CE300
MATLAB PROGRAMMING
WRITING SCRIPTS AND FUNCTIONS

By Şevki ÖZTÜRK

OBJECTIVES

- DEFINING GLOBAL VARIABLES
- STRUCTS
- WRITING SCRIPTS
- WRITING FUNCTIONS

DEFINING GLOBAL VARIABLES

- Each MATLAB function has its own local variables, which are separate from those of other functions, and from those of the base workspace.

- A variable can be shared in different functions by declaring the variable as global.

DEFINING GLOBAL VARIABLES

- global *variable*:defines variable as global.
- whos global: to see global variables
- clear global *variable*
- isglobal(variable): to see whether the variable is global or not. 1:global 0:not

DEFINING GLOBAL VARIABLES

Try it;

```
>> global GRAVITY
>> GRAVITY=9.81
>> GRAVITY*10
```

Try it;

```
>> global A B
>> A=5
>> B=10
>> C=3
>>whos
>>whos global
>>clear g
>>whos g
```

DEFINING GLOBAL VARIABLES

Try it;

```
>> global TICTOC
>> TICTOC=clock
>> t=etime(clock,TICTOC)
>>t
>> t=etime(clock,TICTOC)
```

- In MATLAB a function is defined called “tic” to measure the elapsed time btw tic and toc.

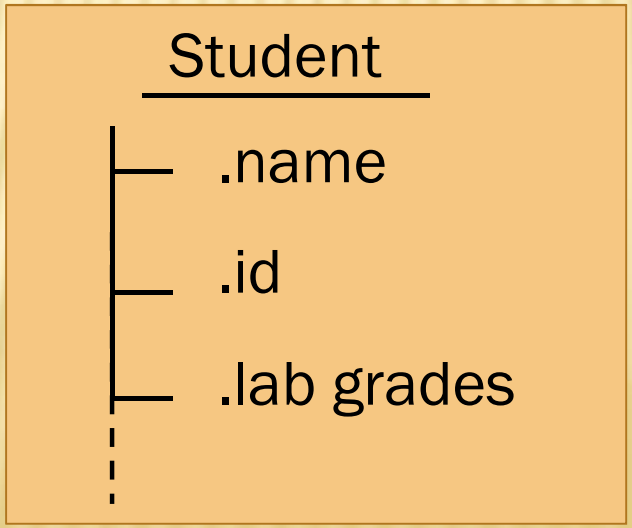
Try it;

```
>> tic
>> toc
```

STRUCTS

- are **array-oriented** data constructs
- group related **data of diffetent types**
- can contain **any kind of data** (text, a scalar, matrix,...)

Ex:



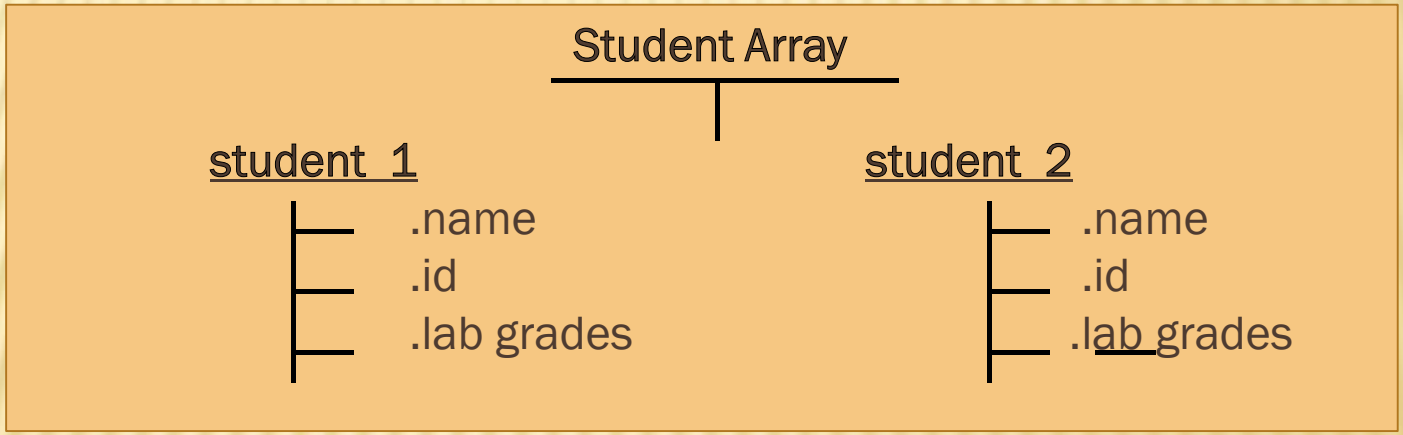
STRUCTS

Try it;

```
>> student.name='Fercan Orhan';
>> student.id= 1622513;
>> student.labgrades=[84,100,70,96,100];
>> student
>> student.name
>> student.labgrades
>> student.labgrades(2)
```

STRUCTS

- By adding subscripts after structure name, you can build a structure array; group related data of different types



STRUCTS

Try it;

```
>> student(2).name='zeki seckin demiral';
>> student(2).id= 1493089;
>> student(2).labgrades=[89,90,97,100,100];
>> student
>> student(1).name
>> student(2).labgrades
```

STRUCTS

Try it;

```
>> student(:).name
>> a=student(:).name
>>whos
>> a={student(:).name}
>>whos
```

Also try;

```
>> subplot(1,2,1);
>> bar(student(1).labgrades)
>> subplot(1,2,2);
>> bar(student(2
```

STRUCTS

- You can create a structure array with struct function:

  s = struct('field1', values1, 'field2', values2, ...)

Try it;

```
>> student=struct('name',{'fercan orhan', 'zeki seckin demiral'},...
'id',{1622513,1493089},...
'labgrades',{[84,100,70,96,100],[89,90,97,100,100]})
```

STRUCTS

- fieldname : to return fieldnames for structure array,.

```
>>fnames1=fieldnames(student(1))
```

- rmfield : to remove a given field.

```
>>rmfield(student, 'name')
```

WRITING SCRIPTS

scripts;

- do **not accept input** arguments or **return output** arguments,
- any variables that they create **remain in workspace** and can be used in computations,
- can operate on existing data in the workspace, or they can create new data on which to operate.
- are useful for computations you have to **perform repeatedly** from the command line

WRITING SCRIPTS

- In command line typing;

  edit *scriptname:* creates a script file named as edit.m

  sciptname: runs the script

Try it;

```
>> edit squares
```

In editor;

```
n=1:10
n.^2
```

In command window;

```
>> squares
```

WRITING SCRIPTS

Write the following in editor;

```
grade=input('Write the value of grade:');
%Take the written value as grade%
if (grade>=90)     lettergrade= 'A'
%Condition for lettergrade of A%
elseif grade<90 && grade>=80) lettergrade= 'B'
%Condition for lettergrade of B%
else  lettergrade= 'C'
%Condition for lettergrade of C%
end
```

Save as letter.m

Press F5 or use  button and Go to the command window.

or write *letter* in command window.

WRITING SCRIPTS

```
k=5;
matrix=zeros(k,k);
for m=1:k
for n=1:k
matrix(m,n)=m*n;
end
end
matrix
```

WRITING FUNCTIONS

functions;

- accept input arguments and return output arguments,
- define MATLAB functions in a file that begins with a line containing the function keyword
- operate on variables within their own workspace
- each function in a file has an area of memory, separate from the MATLAB base workspace, in which it operates.

WRITING FUNCTIONS

Types of functions;

- **Anonymous function:** single Matlab expression

Ex:

```
>> square=@(x) x.^2
```

- **Primary functions:** not anonymous, defined within a file
- **Nested functions:** functions defined within a function body

```
a=2;
b=3;
c=func1(a,b)
```

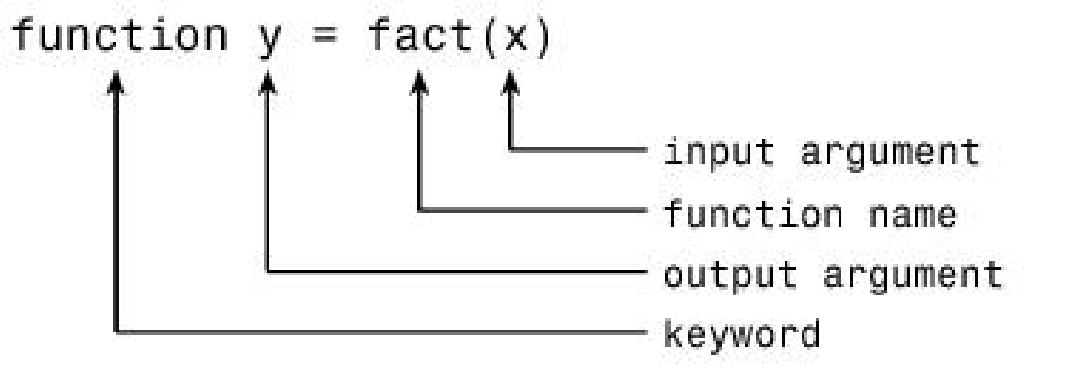
```
function x=func1(x1,x2)
x3=x1+x2;
x=func2(x3);
    function y=func2(x4);
        y=2*x4;
    end
end
```

func2 is nested in funct1.

WRITING FUNCTIONS

- Function definition:

  function [output variables]=name(input variables)



WRITING FUNCTIONS

- In editor write the following;

```
function y = square(x)
y = x*x;
end
```

- save it as square.m

Note: try to make filename same with the function name.

WRITING FUNCTIONS

- **Calling function:** call by the name of .M file

  Call the written function by typing (in command window);

```
>> square(2)
```

- **Information returned:** Output variables

```
>> square(2)

ans =

     4
```

WRITING FUNCTIONS

Comments: write comments using percent character (%).

comments before function in order to give a brief summary about the function.

comments within the function to explain function steps.

- help *functionname*: displays the comments written before function
- type *functionname:* displays the whole function with all comments

- Write the following comments in square.m

```
%This is a simple function
%calculating the square of
%a given number
function y = square(x)%adad
y = x*x; %Take the square of the in
end
```

WRITING FUNCTIONS

- Type **help square** and **type square** in command window.

```
>> help square
 This is a simple function
 calculating the square of
 a given number
```

```
>> type square

%This is a simple function
%calculating the square of
%a given number
function y = square(x)%adad
y = x*x; %Take the square of the input
end
```

WRITING FUNCTIONS

- **Multiple outputs:**

  function *[output1,output2]=functionname (input)*

  Ex: a function giving moment ans shear values for a given load is;

  function *[moment,shear]=f1 (load)*

- **Multiple inputs:**

  function *(output)=name (input1,input2)*

  Ex: a function giving moment for a given load and load position is;

  function *(moment)=f2 (load,loadposition)*

WRITING FUNCTIONS

Some build-in functions

round(x)

max(x)

min(x)

sum(x)

find(x) Ex:find(y>3), find(A==0)

...etc

WRITING FUNCTIONS

Some commands for plots of functions

- fplot *(@funtionname,[x1,x2]):*plots function btw x1 and x2
- fminsearch*(@functionname,x0):* starts at the point x0 and returns a value x that is a local minimizer of the function described in function.
- quad*(@functionname,a,b):* numerical integration of function between a and b.
- fzero*(@functionname,x0):* tries to find a zero of fun near x0.

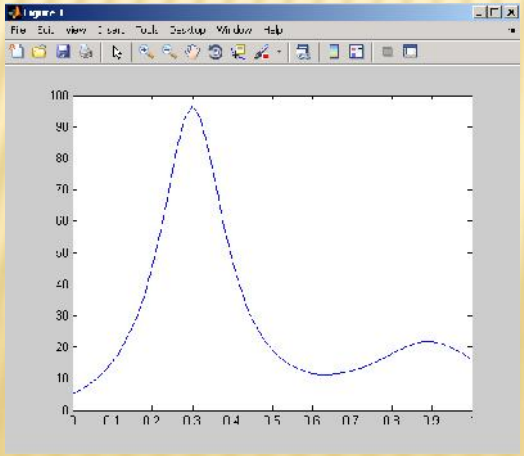
WRITING FUNCTIONS

```
function y=example(x)
y=1./((x-.3).^2+.01)+1./((x-.9).^2+.04)-6;
end
```

```
>> x=0:.02:1;
>> y=example(x);
>> plot(x,y)
```

or

```
>> fplot(@example,[0 1])
```



```
>> a=fminsearch(@example,.5);
>> b=quad(@example,0,1);
>> c=fzero(@example,.5)
```

```
>> a,b,c

a =

    0.6370


b =

   29.8583


c =

   -0.1316
```

WRITING FUNCTIONS

```
%This is a function for displaying lettergrade.
%The lettergrade conditions are
% A= grade>=90
% B= 90>grade>=80
% C= 80>grade
function y=grade(x)
if (x>=90)
    y= 'A';          %Condition for lettergrade of A%

elseif (x<90 && x>=80)
    y= 'B';          %Condition for lettergrade of B%

else
    y= 'C';          %Condition for lettergrade of C%

end
```

WRITING FUNCTIONS

```
%A script using a function in order
%to determine if x is greater or than
%zero.
%Save the script greater.m
x=input('Enter the value of x:\n');
greater_smaller (x);
fprintf('The run ends.\n')
```

```
function greater_smaller(x)
if x<0
    fprintf('x is less than zero.\n')
    return
elseif x>0
    fprintf('x is greater than zero.\n')
    return
else
    fprintf('x is equal to zero.\n')
end
```

SUMMARY

Have you learned?

- How to define global variables,
- How to create structs,
- What is the script, How to write scripts,
- What is the function, How to write a function?

ANY QUESTIONS?