COGS 119/219
MATLAB for Experimental Research

Fall 2014 – Week 4

Functions

User-defined Functions

- A user-defined function is a MATLAB program that is created by the user, saved as a function file, and then can be used like a built-in function.

- The function can be simple single mathematical expression or a complicated and involved a series of calculations.

User-defined Functions

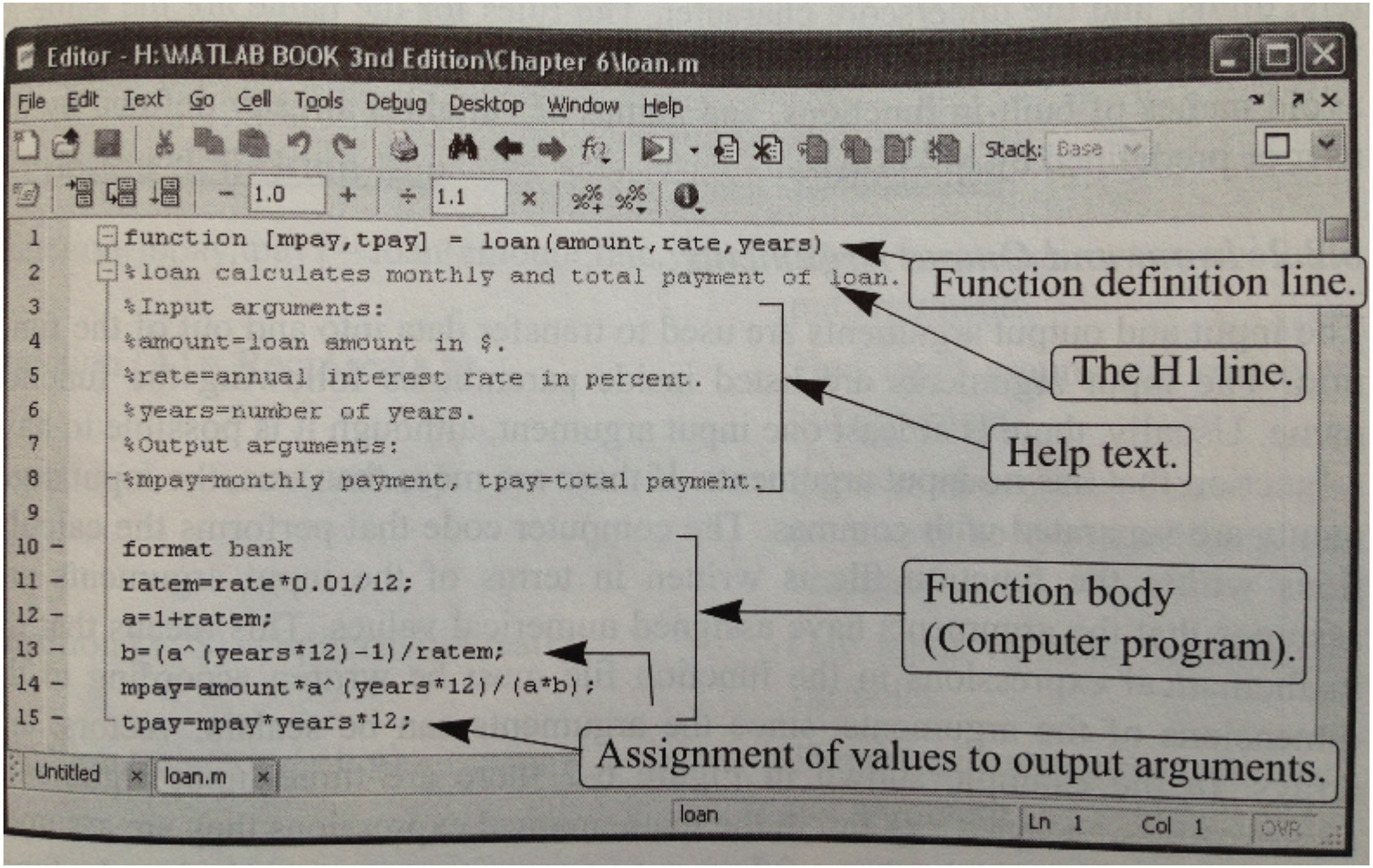
- Calculations inside the function file are carried out using the input data.
- The results of the calculations are transferred out of the function file by the output.
- The input and output can be one or several variables, and each can be a scalar, vector, or an array of any size

How to create a user-defined function in MATLAB?

Structure of a function file

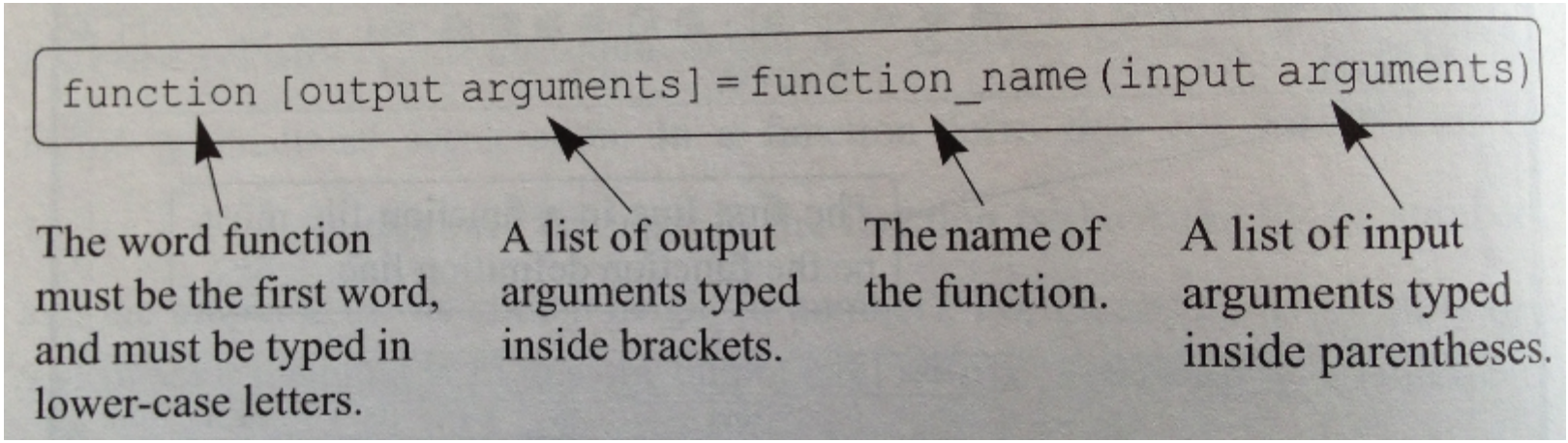
- Let's consider a function which calculates the monthly payment and total payment of a loan.
- INPUTS: the amount of loan, the annual interest, and the duration of the loan
- OUTPUT: the monthly payment and the total payment

Structure of a function file



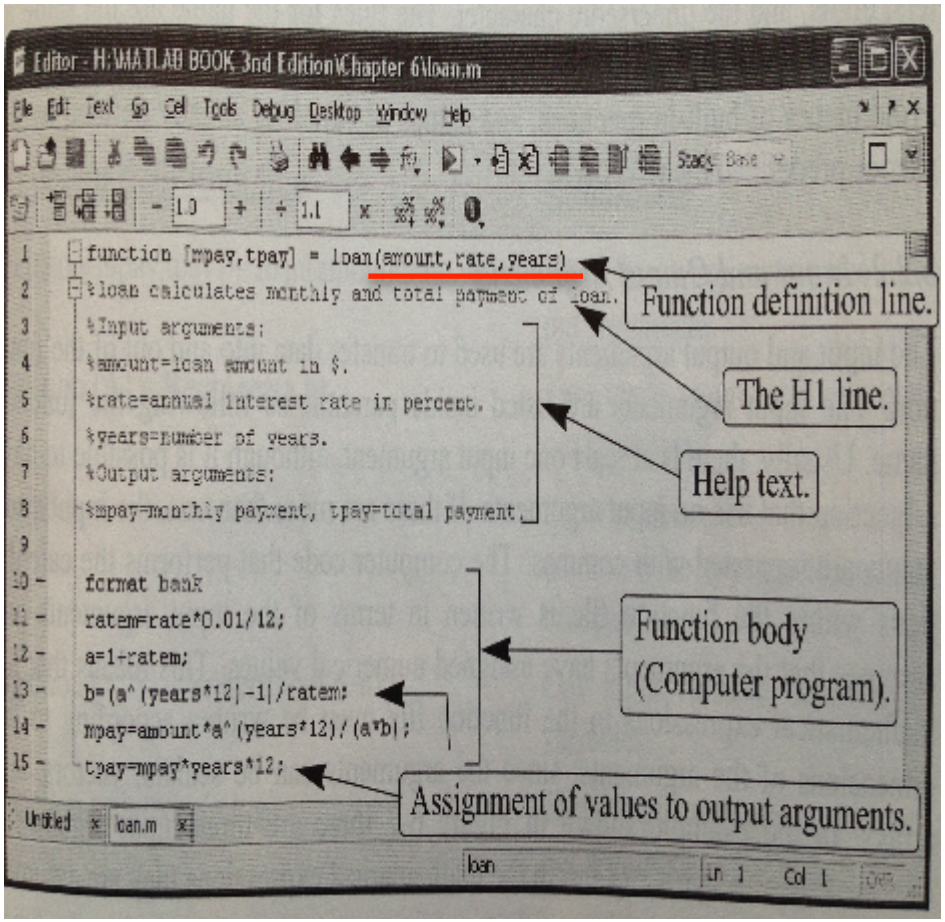
Function definition line

- Defines the file as function file.
- Defines the name of the function.
- Defines the number and order of the input and output arguments.



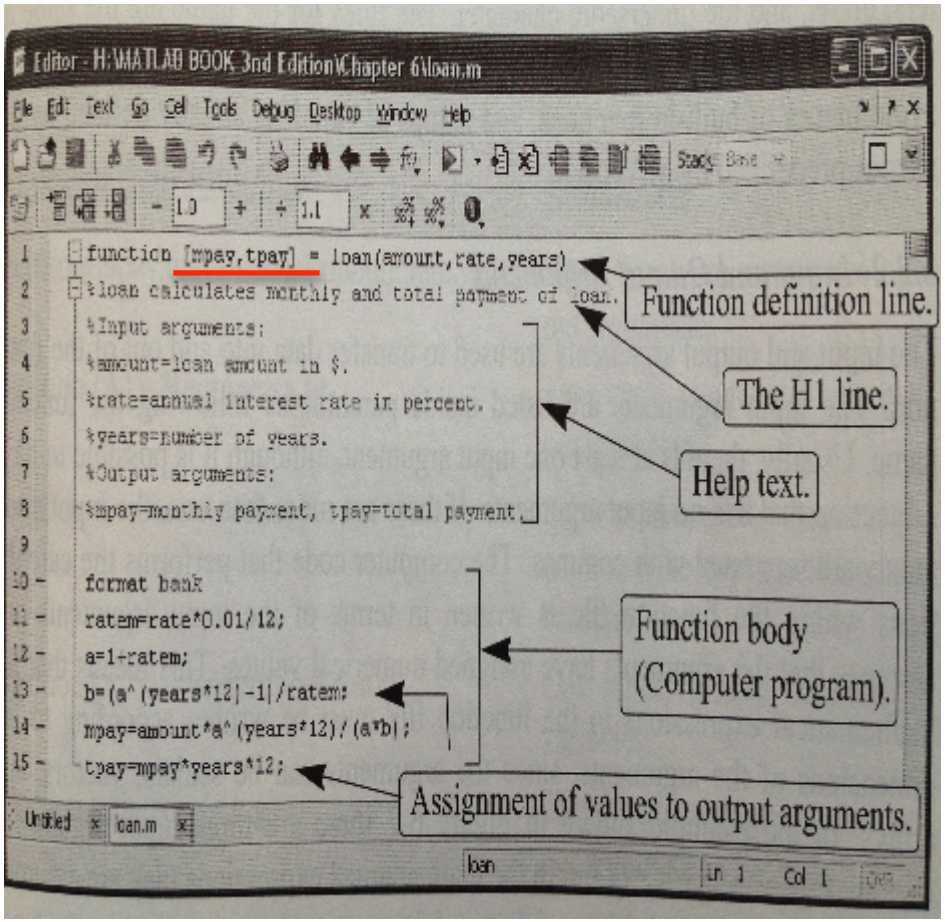
Input Arguments

- Input arguments are listed in parentheses following the function name.
- It is possible to have function with no input arguments.
- The actual values of the input arguments are assigned when the function is used (called).



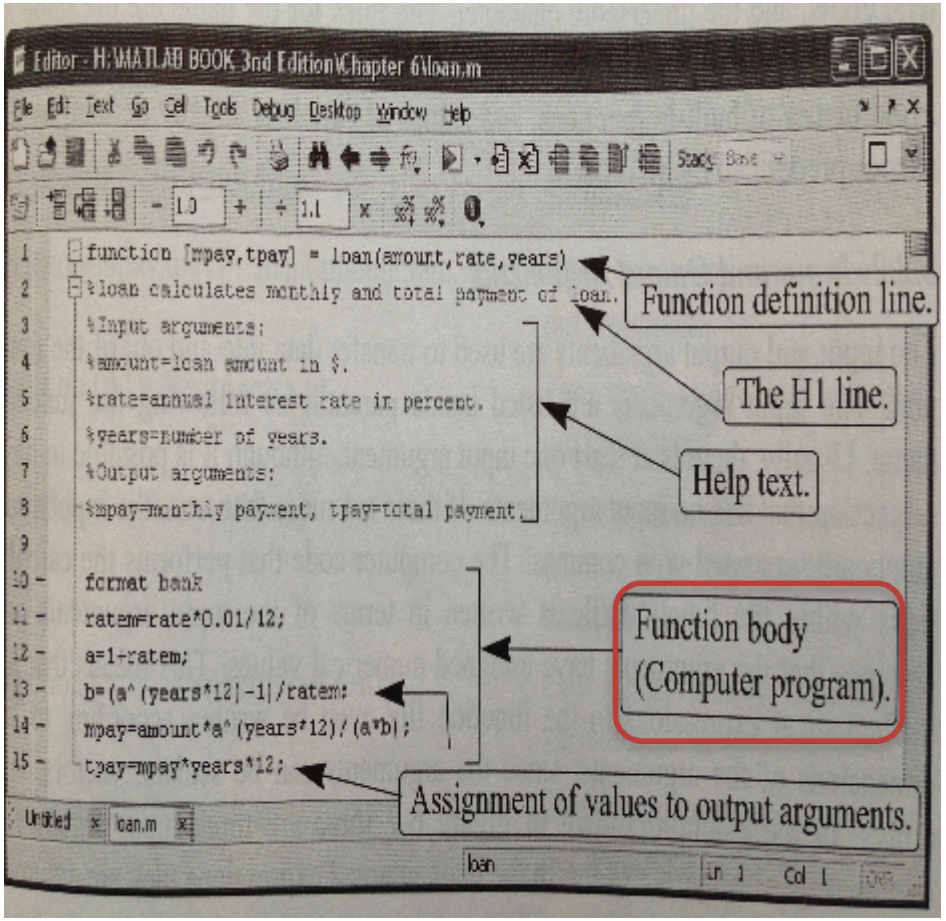
Output Arguments

- Output arguments are listed inside brackets on the left side of the assignment operator in the function definition line.
- A function can have none, one, or several output arguments.
- In order for the function to work, the output arguments must be assigned values in the computer program that is in the function body.



Function body

- The function body contains the computer program that actually performs the computations.
- The program can use all MATLAB programming features (assignments, built-in or user-defined functions, if-statements, loops, etc.)



Variable Scope in Functions

- All variables in a function are local.
- This means that the variables are defined and recognized only inside the function file.
- When a function file is executed, MATLAB uses an area of memory that is separate from the workspace (the memory space of the Command Window and script files).
- Thus, a function can have variables with the same name as variables in the Command Window or in script files.

Variable Scope in Functions

- The function file does not recognize variables with the same name that have been assigned values outside the function.

- The assignment of values to these variables in the function file will not change their assignment elsewhere.

Example

```
a = 3;

In m file

function testfunction()
a=1;
```

```
>> testfunction
```

a will remain 3 because it momentarily became 1 within the function but since the function does not have an output, this doesn't affect the existing variable a.

Example 2

```
function a = testfunction()
a=1;
```

```
>> testfunction
```

a will remain 3 but check out *ans*.

Getting response times

- When we start Psychophysics toolbox, we will use functions that have more precise timing – very important for experiments
  - i.e., we will stop making up reaction time data
- Matlab has its own timing-related functions
- In particular, **tic** and **toc** to get 'quick and dirty' reaction times
- help tic, help toc

tic and toc

- You must use tic before toc
- tic starts a timer, toc then computes time since tic
- You don’t need a variable to call tic and toc. But you can also use variables if you need more than one “timer”

Examples of use

```
>> tic
>> toc
Elapsed time is 1.356002 seconds.

>> t1 = tic
t1 =
     1382075169636112
>> t2 = tic
t2 =
     1382080174362295
>> toc (t1)
Elapsed time is 9.175681 seconds.
>> toc (t2)
Elapsed time is 7.398642 seconds.
```

pause

`>> pause` → Matlab will wait until a key is pressed

`>> pause(2)` → Matlab will wait 2 sec

`>> pause(0.3)` → Matlab will wait 0.3 sec

We will use more precise timer functions with Psychtoolbox soon

```
>> tic; pause (0.3); toc;
Elapsed time is 0.302299 seconds.
>> tic; pause (2); toc;
Elapsed time is 2.000641 seconds.
```

Try help pause for more options

Exercise

- Write a function miniRT that inputs number of trials, numtrials and outputs an array that contains user's reaction time for those trials (i.e., 1 x numtrials array).
- The program displays to user:

```
I say NOW, you press a button as fast as you can
OK? (Press a key to continue)
```

- Then for each trial, the program displays: `Now!`
- The program waits for a key press, calculates reaction time and adds it to the array

Example user-defined function

```
function RTdata = miniRT(numtrials)

RTdata = [];
disp ('I say NOW, you press a button as fast as you can, OK? (Press any
key to continue)');
pause;

for trial = 1:numtrials
    disp ('NOW!');
    tic;
    pause;
    rt = toc;
    RTdata = [RTdata; rt];
    disp ('Press any key to continue');
    pause;
end
```

1. What is the input?
2. What is the output?
3. What does the function do?

Exercise (For You)

- Can you extend the struct example code from last week that made up RT data using rand instead to get user reaction times using tic and toc?
- Can you figure out how you could use miniRT the function or a similar function to do the same?