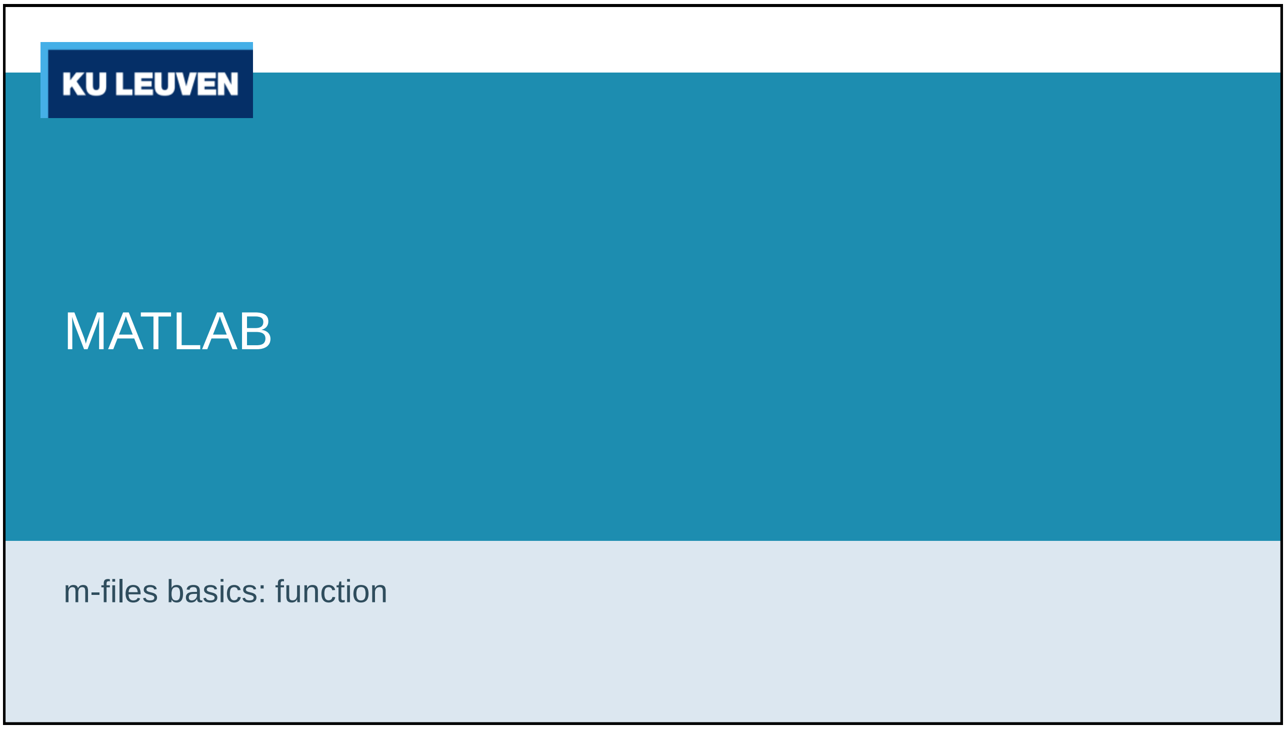
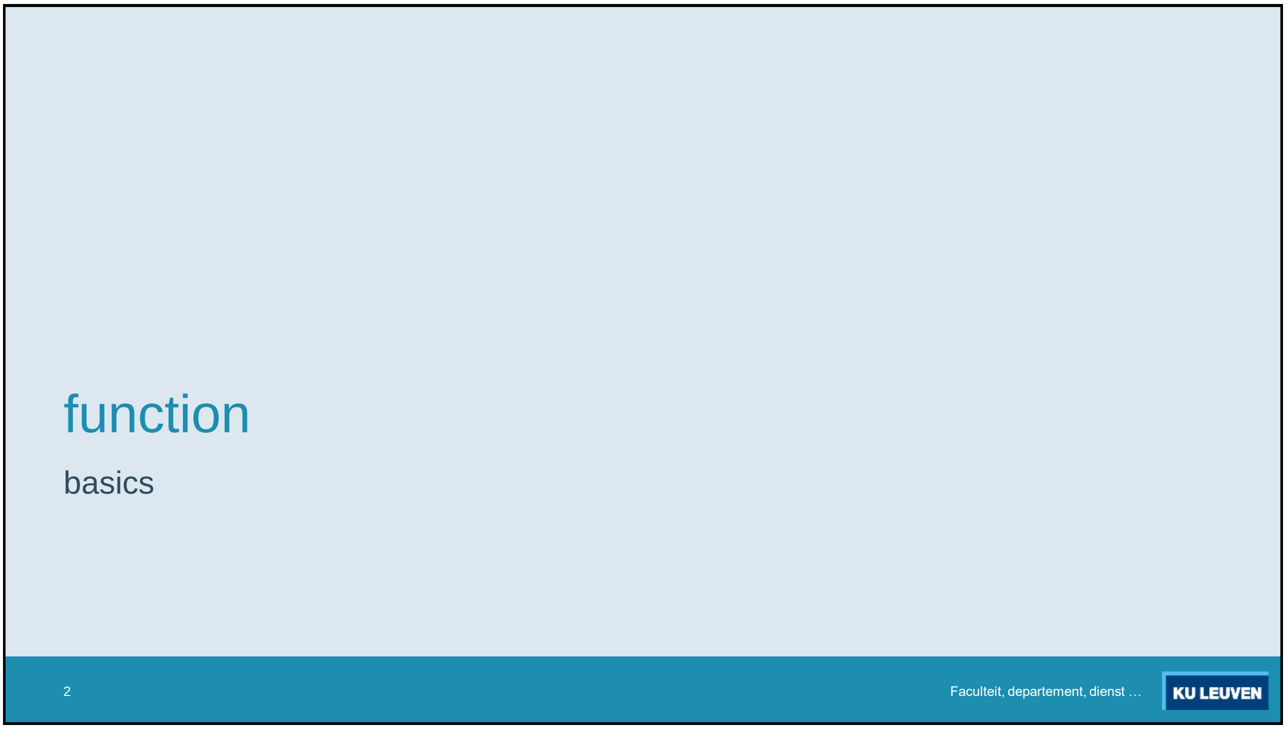


MATLAB

m-files basics: function

1



function

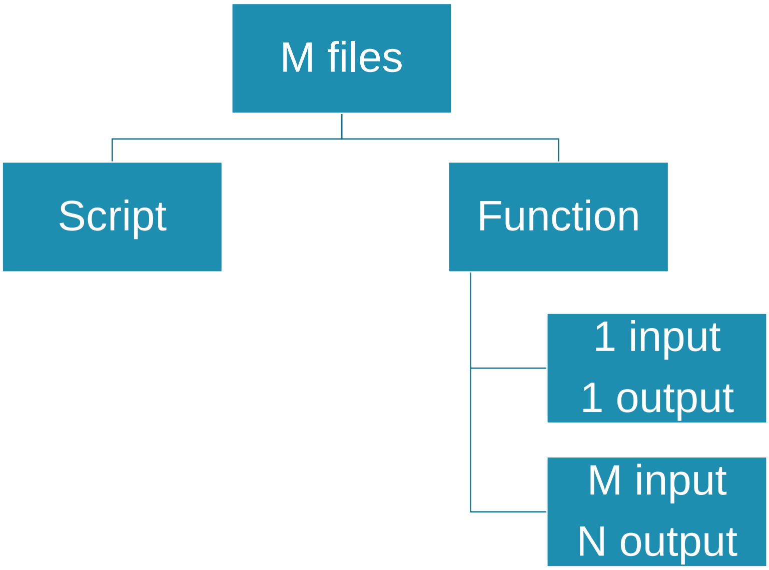
basics

2

M-files

- ASCII text files that contain sequences of MATLAB commands (.m file extension)
- 2 kinds of M-files:
  - **script files** - automate long sequences of commands
  - **function files** - extend MATLAB by developing new commands
- Can be created in MATLAB editor or an editor of your choice

3


M files

Script

Function

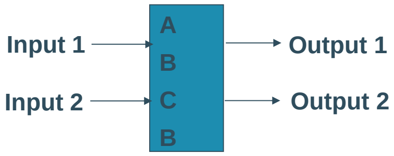
1 input
1 output

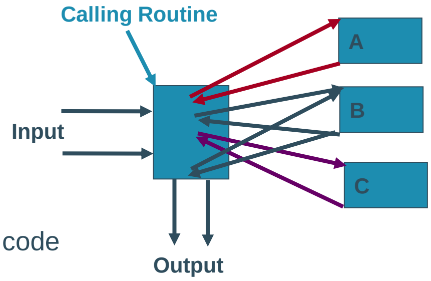
M input
N output

4

Why functions?

- Decompose the problem
  - Break a large block of code into several smaller functions
  - Reduce complexity of the overall code
  - Isolate complex operations
- Reuse
  - Avoid duplicate code
- Encapsulation
  - Local variables
  - Easier to develop in a team
- Maintenance
  - Make debugging and error isolation easier
  - Limit the effect of changes to specific sections of the code



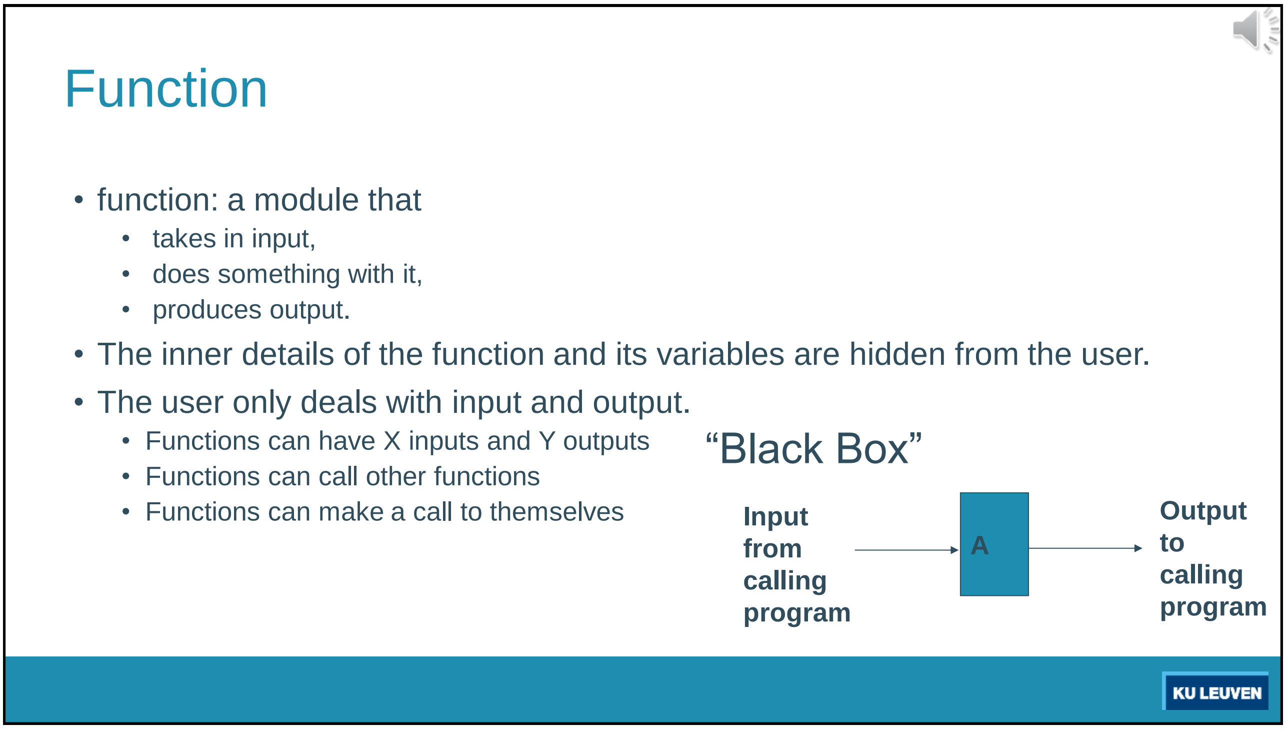


Automate?

- MATLAB Command prompt: enter commands to be executed immediately
  - You can see what you’ve done but it must be re-entered at the command prompt to be recalculated
  - Only the results (variables) are retained in the MATLAB workspace
- Calculate surface of a triangle

```
>> b = 5;
>> h = 3;
>> a = 0.5*(b * h)
```

  - What if you want to enter different values for b and h?
- *File: geometric_object_v1.m*
- *File: geometric_object_v2.m*
- *File: geometric_object_v3.m*



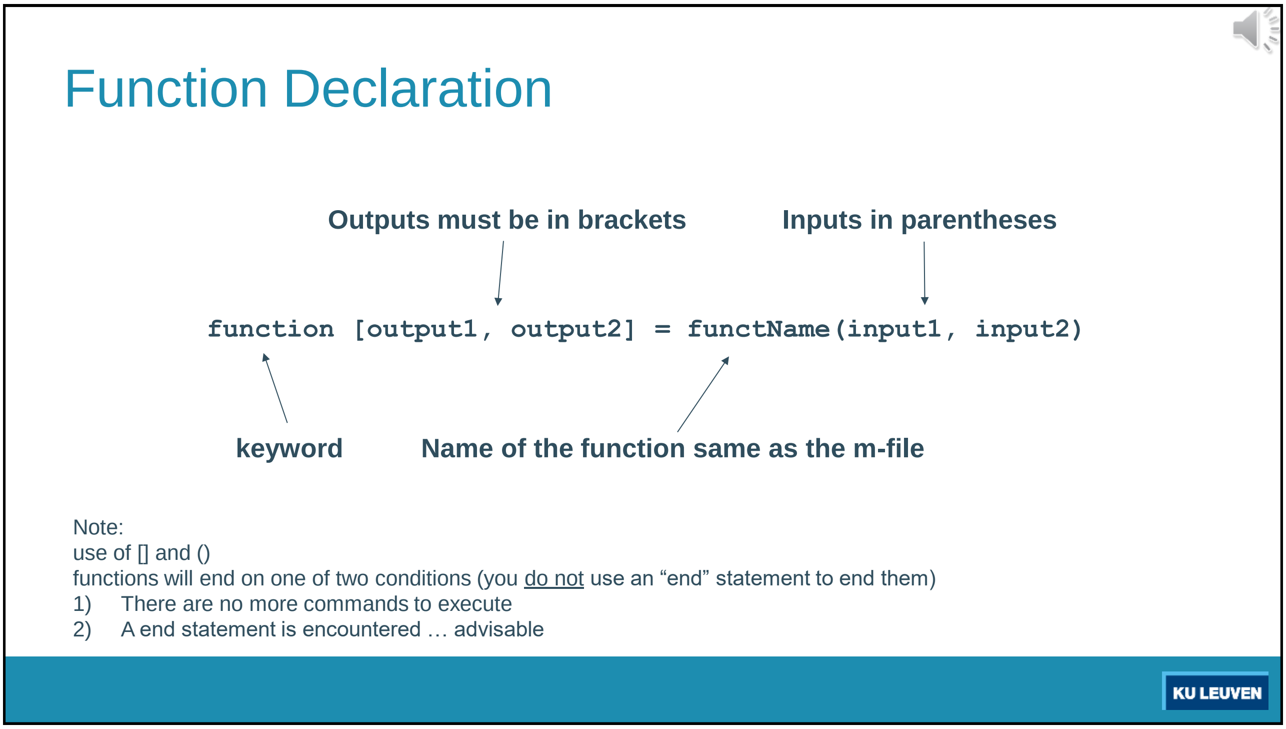
Function

- function: a module that
  - takes in input,
  - does something with it,
  - produces output.
- The inner details of the function and its variables are hidden from the user.
- The user only deals with input and output.
  - Functions can have X inputs and Y outputs
  - Functions can call other functions
  - Functions can make a call to themselves

“Black Box”

Input from calling program
A
Output to calling program

KU LEUVEN



Function Declaration

Outputs must be in brackets

Inputs in parentheses

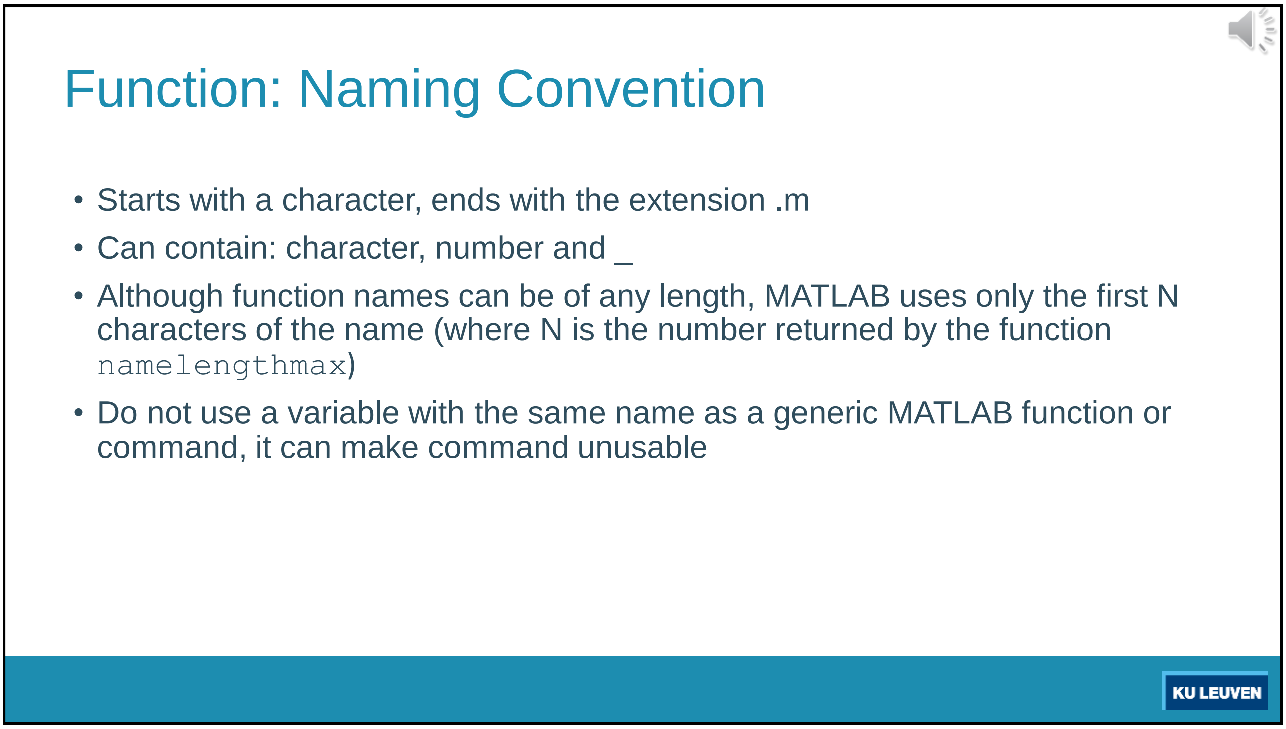
```
function [output1, output2] = functName(input1, input2)
```

keyword

Name of the function same as the m-file

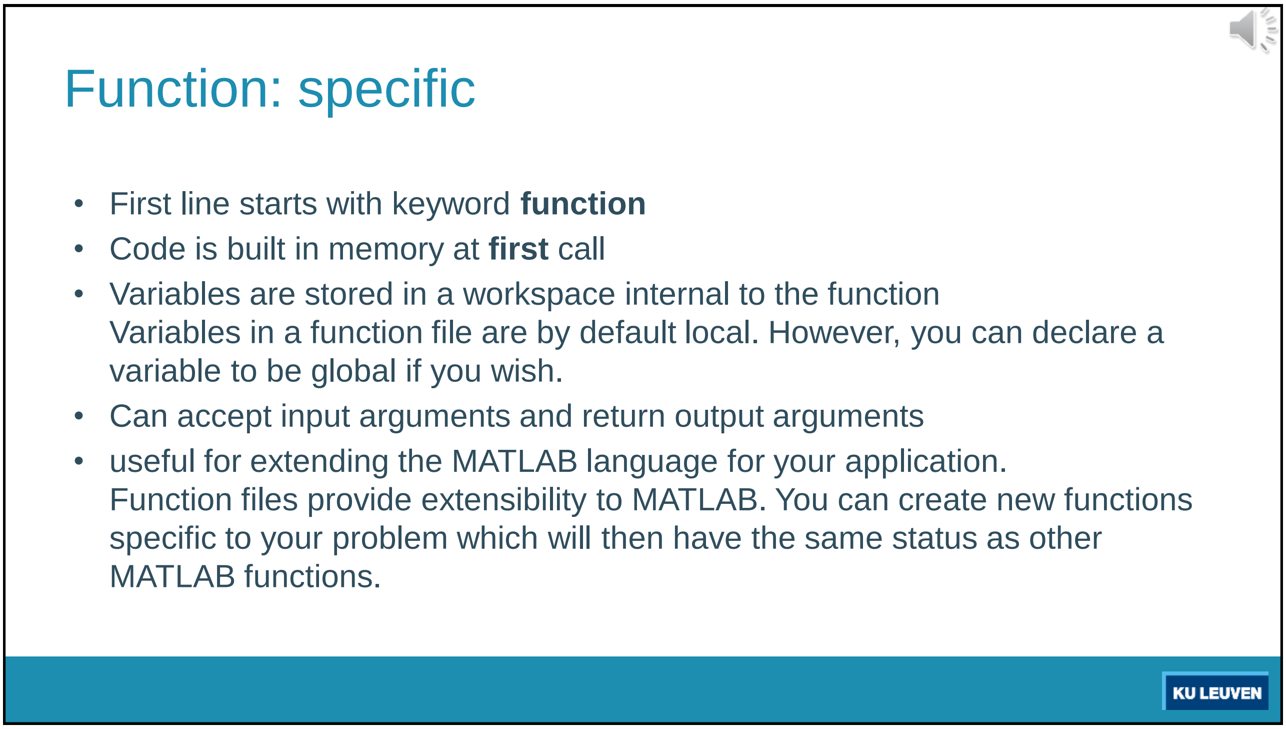
Note:
use of [] and ()
functions will end on one of two conditions (you do not use an “end” statement to end them)
1) There are no more commands to execute
2) A end statement is encountered … advisable

KU LEUVEN



Function: Naming Convention

- Starts with a character, ends with the extension .m
- Can contain: character, number and _
- Although function names can be of any length, MATLAB uses only the first N characters of the name (where N is the number returned by the function `namelengthmax`)
- Do not use a variable with the same name as a generic MATLAB function or command, it can make command unusable

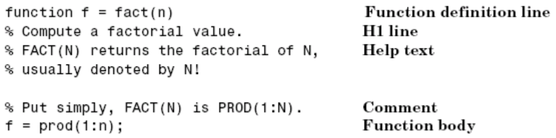


Function: specific

- First line starts with keyword **function**
- Code is built in memory at **first** call
- Variables are stored in a workspace internal to the function
  Variables in a function file are by default local. However, you can declare a variable to be global if you wish.
- Can accept input arguments and return output arguments
- useful for extending the MATLAB language for your application.
  Function files provide extensibility to MATLAB. You can create new functions specific to your problem which will then have the same status as other MATLAB functions.

Function

- Ingredients:
  - Function definition
  - H1-line
  - Help text
  - Body
  - Comments



```
function f = fact(n)
% Compute a factorial value.
% FACT(N) returns the factorial of N,
% usually denoted by N!

% Put simply, FACT(N) is PROD(1:N).
f = prod(1:n);
```

Function definition line
H1 line
Help text
Comment
Function body

- Function definition

```
function y = function_name(x)
```

  - function: key word
  - y: output argument (Results must be stored in variable(s) with the same name as the output arguments)
  - function_name: function name
  - x: input argument

```
function printresult(x) or function [] = printresult(x)
function [x, y, z] = sphere(theta, phi, rho)
```

Summary: Function vs Script

script

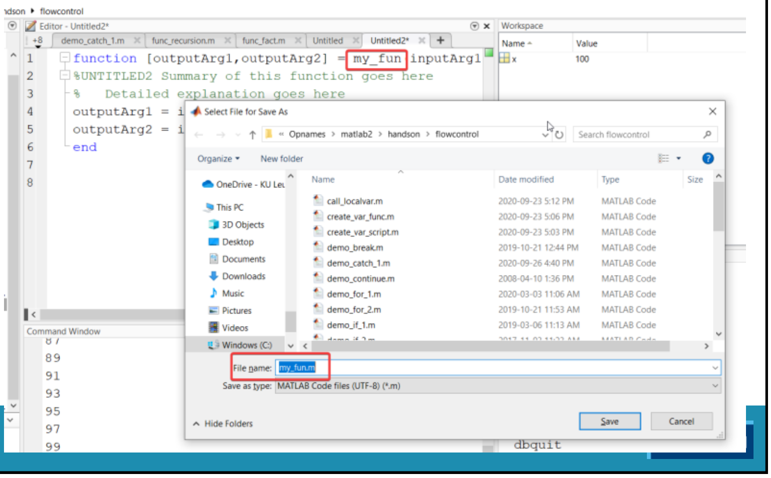
- execute a series of MATLAB statements.
- no arguments.
- operates on data in workspace
  - Data can already be in workspace
  - Leaves the data in the workspace
- scripts share the workspace they are called from.
- scripts are useful for tasks that don't change.
- *File: script_as_function.m*

function

- can accept input arguments and return output arguments.
- internal variables local to function by default.
- useful for extending functionality of MATLAB (create your own functions).
- functions are much more flexible.

Save a function

- The file name needs to be the same as the function name



```
function [outputArg1,outputArg2] = my_fun inputArg1
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
outputArg1 = i
outputArg2 = i
end
```

Name	Value
x	100

```
89
91
93
95
97
99
```

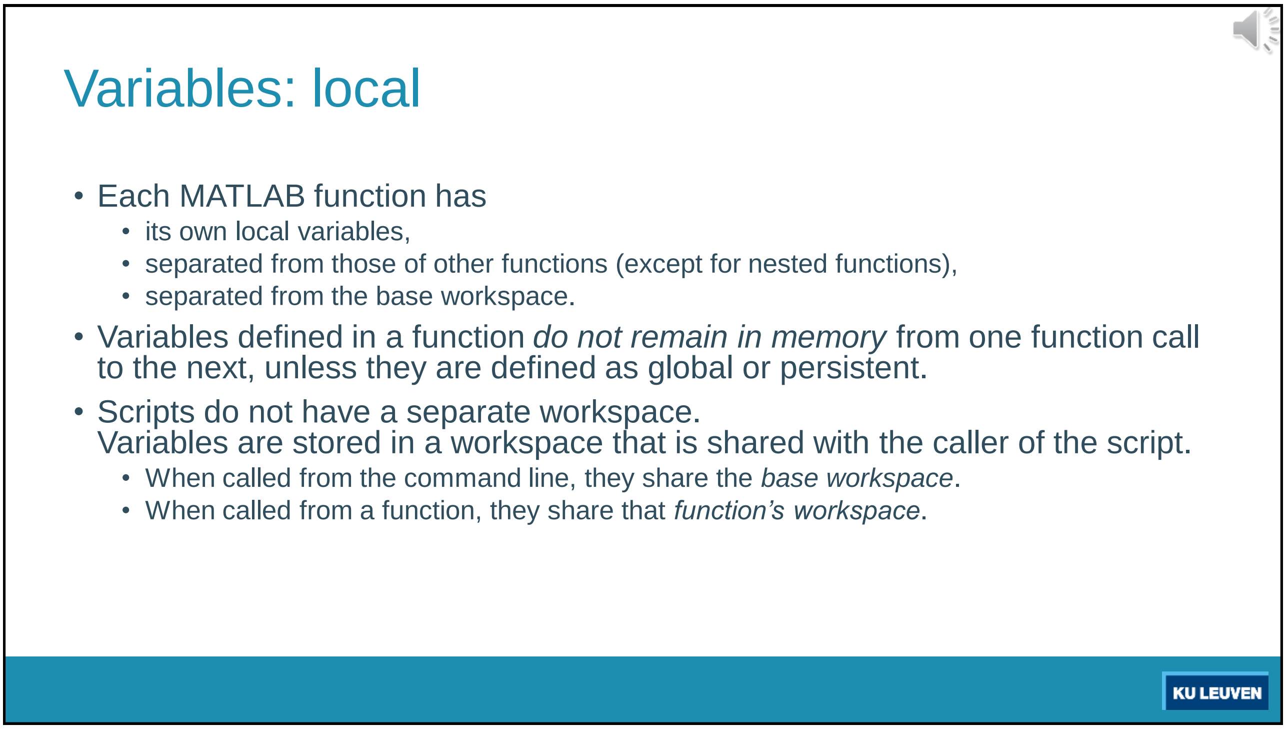
```
dbquit
```

Name	Date modified	Type	Size
call_localvar.m	2020-09-23 5:12 PM	MATLAB Code	
create_var_func.m	2020-09-23 5:06 PM	MATLAB Code	
create_var_script.m	2020-09-23 5:03 PM	MATLAB Code	
demo_break.m	2019-10-21 12:44 PM	MATLAB Code	
demo_catch_1.m	2020-09-26 4:40 PM	MATLAB Code	
demo_continue.m	2008-04-10 1:36 PM	MATLAB Code	
demo_for_1.m	2020-03-03 11:06 AM	MATLAB Code	
demo_for_2.m	2019-10-21 11:53 AM	MATLAB Code	
demo_if_1.m	2019-03-06 11:13 AM	MATLAB Code	

File name: my_fun.m

Save as type: MATLAB Code files (UTF-8) (*.m)

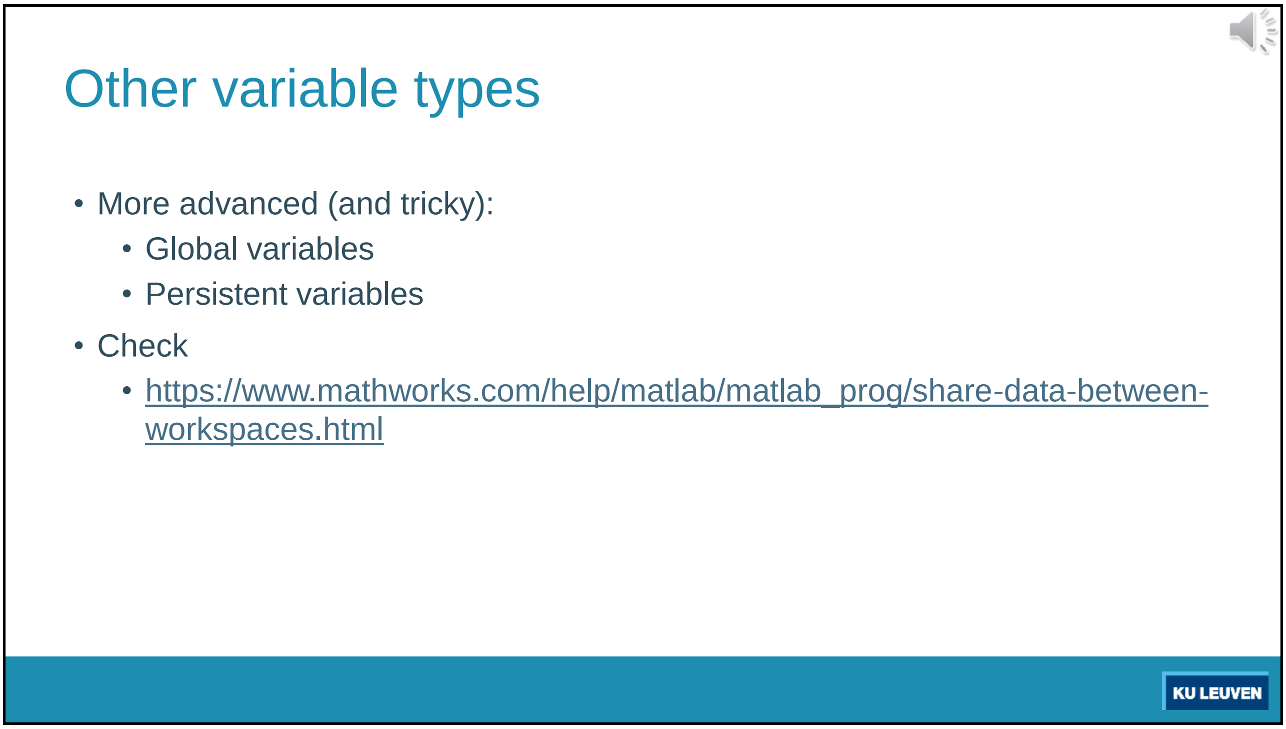
Variable scoping



Variables: local

- Each MATLAB function has
  - its own local variables,
  - separated from those of other functions (except for nested functions),
  - separated from the base workspace.
- Variables defined in a function *do not remain in memory* from one function call to the next, unless they are defined as global or persistent.
- Scripts do not have a separate workspace.
  Variables are stored in a workspace that is shared with the caller of the script.
  - When called from the command line, they share the *base workspace*.
  - When called from a function, they share that *function's workspace*.

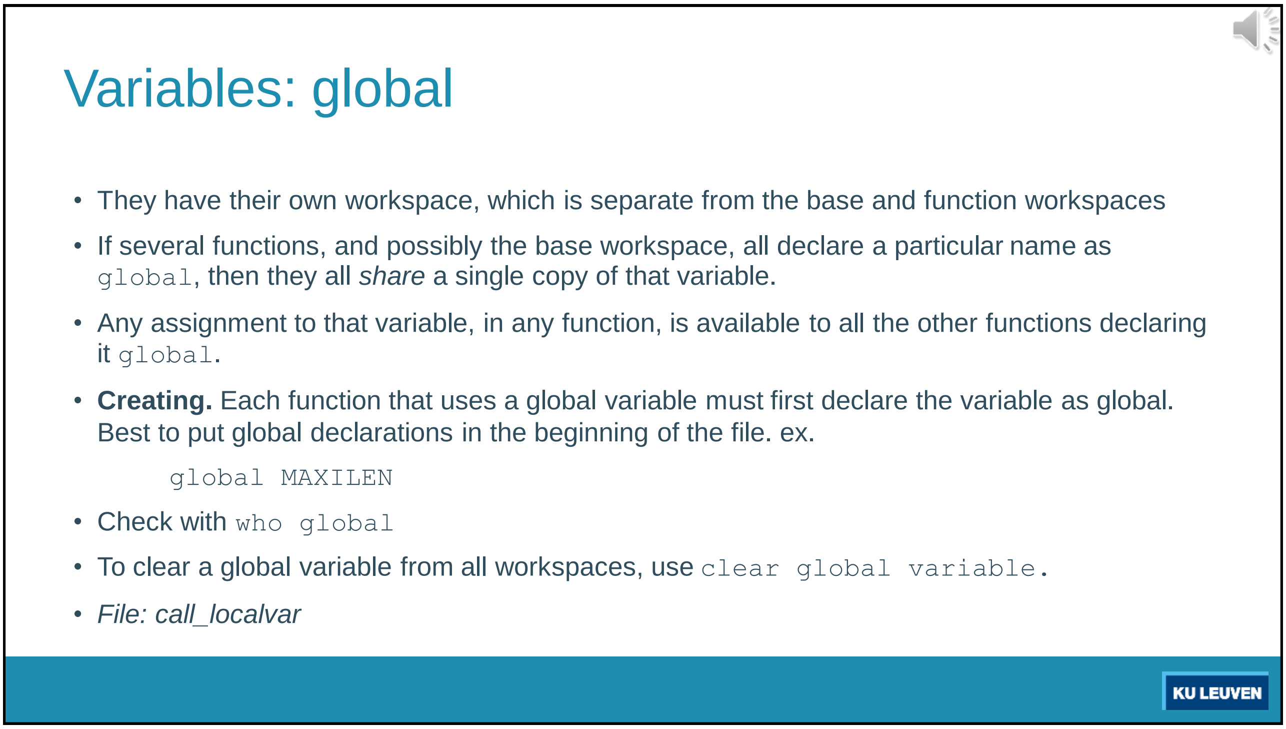
19



Other variable types

- More advanced (and tricky):
  - Global variables
  - Persistent variables
- Check
  - https://www.mathworks.com/help/matlab/matlab_prog/share-data-between-workspaces.html

22

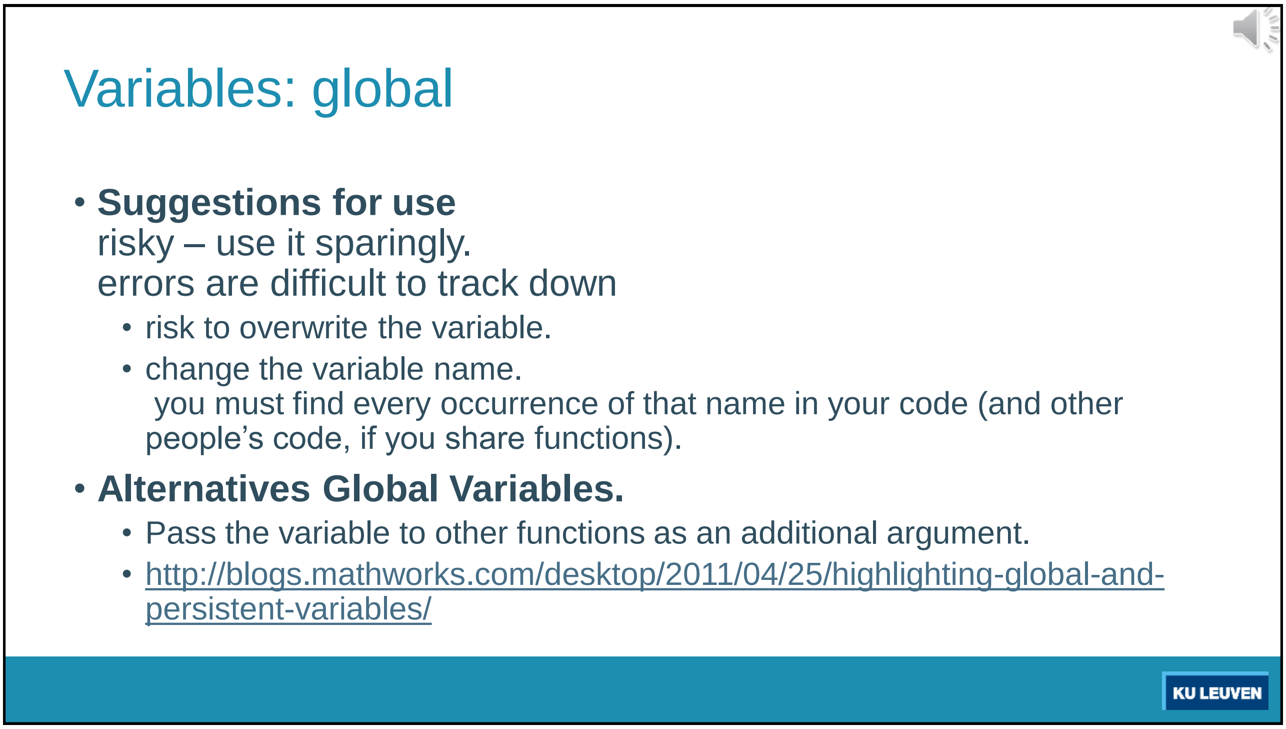


Variables: global

- They have their own workspace, which is separate from the base and function workspaces
- If several functions, and possibly the base workspace, all declare a particular name as `global`, then they all *share* a single copy of that variable.
- Any assignment to that variable, in any function, is available to all the other functions declaring it `global`.
- **Creating.** Each function that uses a global variable must first declare the variable as global. Best to put global declarations in the beginning of the file. ex.

```
global MAXILEN
```

- Check with `who global`
- To clear a global variable from all workspaces, use `clear global variable.`
- *File: call_localvar*



Variables: global

- **Suggestions for use**
  risky – use it sparingly.
  errors are difficult to track down
  - risk to overwrite the variable.
  - change the variable name.
    you must find every occurrence of that name in your code (and other people's code, if you share functions).
- **Alternatives Global Variables.**
  - Pass the variable to other functions as an additional argument.
  - http://blogs.mathworks.com/desktop/2011/04/25/highlighting-global-and-persistent-variables/

Function

More types of functions

Types of functions

- https://nl.mathworks.com/help/matlab/matlab_prog/types-of-functions.html
- Local Functions
- Nested Functions
- Private Functions
- Anonymous Functions

Local function

- Program files can contain multiple functions.
  - useful for dividing programs into smaller tasks, making it easier to read and maintain your code.
  - local functions are only available within the function.
  - local functions have their own workspaces that are separate from the base workspace
- https://nl.mathworks.com/help/matlab/matlab_prog/local-functions.html

```
function b = myfunction(a)
   b = squareMe(a)+doubleMe(a);
end
function y = squareMe(x)
   y = x.^2;
end
function y = doubleMe(x)
   y = x.*2;
end
```

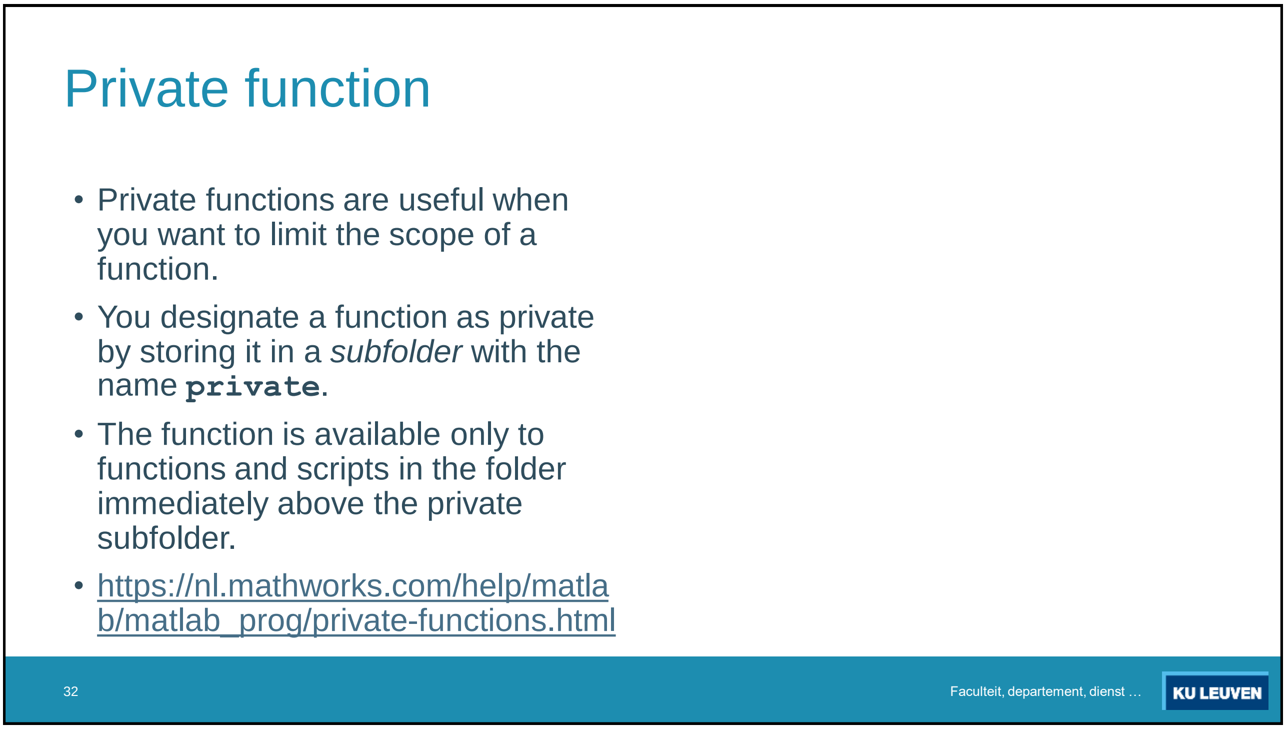
- File: myfunction.m

Nested function

- A nested function is a function that is completely contained within a parent function. Any function in a program file can include a nested function.
- Primary difference between nested functions and other types of functions is that they can access and modify variables that are defined in their parent functions
- https://nl.mathworks.com/help/matlab/matlab_prog/nested-functions.html

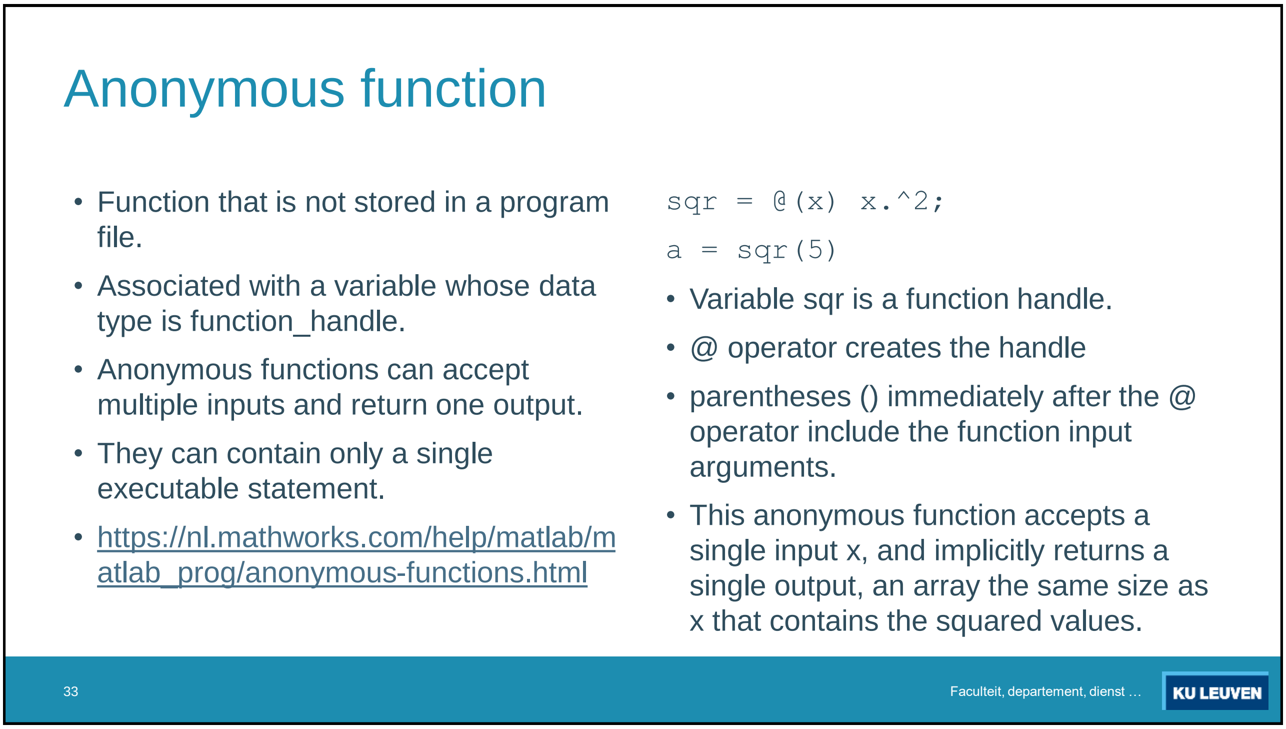
```
function nestmain1
x = 5;
nestfun1
   function nestfun1
     x = x + 1;
   end
end

function nestmain2
nestfun2
   function nestfun2
      x = 5;
   end
x = x + 1;
end
```



Private function

- Private functions are useful when you want to limit the scope of a function.
- You designate a function as private by storing it in a *subfolder* with the name **`private`**.
- The function is available only to functions and scripts in the folder immediately above the private subfolder.
- https://nl.mathworks.com/help/matlab/matlab_prog/private-functions.html



Anonymous function

- Function that is not stored in a program file.
- Associated with a variable whose data type is function_handle.
- Anonymous functions can accept multiple inputs and return one output.
- They can contain only a single executable statement.
- https://nl.mathworks.com/help/matlab/matlab_prog/anonymous-functions.html

```
sqr = @(x) x.^2;
a = sqr(5)
```

- Variable sqr is a function handle.
- @ operator creates the handle
- parentheses () immediately after the @ operator include the function input arguments.
- This anonymous function accepts a single input x, and implicitly returns a single output, an array the same size as x that contains the squared values.