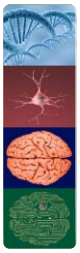
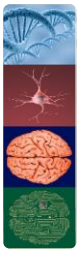


Introduction to Programming 2017/2018

Tirgul 7: Advanced Functions

Michal Israelashvili
Yocheved Loewenstern

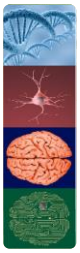


Lesson Outline

- Internal functions
- Variable number of arguments
- Evaluations

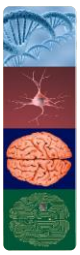
Internal functions

- Instead of writing one function in one file, we can include a few functions in the same file.
- The first function (identical to the name of the file) – the main function.
- The rest of the functions are **internal functions**, available only to the functions in the file.



Internal functions

- Unlike the internal functions, the main function can be called by any other external program.
- **Note that each internal function still has its own private scope.**

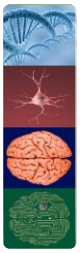


Internal functions - implementation

- The main function is written as usual.
  - A line starts with *function* and then the function code.
- Internal functions come after the main function, each starts as a usual function.
  - A line that starts with *function* indicates the end of the previous functions and beginning of a new function.
- Example: *interFuncs.m*

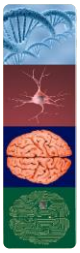
Variable number of arguments

- Matlab allows for a more flexible form of functions programming by using a variable number of input / output arguments.
- This is implemented by the **varargin** & **varargout** expressions.
- Both variables are **cell arrays** and appear as the last arguments of the **input** & **output** arguments of the function respectively.
- Syntax:
  function [var1,…,*varargout*] = myFunc(var1,…,*varargin*)



Number of arguments

- Two main ways to obtain the number of inputs received/outputs required when a function is called:
  - Looking at the number of elements in varargin/varargout.
  - Using the nargin/nargout expressions.
- This allows a different behavior of the function based on the number of received and expected arguments.
- Examples: *varargEx1.m, varargEx2.m, varargoutEx.m*

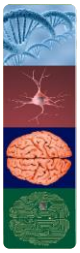


Evaluations

- String evaluation adds flexibility to the code, enabling execution of user-supplied strings and construction of executable strings.
- The *eval* function evaluates a string that contains a MATLAB expression or statement, and returns the result/s of this expression.
- Syntax: eval('string')
- Example file: *evalEx.m*
- Execution of functions: feval.
- Syntax: feval('funcName', *funcArgs*,…)

Practice

- Write a function called **varInP** with one mandatory input argument – a numerical data matrix, and one **optional** input argument – an integer. The function should calculate the mean of the numbers in one of the rows of the input data matrix – the row whose number is given by the second input argument – and return this calculated mean (a scalar) as its output argument.
  - If no second input was provided the function should calculate the mean of the first row.
  - If the second input that was provided by the user has an invalid value then the function should return an empty array as its output. Invalid values of the second input include: a negative number (or zero), or a number larger than the number of rows in the matrix.
- Use 'varargin' and/or 'nargin' in your code.



Functions/Commands List

- varargin, varargout
- nargin, nargout

- eval
- feval