The How and Why of Writing Functions

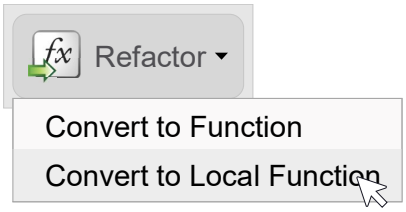
Review

Summary

Summary: The How and Why of Writing Functions

Refactor Code into a Function

1. Highlight the code that you want the function to execute.
2. Click **Refactor** in the **Code** section of the MATLAB Toolstrip.
3. Click **Convert to Function** or **Convert to Local Function**.



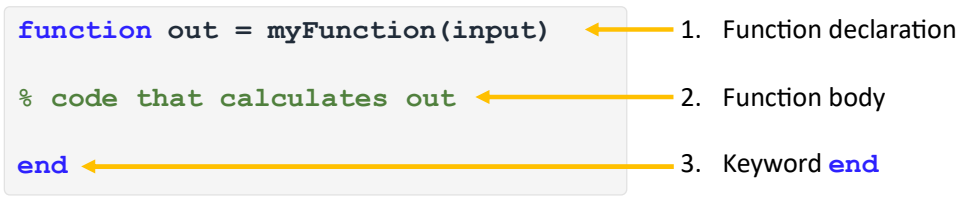
Local Functions vs. Function Files

Function Type	Function Visibility
Local function Function defined within a script.	Visible only within the file where they are defined.
Function file Function defined in a separate file.	Visible to other script and function files.

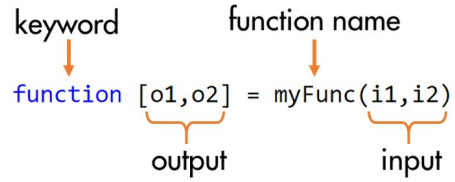
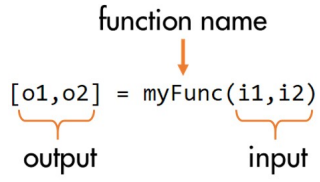
Define a Function

A function definition has three components.

1. The **function declaration** starts with the keyword `function` and defines the syntax for calling the function.
2. The **function body** contains commands to calculate the function output(s) using the input(s). The output(s) must be defined in the body of the function.
3. The `end` keyword is the last line of a function.



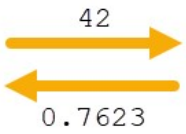
The function declaration defines how users call your function.

Function declaration	Function call
	

Workspaces

A function maintains its own workspace to store variables created in the function body.

```
a = 42;
b = foo(a);
```



foo.mlx

```
1. function y = foo(x)
2.     a = sin(x);
3.     x = x + 1;
4.     b = sin(x);
5.     y = a*b;
6. end
```

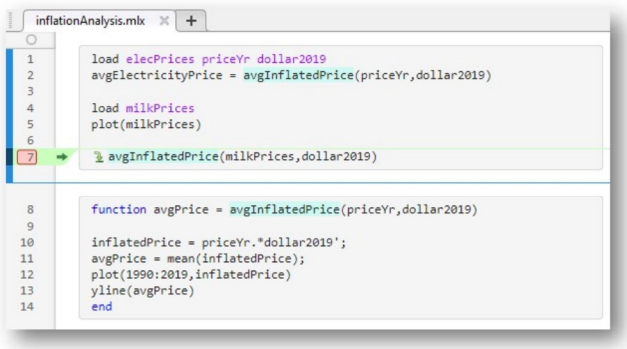
Base Workspace	
a	42
b	0.7623

Function Workspace	
a	-0.9165
b	-0.8318
x	43
y	0.7623

Debugging

You can set breakpoints in functions to stop code execution and view the current workspace. Breakpoints work particularly well with functions, since you typically don't have access to the function workspace. Breakpoints give you access to the same tools you have in scripts for inspecting variables.

Add breakpoints by clicking a line number. You can add breakpoints anywhere in your code file, including inside a function.



 **Continue**: Run code until the next breakpoint (or the end of the script).

 **Step**: Run only the next line of code.

 **Stop**: Stop code execution and exit debug mode.

 **Step In**: Step into a function to see its workspace.

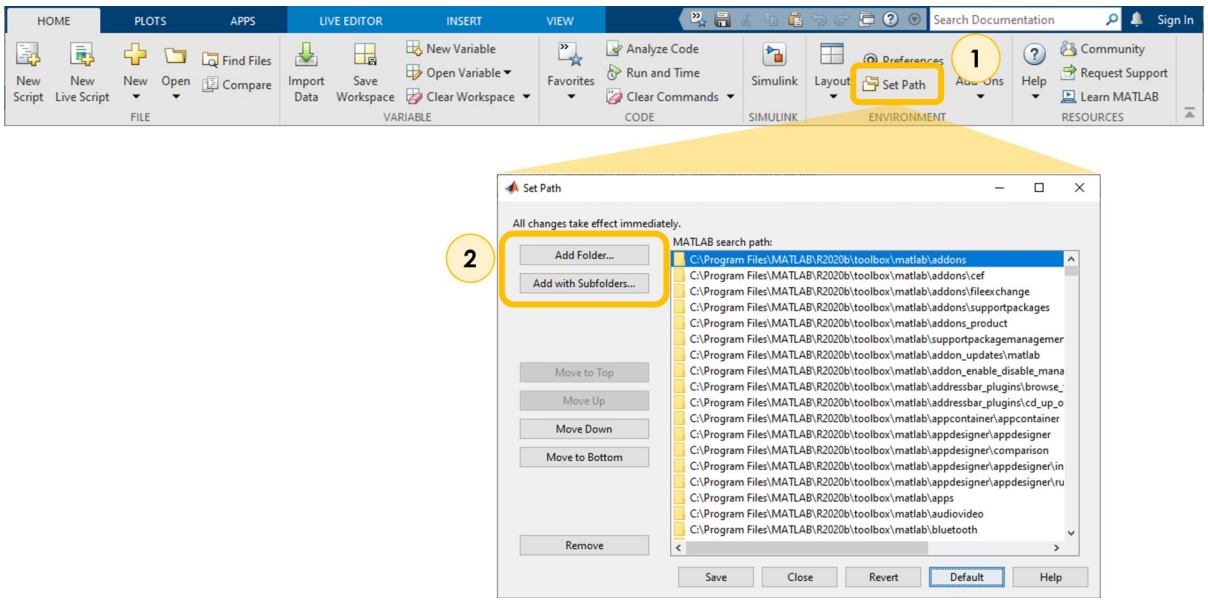
 **Step Out**: Step out of a function to see the workspace where it was called.

When you are finished debugging, remember to clear your breakpoints and save your work.

MATLAB Path

To add folders to the MATLAB search path:

1. On the **Home** tab in the MATLAB Toolstrip, in the **Environment** section, click **Set Path**.
2. Add a single folder or a set of folders using the corresponding buttons.



Calling Precedence

In MATLAB, there are rules for interpreting any named item. These rules are referred to as the *function precedence order*. Most common reference conflicts can be resolved using the following order:

1. Variables
2. Functions defined in the current script
3. Files in the current folder
4. Files on the MATLAB search path

For a more comprehensive list, see Function Precedence Order in the documentation.
