

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

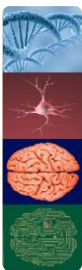
---



Introduction to Programming 2017/2018

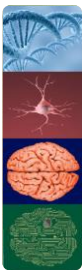
**Tirgul 4: Functions**

Michal Israelashvili  
Yocheved Loewenstern



Functions

- Function – A set of instructions/commands that performs a specific operation, encapsulated by an input/output interface (“black-box”).
- Input – A function can get variables as input – **input arguments**.
- Output – A function can retrieve values – **output arguments**.
- The function recognizes only its arguments and local variables – a **private workspace** (scope) is created when the function runs.



Functions – Implementation

- Functions in MATLAB are saved in .m files (just the same as scripts).
- The file starts with the following line:

`function [output_var] = function_name(input_var)`

- The line starts with the reserved word ‘function’.
- output\_var: the name/s of the variable/s that the function returns (can be more than one and can be none).
- function\_name: the name of the function.
- Input\_var: the name/s of the input argument/s (can be more than one and can be none).

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

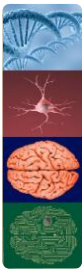
---

---

---

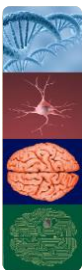
---

---



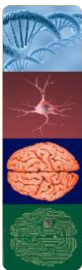
### Functions – Implementation

- The file name must be the same as the function **name** (except for internal functions).
- Every function is saved in a different file (for now...).



### Running a function

- Most functions cannot be run from the editor, but rather **need to be called** (while specifying input/output arguments) from the command window or from another script/function.
- MATLAB only knows functions that are in the current directory (except, of course, MATLAB built-in functions).
- MATLAB runs the latest **saved** version of functions, so make sure all changes are saved before calling your functions.



### Input arguments

- When calling a function you must assign values to **all** input arguments – no more, no less.
- Values are assigned to variables by the **order** they appear in the function definition line.
- When writing a function make sure that it actually makes use of all variables that were defined as input arguments.
- Example: *manyInputFunc*

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

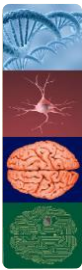
---

---

---

---

---

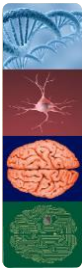


### Output arguments

- When calling a function you **do not** have to assign **all** its output arguments – you can assign all, part of them or none (but trying to assign more output variables than the function has leads to error).
- Output variables are returned by the **order** they appear in the function definition line.
- When no output variables are assigned the function returns the first output argument as the default *ans* variable (so use ; when calling a function to suppress echo!)

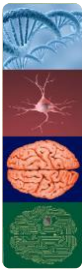
  

- Example: *manyOutputFunc.m*
- nolnOutFunc.m*



### Scopes

- A variable name and value has a meaning only within a scope.
- Different variables may have the same name in different scopes.
- A new scope is created with every call for a function and is destroyed every time it ends.



### Functions – local variables

- Variables are generated and stored locally within the function.
- They are removed from memory upon exit from the function.
- Input arguments are a duplication of the variables used to call the function.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

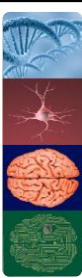
---

---

---

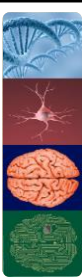
---

---



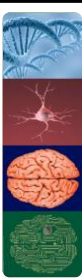
### Scopes – scripts and functions

- Function – when a function is called, a new scope is generated.
- Script – when a script is running, it shares the scope with its parent (or caller).
- The parent of a script/function can be a script/function.
  - Script calls script, script calls function, function calls function, function calls script.
  - Endless hierarchy.



### Scopes – examples

- scopeScript
- func1
- script1
- func2



### Function naming conventions

- Meaningful names:
  - Functions names.
  - Input and output argument names.
  - Don't override existing MATLAB (built-in) functions names or keywords !!!
- Same naming conventions as for variables (Start with a letter, not with a digit, No spaces, etc.).
  - For example: myFuncExample.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

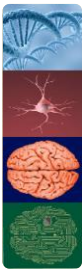
---

---

---

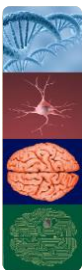
---

---



### Function documentation

- All the comments until the first code line in the function are displayed when typing:  
`help functionName`
- The documentation should include:
  - General description – what the function does.
  - Description of Input and output arguments (what they represent, data type, possible values).
  - Limitations, specifications, requirements etc.
  - Any other required information for the user (or for future code-writing).
- Example: manyInputFunc.m



### Practice

- Write a function called '**circleGeo**' that gets one input argument - the radius of a circle, and returns two output arguments: the perimeter of the circle (first output argument) and the area of the circle (second output argument).
- Test your function using different arrays of circle radiuses (scalar, vector, matrix).
- Notes:
  - Formulas:  $\text{perimeter} = 2\pi R$ ,  $\text{area} = \pi R^2$
  - The value  $\pi$  is represented in MATLAB by `pi`.



### Functions/Commands list

- function