4     6
```

**we get**

```
>> y=find(u>=1)
y =
     1     3     6     7     8    10
```

# find

```
>> A=rand(4,5)
A =
    0.4387    0.1869    0.7094    0.6551    0.9597
    0.3816    0.4898    0.7547    0.1626    0.3404
    0.7655    0.4456    0.2760    0.1190    0.5853
    0.7952    0.6463    0.6797    0.4984    0.2238
>> A( find(A<0.5) )= 0
A =
         0         0    0.7094    0.6551    0.9597
         0         0    0.7547         0         0
    0.7655         0         0         0    0.5853
    0.7952    0.6463    0.6797         0         0
>>
```

# Logical operators

| Symbol | Function |
|---|---|
| `&` | and |
| `\|` | or |
| `~` | not |
| `xor` | xor |
| `all` | all |
| `any` | any |
| `&&` | relop |
| `II` | relop |

**Equivalent expressions**:

- **and(p,q)** and **p & q**.
- **or(p,q)** and **p | q**.
- **not(p)** **and ~ p**.

# Exclusive or

| x | y | and(x,y) | or(x,y) | xor(x,y) |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 |

# Functions all and any

# for loops

```
for  index = initial value (: step) : final value
            statements
end
```

# Example

$$\sum_{k=1}^{1}(k+1)^k$$

**FORTRAN LOGIC!**

```
sum1=0;
for i=1:10
    sum1=sum1+(i+1)^i;
 end
sum1
```

**MATLAB LOGIC!**

```
sum1=sum(((1:10)+1).^(1:10))
```

# Example

$$\prod_{i=1, i\neq 5}^{11} (x_5 - x_i)$$

**NESTED FOR LOOPS**

```
x=(0:10)/10;
product=1;

for i=1:4, product=product*(x(5)-x(i)); end
for i=6:11, product=product*(x(5)-x(i)); end

 product
```

**NESTED FOR-IF LOOPS**

```
x=(0:10)/10;
product=1;
for i=1:11
      if i~=5
        product=product*(x(5)-x(i));
      end
 end
 product
```

# Nested for loops

```
for i=1:m
for j=1:n
        statements with i and j
end
end
```

# Example

$$A = \left( \frac{1}{i + j - 1} \right)_{m \times n}$$

# while loops

```
while relation
        statements
end
```

```
function [n] = xlgmin(x)
% function [n] = xlgmin(x)
%
% Finds the smallest positive integer  n such that
%            log n >= x
%
n = 1 ;
while log(n) < x
   n = n+1;
end

% End of  xlgmin.m
```

# Example

```
function [S] = exp1(x)

% function [S] = exp1(x)
%
% Calculates exp(x) with an accuracy of  0.0001
n = 1;
S = 1;
an = 1;
while abs(an) >= 0.0001
    an = x.^n/factorial(n);
    S = S + an;
    n = n + 1;
end
```

# if statement

```
if relation_1
    statement(s)
elseif relation_2
    statement(s)
        .
        .
else
    statement(s)
end
```

```
if relation_1
    statement(s)
else
    statement(s)
end
```

```
if relation_1
    statement(s)
end
```

# Example

$$g(x) = \begin{cases} x^2, & x \le 0.5 \\ 1/4, & x > 0.5 \end{cases}$$

```
function [G] = gee(x)
% GEE
if x <= 0.5
   G = x^2;
else
   G = 0.25;
end
% end of gee.m
```

```
function [G] = gee(x)
% GEE
G=x.^2;
G(find(x>0.5))=0.5;
% end of gee.m
```

# Useful functions

- break
- continue
- error

***Thank you!!***