

MATLAB PROGRAMMING TECHNIQUES

Pony Lai

support@terasoft.com.tw

Course Outline

- **Writing Functions**
- **Structuring Code**

Advanced MATLAB® Programming Techniques

Writing Functions



Section Outline

- **Creating functions**
- **Calling functions**
- **Workspaces**
- **Path and precedence**

Increasing Automation

```

% Create the time base for the signal.
fs = 4000;
t = 0:(1/fs):1.5;

% Set the fundamental frequency of the call.
f0 = 175;

% Create the harmonics.
y0 = sin(2*pi*f0*t) + ...
      sin(2*pi*2*f0*t) + sin(2*pi*3*f0*t);

% Set the additional parameters in the model.
A0 = 2; % Initial amplitude.
B = 1.5; % Amplitude decay rate.
fm = 0.65 % Frequency of the modulating envelope.
% Create the envelope
A = A0*exp(-B*t).*sin(2*pi*fm*t)

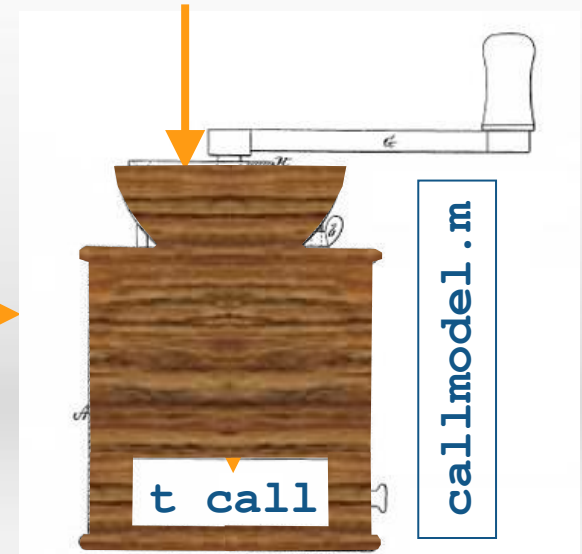
% Create the call
call = A.*y0;

% Plot the model
figure
plot(t,call)
xlabel('Time')
ylabel('Amplitude')
title('\bf Blue')
soundsc(call,fs)
    
```



Name	Value	Size	Bytes	Class
A	<1x6001 doubl...	1x6001	48008	double
A0	2	1x1	8	double
B	1.5000	1x1	8	double
call	<1x6001 doubl...	1x6001	48008	double
f0	175	1x1	8	double
fm	0.6500	1x1	8	double
fs	4000	1x1	8	double
t	<1x6001 doubl...	1x6001	48008	double
y0	<1x6001 doubl...	1x6001	48008	double

f0 A0 B fm



Creating a Function

Function declaration:	Keyword	Output arguments	Function name	Input arguments
-----------------------	---------	------------------	---------------	-----------------

1	<code>function</code>	<code>[call,t]</code>	<code>= callmodel_fun</code>	<code>(N,f0,A0,B,fm)</code>
2				
3	<code>%</code>	<code>CALLMODEL_FUN Models a blue whale B call (func</code>		
4	<code>%</code>			
5	<code>%</code>	<code>Uses a model of the form $y = A.*y0$</code>		
6	<code>%</code>	<code>where $A0 = A*\exp(-B*t).*\sin(2*pi*fm*t)$</code>		
7	<code>%</code>	<code>and $y0$ is a sum of harmonics</code>		
8	<code>%</code>	<code>$yn = \sin(2*pi*n*f0*t)$</code>		

Calling a Function

callmodel_fun.m

```
Command Window
>> [x,t] = callmodel_fun(3,175,2,1.5,0.65);
fx >>
```

```
C:\class\coursefiles\mlbe\whale\callmodel_fun.m
EDITOR PUBLISH VIEW
New Open Save Find Files Insert fx fi Go To Breakpoints Run and Time Run and Advance Advance
FILE EDIT NAVIGATE BREAKPOINTS RUN
1 function [call,t] = callmodel_fun(N,f0,A0,B,fm)
2
3 % CALLMODEL_FUN Models a blue whale B call (functional form).
4 %
5 % Uses a model of the form
6 % y = A.*y0
7 % where A = A0*exp(-B*t) .*sin(2*pi*fm*t)
8 % and y0 is a sum of harmonics
9 % yn = sin(2*pi*n*f0*t)
10 %
11 % Inputs:
12 % N The number of harmonics in the call.
13 % f0 Fundamental frequency of the call.
14 % A Initial amplitude of the call.
15 % B Amplitude decay rate.
16 % fm Frequency of the modulating envelope.
17 %
18 % Outputs:
19 % call The model call.
```

Workspaces

Command Window

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

OVR

42

0.7623

Workspace

Name ^	Value	Size	Class
a	42	1x1	double
b	0.7623	1x1	double

foo.m

```
function y = foo(x)

a = sin(x);
x = x + 1;
b = sin(x);

y = a*b;
```

BlackBox™

Workspace

Name ^	Value	Size	Class
a	-0.9165	1x1	double
b	-0.8318	1x1	double
x	43	1x1	double
y	0.7623	1x1	double

Calling Precedence

>> whale

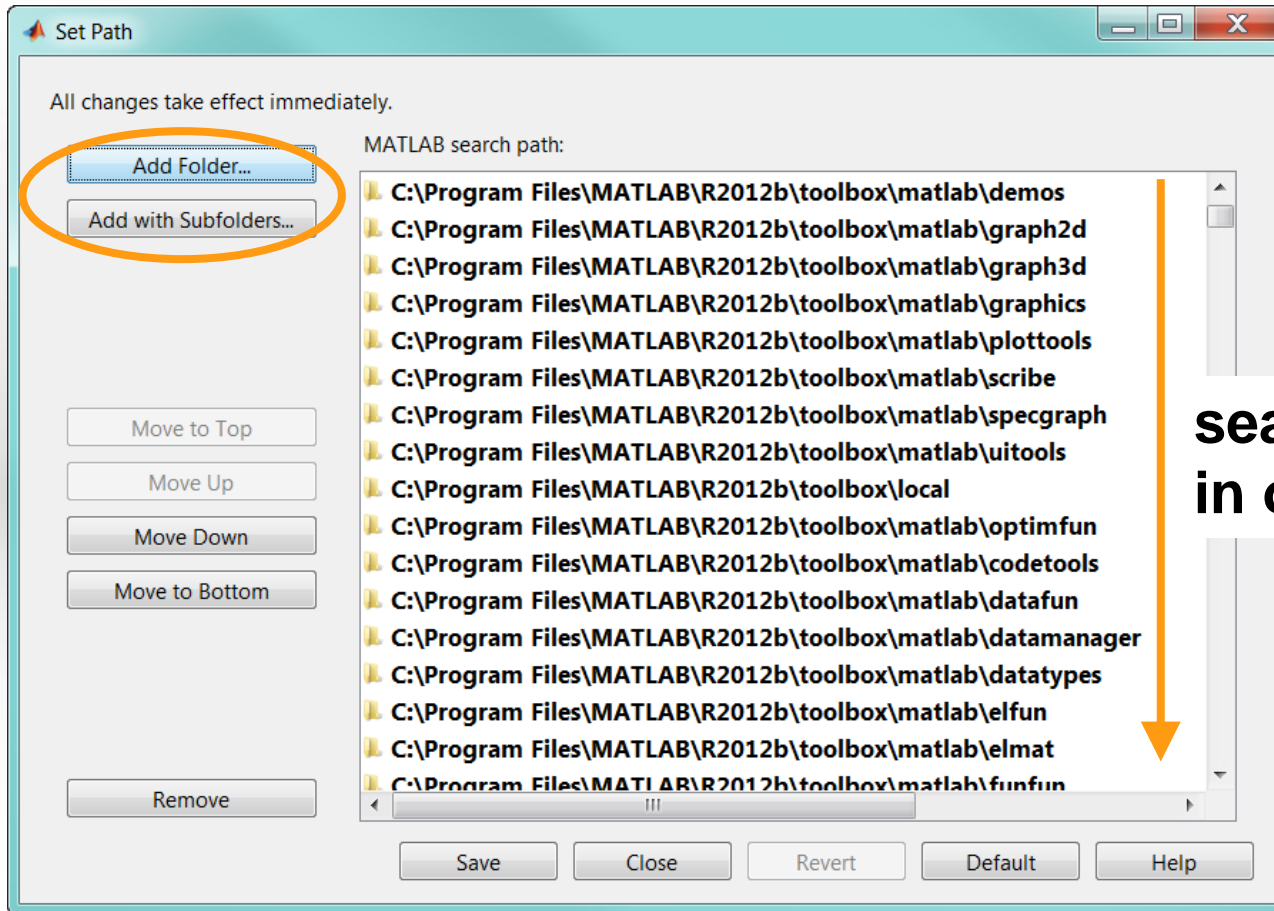
1. Variable
2. Nested function
3. Subfunction
4. Private function
5. Class constructor
6. Overloaded method
7. File in the current directory
8. File on the path



1. whale.bi
2. whale.mexw32
3. whale.mdl
4. whale.p
5. whale.m

The MATLAB® Path

>> pathtool



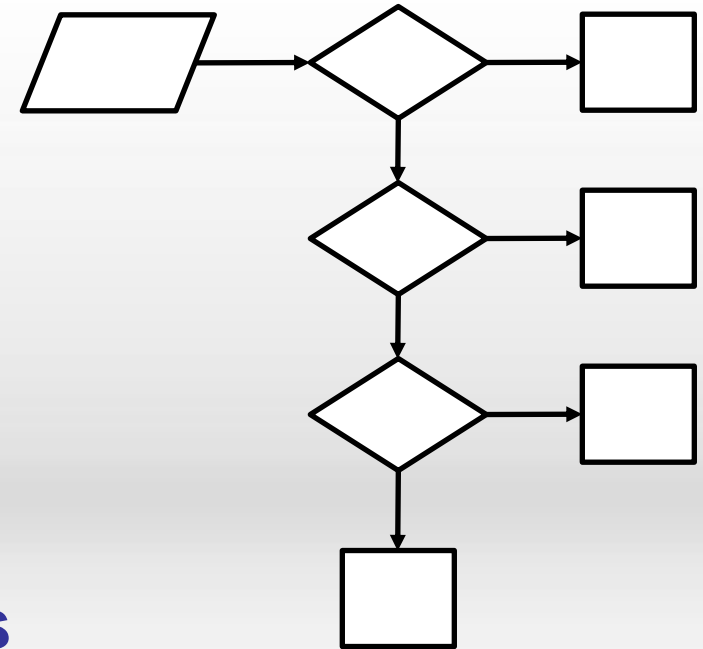
Advanced MATLAB® Programming Techniques

Structuring Code

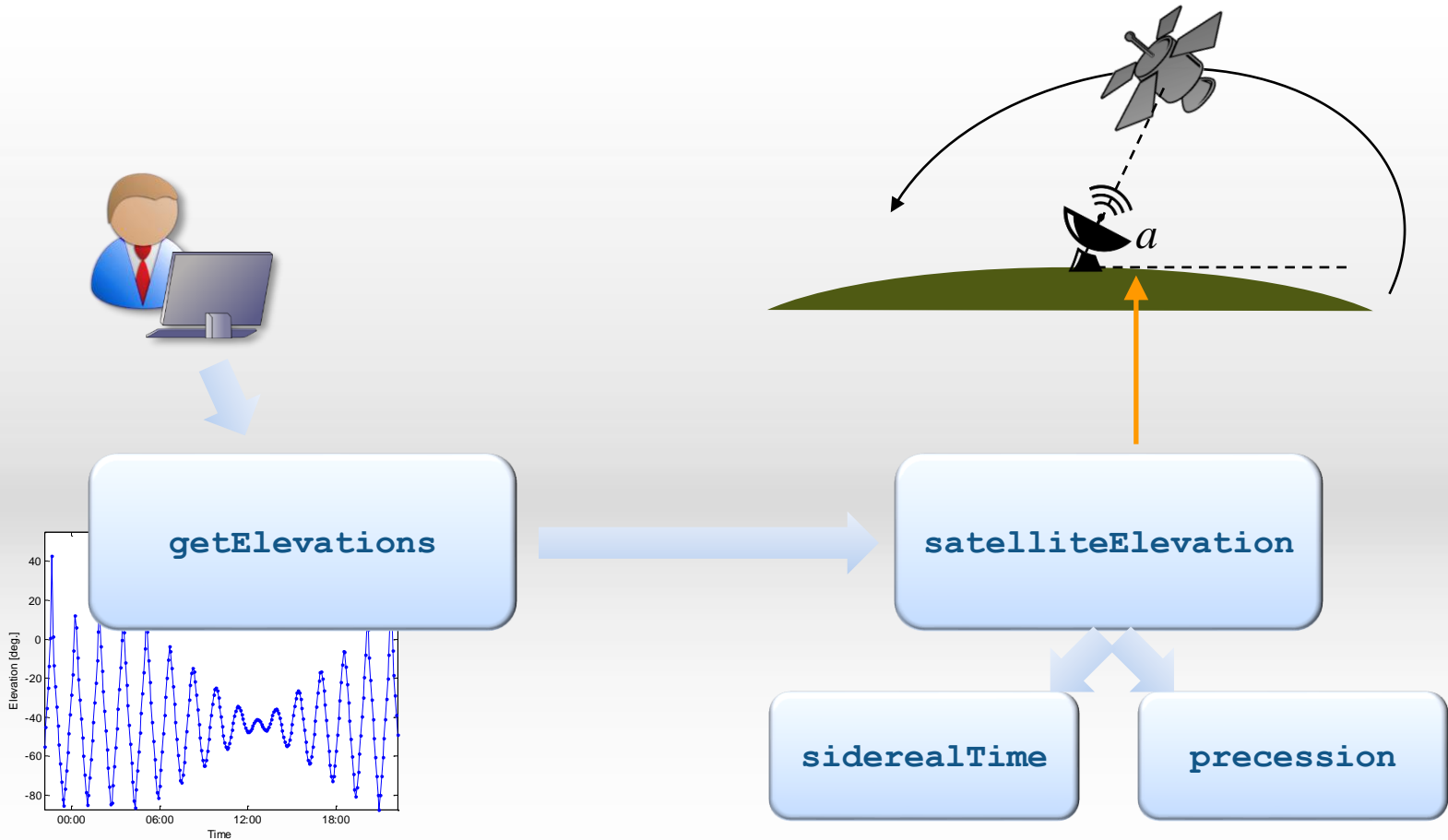


Section Outline

- Private functions
- Subfunctions
- Nested functions
- Function handles
- Anonymous functions
- Precedence rules
- Comparison of function types

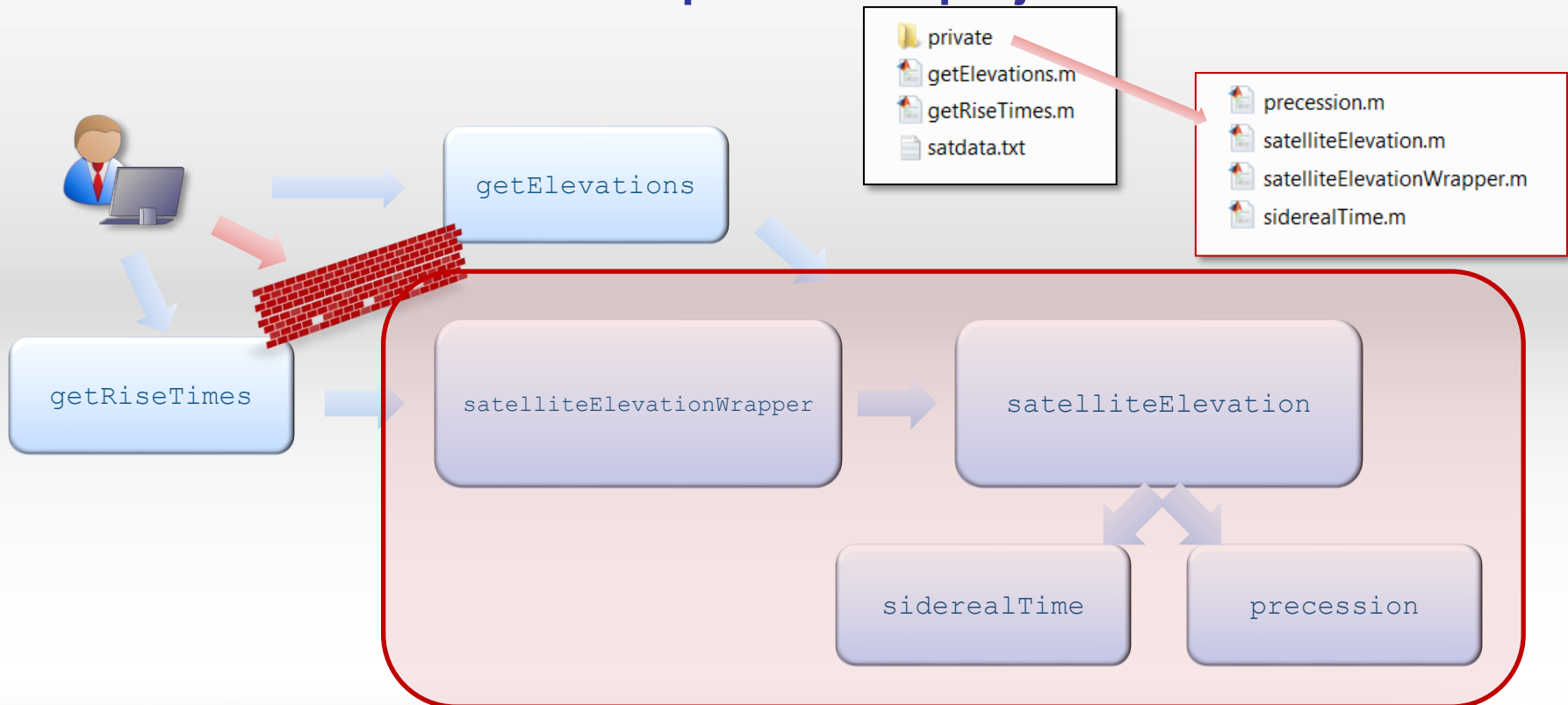


Course Example: Satellite Tracking

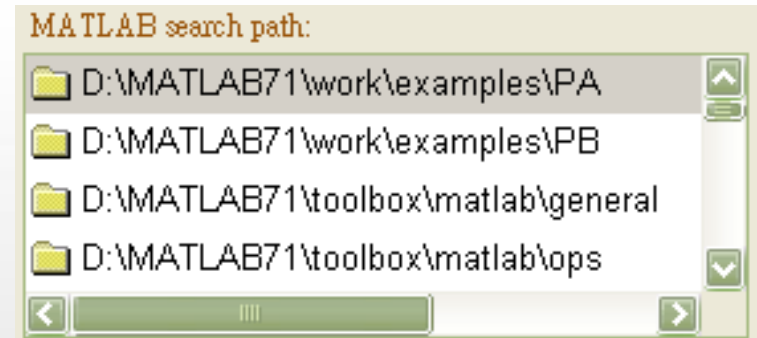
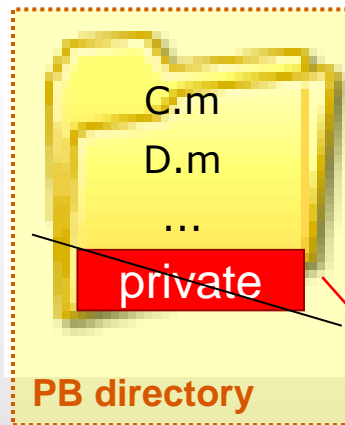
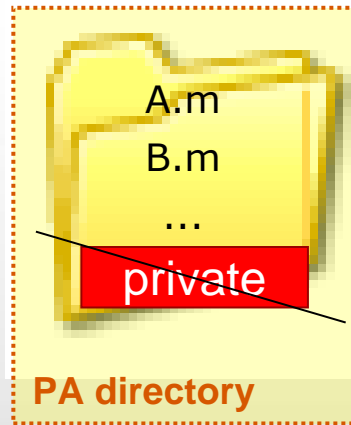


Private Functions

- Function files in a folder named **private**
- Accessible only from within this and the parent folder
- Use case: make function specific to a project



Private Functions



Accessed for parent directory only

```
>> cd([matlabroot ' /work/examples '])
```

```
>> edit AA.m
```

```
function AA(x)
C(x);
```

Subfunctions

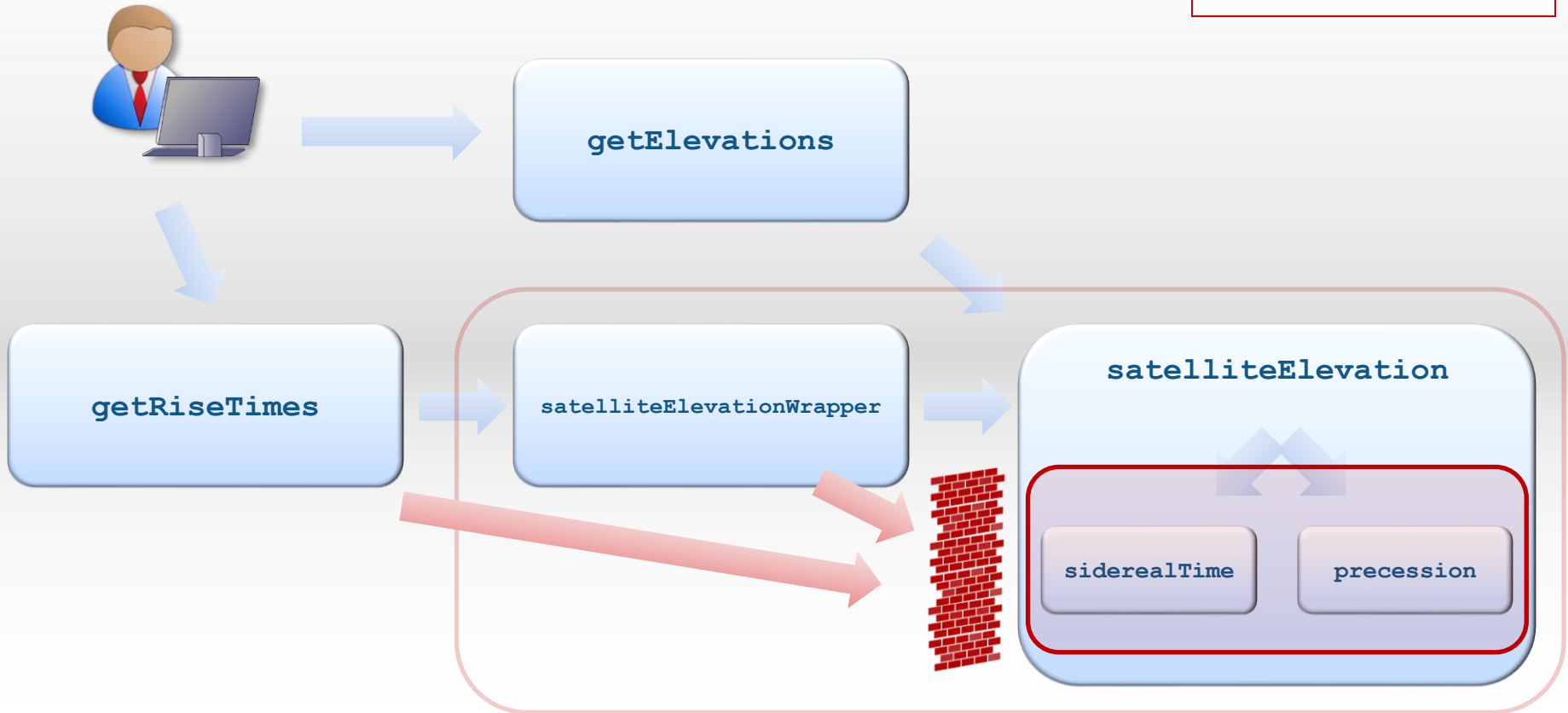
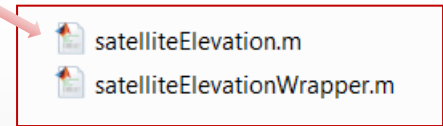
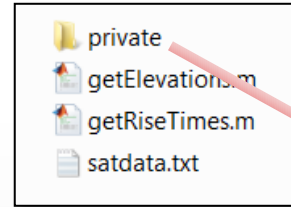
- Several functions in one file
- Keyword `function` used as delimiter
- First function accessible from outside world
- Others accessible only from within the same file
- Use case: hide internal utility functions

Optional when
using only
subfunctions

```

function y = primaryFct(x)
...
end
function y = subFct1(x)
...
end
function y = subFct2(x)
...
end
    
```


Subfunctions

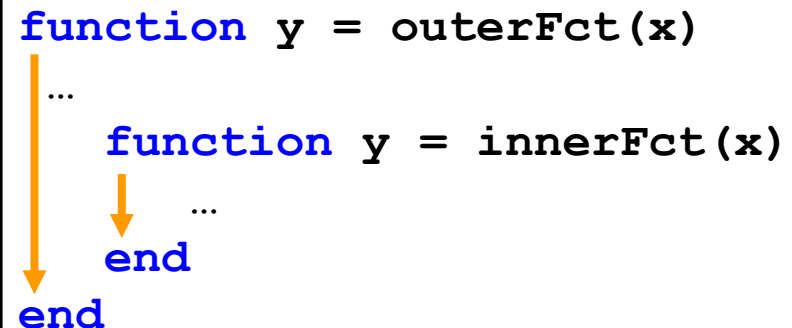


Nested Functions

- Functions nested inside other functions
- Mark their extent using `function` and `end`
- Can be called
 - ◆ From level immediately above
 - ◆ From function at same level within same parent function
 - ◆ From a nested function at any lower level
- Access to superior function workspaces
- Have their own workspace

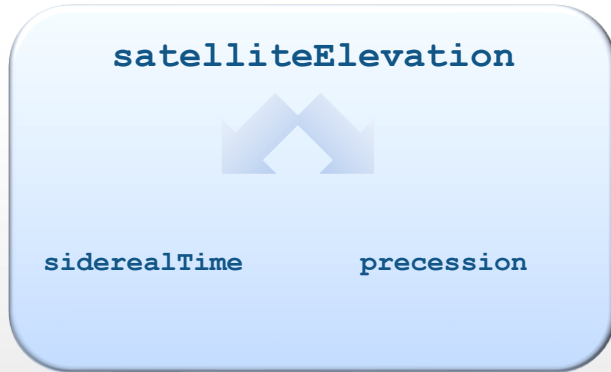
```

function y = outerFct(x)
...
    function y = innerFct(x)
        ...
    end
end
  
```

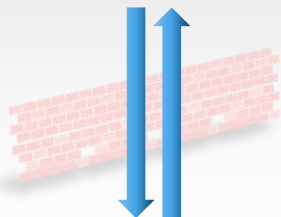


The diagram illustrates the scope of nested functions. A large orange arrow on the left points from the `function` keyword of `outerFct` down to its `end` keyword, indicating its full extent. A smaller orange arrow on the right points from the `function` keyword of `innerFct` down to its `end` keyword, indicating its extent within the `outerFct` function. Ellipses (`...`) are used to represent code between the `function` and `end` keywords of both functions.

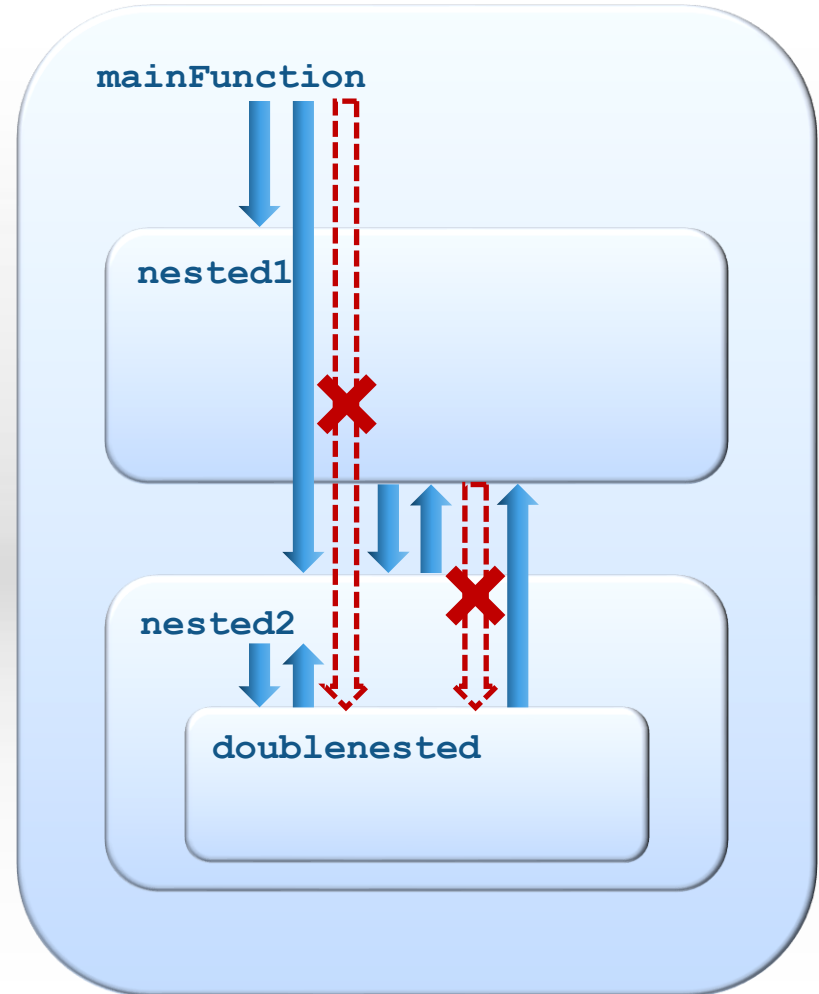
Nested Functions



main function data



nested function



Example: Nested Functions

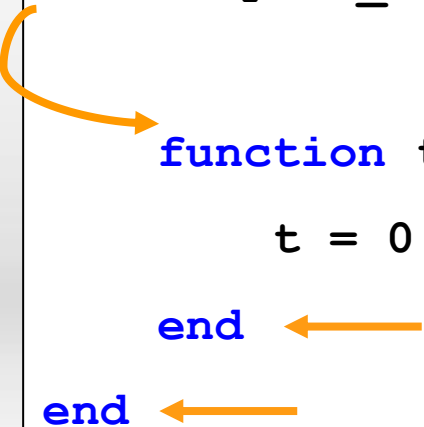
```

function T = tax(income)
adjusted_income = max(income - 6000, 0);
T = compute_tax;

    function t = compute_tax
        t = 0.28*adjusted_income;

    end ←
end ←

```



Scope of a Variable

Using Subfunction

```
function [A,B] = sub_scope(x,y)
```

```
A = subfun1(x);
```

```
B = subfun2(y);
```

```
function v = subfun1(u)
```

```
v = rand(u,1);
```

```
function v = subfun2(u)
```

```
v = randn(u,1);
```

separate
workspaces

Using Nested Function

```
function T = tax(income)
```

```
adj_income = ...
```

```
max(income - 6000, 0);
```

```
T = compute_tax;
```

```
function t = compute_tax
```

```
t = 0.28*adj_income,
```

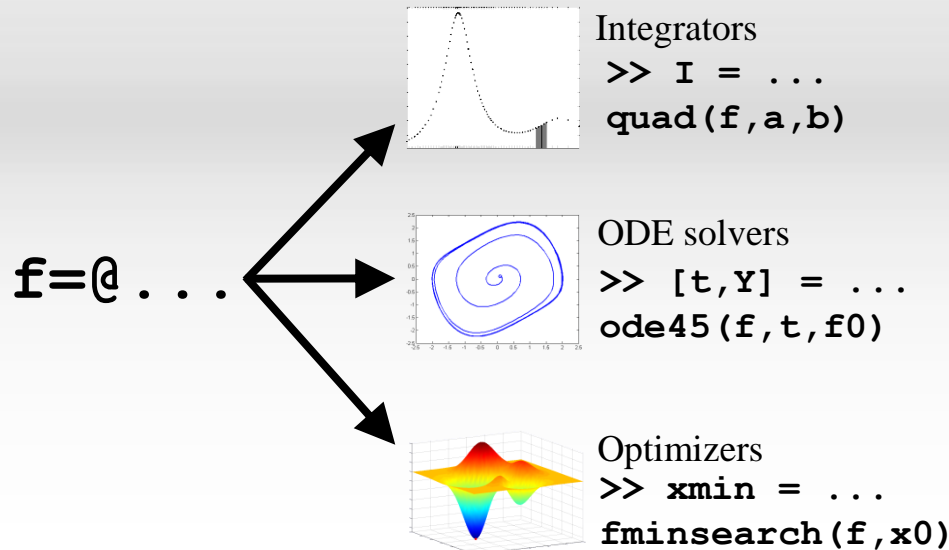
```
end
```

```
end
```

shared
workspaces

Function Handles

- Special MATLAB data type
- Create a variable for calling a function.
- Use case 1: Flexible/dynamic function calls
- Use case 2: Extending the visibility of a function
- Use case 3: Changing the function interface



Creating and Using Function Handles

- **Syntax**

```
fhandle = @functionname
```

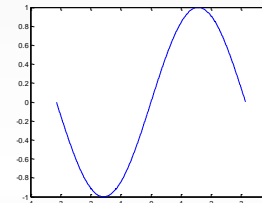
- **Example**

create → `fhandle = @sin;`

use → `fhandle(arg1, arg2, ...);`

```
function plot_fhandle(fhandle, data)
plot(data, fhandle(data));
```

```
>> plot_fhandle(@sin, -pi:0.01:pi)
```



Example:

access to subfunction using function handle

main
function

```
function myFhandle = myFunction(myInput)
SomeOtherValue = 7;
myFhandle = @mySubFunction;

disp(['7 times ' num2str(myInput) ' is ' ...
      num2str(myFhandle(myInput, SomeOtherValue))]);
```

sub-
function

```
function myReturnValue = mySubFunction(x, y)
myReturnValue = x.*y;
```

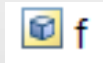
```
>> myTimes = myFunction(7)
```

```
>> myTimes(8,2)
```


Anonymous Functions

- Wrapper to slightly change the function (interface)
- Write @ followed by list of arguments and function call
- No function name
- No file necessary

```
>> f = @myfun;
```



```
function y = myfun(a,b,c)
y = a*(b-sin(c));
```

```
>> f = @(a,b,c) a*(b-sin(c));
```



Comparison of Function Types

Aspect	Private	Sub	Nested	Anonymous
File	Yes	Yes	Yes	No
Workspace	Separate	Separate	Shared	Depends
Access	Files in private and parent folder	Within file	See page 2-6	Via function handle variable
Typical use	Project specific functionality	Hide utilities	Share application data	Change of function interface

Q & A