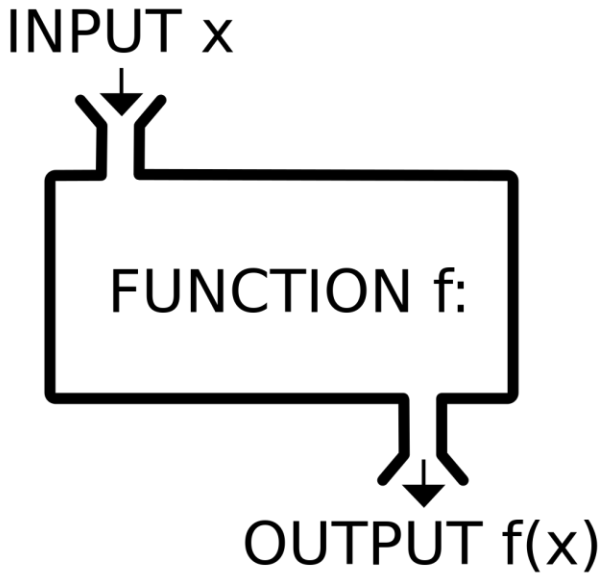
Function .m Files

ME 2004

Function .m Files

- Two types of .m files: scripts and functions
- Function files perform a task and can accept inputs and return outputs
- Two storage options:
  - At the very end of a script file
  - As a standalone .m file (recommended)
- Typically used to break a large problem into smaller, simpler subparts
  - Basis of modular programming



Function .m Files

```
function [out1,out2] = functionName(in1,in2)
% Comments describing the function

code goes here

end
```

Function header

- If the function is saved as a standalone .m file, the filename must be the `functionName`, so choose appropriately!

Function .m Files

```
function [perim,area] = get_geometry(length,width)
% get_geometry: Computes the perimeter and area of a rectangle given its length and
width.
%
% Inputs:
%   length: rectangle's length [m] (scalar)
%   width: rectangle's width [m] (scalar)
%
% Outputs:
%   perim: rectangle's perimeter [m] (scalar)
%   area: rectangle's area [m^2] (scalar)


% Calculate perimeter
perim = 2*length + 2*width;


% Calculate area
area = length*width;
fprintf('The area of the rectangle is %4.4f m^2\n\n',area)
end
```

Function .m Files

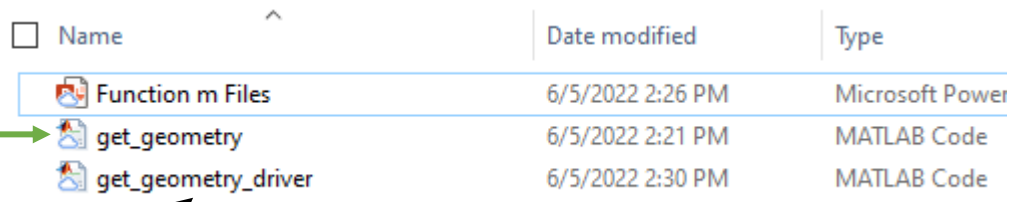
- Documentation is important!

Command Window

```
>> help get_geometry
  get_geometry: Computes the perimeter and area of a rectangle given its length and width.

  Inputs:
    length: rectangle's length [m] (scalar)
    width:  rectangle's width [m] (scalar)

  Outputs:
    perim:  rectangle's perimeter [m] (scalar)
    area:   rectangle's area [m^2] (scalar)

fx >>
```

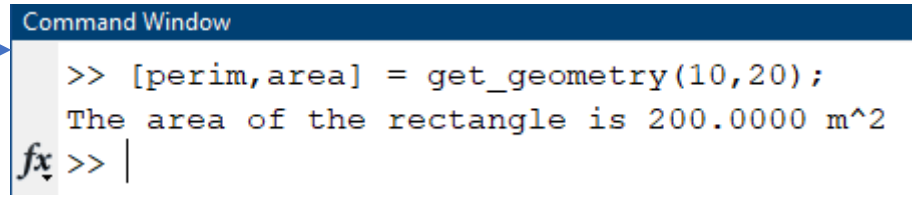
Function .m Files

- Once functions are created, they are stored in the appropriate folder until *called* (or *invoked*)



Name	Date modified	Type
Function m Files	6/5/2022 2:26 PM	Microsoft Power
get_geometry	6/5/2022 2:21 PM	MATLAB Code
get_geometry_driver	6/5/2022 2:30 PM	MATLAB Code

- Calling the function can happen:
  - In the Command Window
  - In a script (often referred to as a *driver script* or *driver file*)



Function .m Files

- Functions can have $0 - \infty$ input and output arguments

Use Case	# Inputs	# Outputs
Print text (such as an introduction or usage instructions)	0	0
Print a customized message (such as a greeting)	1	0
Compute a quantity or multiple quantities	1+	1+

Function .m Files

```
function print_hamlet()
% print_hamlet(): Prints the first 14 lines of Shakespeare's famous "to be or not to be" soliloquy.
%
% Inputs:
%   None
% Outputs:
%   None


fprintf('To be, or not to be, that is the question:\n')
fprintf("Whether 'tis nobler in the mind to suffer\n")
fprintf('The slings and arrows of outrageous fortune,\n')
fprintf('Or to take arms against a sea of troubles\n')
fprintf('And by opposing end them. To die—to sleep,\n')
fprintf('No more; and by a sleep to say we end\n')
fprintf('The heart-ache and the thousand natural shocks\n')
fprintf("That flesh is heir to: 'tis a consummation\n")
fprintf("Devoutly to be wish'd. To die, to sleep;\n")
fprintf("To sleep, perchance to dream—ay, there's the rub:\n")
fprintf('For in that sleep of death what dreams may come,\n')
fprintf('When we have shuffled off this mortal coil,\n')
fprintf("Must give us pause—there's the respect\n")
fprintf('That makes calamity of so long life.\n')
end
```

Command Window

```
>> print_hamlet()
To be, or not to be, that is the question:
Whether 'tis nobler in the mind to suffer
The slings and arrows of outrageous fortune,
Or to take arms against a sea of troubles
And by opposing end them. To die—to sleep,
No more; and by a sleep to say we end
The heart-ache and the thousand natural shocks
That flesh is heir to: 'tis a consummation
Devoutly to be wish'd. To die, to sleep;
To sleep, perchance to dream—ay, there's the rub:
For in that sleep of death what dreams may come,
When we have shuffled off this mortal coil,
Must give us pause—there's the respect
That makes calamity of so long life.
fx >>
```

Function .m Files

```
function make_greeting(name)
% make_greeting(): Prints a greeting given a person's name.
%
% Inputs:
%   name: Name of the person to be greeted (string or character vector)
% Outputs:
%   None


fprintf('Hello, %s! How are you?\n',name)
fprintf('I hope you have a great day!\n')
end
```

Command Window

```
>> make_greeting('Jaisohn')
Hello, Jaisohn! How are you?
I hope you have a great day!
>>
```

Function .m Files

- When calling a function, you can pass in a variable with a different name because the argument is assigned to the corresponding parameter

```
function make_greeting(name)
% make_greeting(): Prints a greeting given a person's name.
%
% Inputs:
%   name: Name of the person to be greeted (string or character vector)
% Outputs:
%   None

fprintf('Hello, %s! How are you?\n',name)
fprintf('I hope you have a great day!\n')
end
```

Command Window

```
>> zxlskdmkjx = 'John Doe';
>> make_greeting(zxlskdmkjx)
Hello, John Doe! How are you?
I hope you have a great day!
>> second_person = 'James';
>> make_greeting(second_person)
Hello, James! How are you?
I hope you have a great day!
>>
```

- The `zxlskdmkjx` variable is mapped to the `name` parameter

Function .m Files

- To ignore an output, replace the variable name in the function call with a ~

```
[items,~,~,things] = do_random_stuff(x,y)
```

  - 2nd and 3rd outputs of the `do_random_stuff` function will be ignored

Summary

- Function .m files perform a particular task and are capable of accepting inputs and outputs (unlike scripts)
- Functions are stored at the very end of a script file, or as a standalone .m file (recommended)
- Calling the function can happen in the Command Window, or in a driver script/driver file (recommended)
- Functions are primarily used to decompose a large problem into more manageable chunks (modular programming)