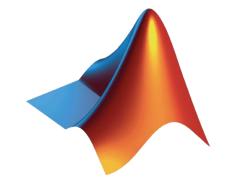


#### MATLAB tools for Sustainable Research Software Development

27.02.2025



**Dr. Mihaela Jarema** Academia Group mjarema@mathworks.com



© 2025 The MathWorks, Inc.



# **Warning:** Your<sup>\*</sup> findings and data can only be understood, reproduced, and built upon if the software you used in the research project is available, (re)usable, and extendable.

\* findings and data of those you collaborate with as RSEs



**Note:** Today's content is designed for RSEs, researchers, and developers who have a foundational understanding of clean code practices and wish to **enhance** their **MATLAB skills for sustainable software development**.

# Intere is related and more detailed content for those without this foundation!



📣 MathWorks

#### This is the (overly simplified) code and data we'll use to get started

Access some data

Explore data

Process data

Visualize processed data

	essedIte	Items.m X +						
1		%% Access data						
2		% Load variables from binary MATLAB file into workspace						
3		<pre>load timestampsCN.mat entryTimestamps exitTimestamps</pre>						
4								
5		%% Explore data						
6	F	% A MAT-file is a binary MATLAB file that stores workspace variables.						
7		% Use whos to list all variables in the workspace, including their type and						
8	L	% size Name Size Bytes Class						
9		whos						
10		entryTimestamps 100x1 1600 datetime						
11		%% Process data         exitTimestamps         100x1         1600         datetime						
12		% Compute elapsed durations between entryTimestamps and exitTimestamps						
13		<pre>durations = exitTimestamps-entryTimestamps;</pre>						
14		% Compute the number of durations						
15		<pre>numberOfDurations = numel(durations); 08:00:00</pre>						
16		% Compute the average duration						
17		averageDuration = mean(durations);						
18		04:00:00						
19		%% Visualize data						
20		% Plot all durations						
21		figure, plot(durations)						
22		% Add a horizontal line with the average duration over the $^{40}$ current $^{80}$ plot $^{100}$						
23		hold on						
24		<pre>plot([1,numberOfDurations],[averageDuration, averageDuration])</pre>						
25		hold off						

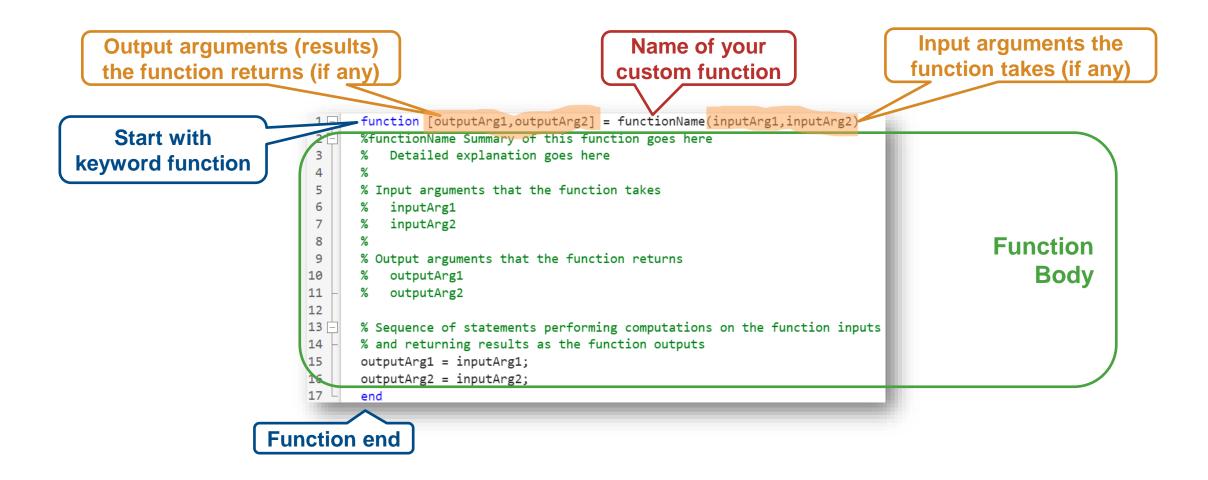
Our goal: Write code to analyze our data in a manner that is easy to maintain, reuse, modify, and extend (also by others).



#### **Code Refactoring Tools**



# Extracting logically-related code into well-designed functions improves readability and reusability







#### MATLAB makes it easy to extract code into functions

 We can refactor our code by extracting code into functions: divide code into smaller logical units by moving fragments of logically-related code into new functions with names that explain what the code fragments do.

HOME PLOTS APPS	EDITOR PUBLISH VIEW Search Documentation	🔎 🜻 Mihaela
Image: Provide state     Image: Provide state     Image: Provide state     Image: Provide state       New     Open     Save     Image: Print     Go To       FILE     Image: Print     Go To	Bookmark         Image: Constraint in the section         Section         Run to End           NAVIGATE         CODE         ANALYZE         SECTION         RUN	
	ma > OneDrive - MathWorks > Workshops > Programming and OOP > CodingPractices > exercise4 © Z Editor - C\Users\miarema\OneDrive - MathWorks\Workshops\Programming and OOP\CodingPractices\exercise4!evaluateProcessed!tems.m	•
Current Folder	Zatior - C.Usersymptema(uneUnive - Mathworks/Workshops/Programming and OUP(LooingPractices/exercise4(evaluateProcesseditems.m)     Zatior - C.Usersymptema(uneUnive - Mathworks/Workshops/Programming and OUP(LooingPractices/exercise4(evaluateProcesseditems.m)	C
timestampsUK.mat     timestampsCN.mat     evaluateProcessedItems.m	1       3% Access data         2       % Load variables from binary MATLAB file into workspace         3       % Load timestamps(K.mat entryTimestamps exitTimestamps         4       load timestampsCN.mat entryTimestamps exitTimestamps         5       3% Explore data         7       % A MAT-file is a binary MATLAB file that stores workspace variables.         8       % Use whos to list all variables in the workspace, including their type and         9       % size         10       whos	
evaluateProcessedItems.m (Script) Workspace	12       3% Process data         13       1% Compute elapsed durations between entryTimestamps and exitTimestamps         14       durations = exitTimestamps-entryTimestamps;         15       % Compute the number of durations;         16       numberOfDurations = numel(durations);         18       averageDuration = mean(durations);         19       19	
Name ^ Value	20       %% Visualize data         21       % Plot all durations         22       figure, plot(durations)         23       % Add a horizontal line with the average duration over the current plot         24       hold on         25       plot([1,numberOfDurations],[averageDuration, averageDuration])         26       hold off	
	Command Window	
	∫x ≫	



#### Your turn: Refactor the code to process timestamps

#### • Prerequisites:

- Copy the materials via the MATLAB Drive <u>https://drive.mathworks.com/sharing/6f91641d-8ddc-4c5d-98d1-90398fbb7197</u>
  - click Download Shared Folder to save a .zip file and work with the files on your MATLAB Desktop if you have a recent MATLAB version available.
  - click Add to my Files > Copy Folder to work with the files in MATLAB Online. You
    will need a MathWorks account with the email address at your institution.
- Your turn:
  - Go to your version of **CleanCodePractices>exercise1** to refactor the code.



## **Tip:** We can use the **Comparison Tool** to display the differences between selected pairs of files

-	luateProcessedItems.m		:
COM	PARISON		
	🚽 🗳 Swap Sides 🍸 😥 🛁		
reviou	s Next C Refresh Filter Publish Merge		
	🔍 Find 👻 👻 Mode		
	NAVIGATE FILTER PUBLISH MERGE		
valua	teProcessedItems.m	evalua	ateProcessedItems.m
1	%% Access data	1	%% Access data
2	% Load variables from binary MATLAB file into workspace	2	% Load variables from binary MATLAB file into workspace
3	<pre>load timestampsCN.mat entryTimestamps exitTimestamps</pre>	3	<pre>load timestampsCN.mat entryTimestamps exitTimestamps</pre>
4		4	
5	%% Explore data	5	%% Explore data
6	% A MAT-file is a binary MATLAB file that stores workspace variables.	6	% A MAT-file is a binary MATLAB file that stores workspace variables.
7	% Use whos to list all variables in the workspace, including their type and	7	% Use whos to list all variables in the workspace, including their type and
8	% size	8	% size
9	whos	9	whos
10		10	
11	%% Process data	11	%% Process data
		12	% Compute elapsed durations between entryTimestamps and exitTimestamps
		13	<pre>durations = exitTimestamps-entryTimestamps;</pre>
		+ 14	% Compute the number of durations
		15	<pre>numberOfDurations = numel(durations);</pre>
2		16	% Compute the average duration
12	[durations,numberOfDurations,averageDuration] = processTimestamps(exitTimestamps,	<b>≠</b> 17	averageDuration = mean(durations);
13		18	
14	%% Visualize data	19	%% Visualize data
15	% Plot all durations	20	% Plot all durations
16	figure, plot(durations)	21	figure, plot(durations)
17	% Add a horizontal line with the average duration over the current plot	22	% Add a horizontal line with the average duration over the current plot
18	hold on	23	hold on
19	( )	24	plot([1,numberOfDurations],[averageDuration, averageDuration])

9



#### **Testing Frameworks**



# Imagine we want to add functionality to, e.g., determine which processing durations took too long

 Let's add a function to compute which & how many processing durations took longer than a given threshold ...

```
getLongProcessingDurations.m 💥 🕇
       function [numberOfDurations, isLong] = getLongProcessingDurations(durations, durationThreshold)
 1 🗐
       % items with long processing durations larger than delayThreshold
       %
 3
       %
           [numberOfDurations, isLong] = getLongProcessingDurations(durations,...
 4
           durationThreshold) returns the number of items with a processing
 5
       %
       %
           duration larger than the provided durationThreshold and a logical array
 6
           storing which item had a long processing duration.
 7
       %
       isLong = durations>durationThreshold;
 8
       numberOfDurations = length(isLong);
 9
10
       end
11
```

You can find the function in **CleanCodePractices>exercise3** 



# Imagine we want to add functionality to, e.g., determine which processing durations took too long

 Is our newly added code correct? Let's run the function with a durationThreshold of 0 seconds (e.g., all our 100 processing durations should take too long...).

```
getLongProcessingDurations.m 🗶 🕂
       function [numberOfDurations, isLong] = getLongProcessingDurations(durations, durationThreshold)
 1日
 2 E
       % items with long processing durations larger than delayThreshold
 3
       %
       %
           [numberOfDurations, isLong] = getLongProcessingDurations(durations,...
 4
 5
           durationThreshold) returns the number of items with a processing
       %
       %
           duration larger than the provided durationThreshold and a logical array
 6
 7
       %
           storing which item had a long processing duration.
       isLong = durations>durationThreshold;
 8
       numberOfDurations = length(isLong);
 9
10
       end
                                >> numberOfDurations = getLongProcessingDurations(durations, seconds(0))
11
                                 numberOfDurations =
                                   100
```



# Imagine we want to add functionality to, e.g., determine which processing durations took too long.

 Is our newly added code correct? Let's run the function with a durationThreshold so large that none of our processing durations take too long ...

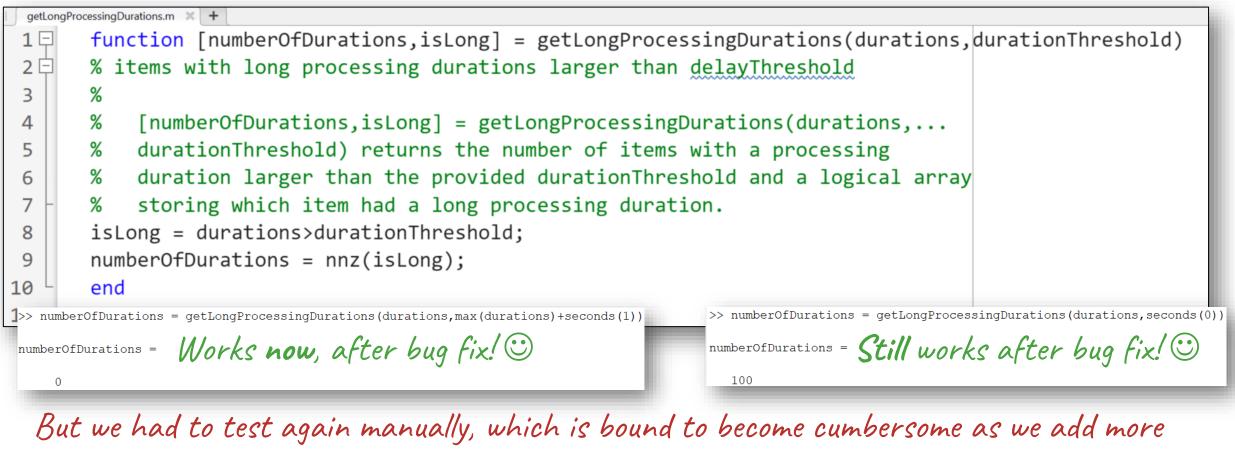
```
getLongProcessingDurations.m 💥 🕂
       function [numberOfDurations, isLong] = getLongProcessingDurations(durations, durationThreshold)
 1 F
 2 E
       % items with long processing durations larger than delayThreshold
 3
       %
           [numberOfDurations, isLong] = getLongProcessingDurations(durations,...
       %
 4
 5
           durationThreshold) returns the number of items with a processing
       %
           duration larger than the provided durationThreshold and a logical array
       %
 6
 7
           storing which item had a long processing duration.
       isLong = durations>durationThreshold;
 8
       numberOfDurations = length(isLong);
 9
10
       end
                                   >> numberOfDurations = getLongProcessingDurations(durations,max(durations)+seconds(1))
11
                                   numberOfDurations =
                                     100
```

length returns the number of all elements in the array instead of how many of them meet the condition ! 13



# Imagine we want to add functionality to, e.g., determine which processing durations took too long.

- Let's change the code to fix the bug ... and recheck everything ...



functionality and need to test for the same (and more) things ... 🟵



#### What if we could automatize testing our code behaves as intended?

 We can use unit tests to test each unit (e.g., function) in our code behaves as intended.

When our code is used in this way,

actualNumberOfDurations = getLongProcessingDurations(durations,seconds(0));

a certain result/behavior expectedNumberOfDurat

expectedNumberOfDurations = length(durations);

should occur.

assert(actualNumberOfDurations == expectedNumberOfDurations)

\* assert throws an error if the tested condition is false.







# We can autogenerate a sample test class with a sample test function directly from the Editor

function [numberOfDurations,isLong] = getLongProcessingDurations(durations,durationThreshold)
% items with long processing durations larger than delayThreshold
%
% [numberOfDurations,isLong] = getLongProcessingDurations(durations,...
Evaluate Selection in Command Window F9
% [numberOfDurations,isLong] = getLongProcessingDurations(durations,...

Help on Selection F1 essing % This is an autogenerated sample test for file getLongProcessingDurations.m Cut Ctrl+X classdef testgetLongProcessingDurations < matlab.unittest.TestCase</pre> Ctrl+C 2 🗆 logical array Paste Ctrl+V 3 Ctrl+J Wrap Comments 4 🗄 methods (Test) Ctrl+R Comment Ctrl+Shift+R Uncomment 5 Duplicate Line(s) Ctrl+Shift+C function test getLongProcessingDurations(testCase) 6 Ctrl+Shift+A 7 🗄 Smart Indent Ctrl+I % Specify the input(s) of Convert to Function % getLongProcessingDurations 8 Convert to Local Function 9 Section Break Ctrl+Alt+Enter Run Section Ctrl+Enter durations = ; 10 Split Screen durationThreshold = ; 11 Create Test Ctrl+Shift+H 12 Requirements 13 🗄 % Specify the expected output(s) of 14 % getLongProcessingDurations 15 expected numberOfDurations = ; 16 17 expected isLong = ; 18 % Exercise the function getLongProcessingDurations 19 [actual numberOfDurations, actual isLong] = getLongProcessingDurations(durations, durationThreshold); 20 21 22 testCase.verifyEqual(actual\_numberOfDurations, expected\_numberOfDurations); 23 testCase.verifyEqual(actual\_isLong, expected\_isLong); 24 end 25 end 26 end

! You can find the correct function in CleanCodePractices>exercise4, and the function and sample test in exercise5



# **Your turn:** Generate a sample test and fill it in to test that all processing durations take longer than 0 seconds

- Go to your version of CleanCodePractices>exercise4, generate a sample test, and fill it in to test that all processing durations take longer than 0 seconds.
- Use the skeleton code provided in CleanCodePractices>exercise5 if you cannot generate a sample test.





#### We can complete the test ...

1	% This is an autogenerated sample test for	file getLongProcessingDurations m
2 📮	classdef testgetLongProcessingDurations <	
3	classici testgettongi ocessingbul acions (	
4 🚍	methods (Test)	
5		
6 d	function test getLongProcessingDur	ations(test(ase)
6 🕂 7 📮	% Specify the input(s) of	1 % This is an autogenerated sample test for file getLongProcessingDurations.m
8 -	% getLongProcessingDurations	
9		2 classdef testGetLongProcessingDurations < matlab.unittest.TestCase
10	durations = ;	3
11	durationThreshold = ;	4 methods (Test)
12		5
13 🖯	% Specify the expected output(	6 🗇 function test_AllLong(testCase)
14	% getLongProcessingDurations	7 🛱 % Specify the input(s) of
15		8 - % getLongProcessingDurations
16	<pre>expected_numberOfDurations = ;;</pre>	9
17	expected_isLong = ;	10 load timestampsCN.mat entryTimestamps exitTimestamps
18		11 durations = exitTimestamps-entryTimestamps;
19	% Exercise the function getLon	<pre>12 durationThreshold = seconds(0);</pre>
20	[actual_numberOfDurations, act	
21	,	
22	testCase.verifyEqual(actual_nu	14 % Specify the expected output(s) of
23	testCase.verifyEqual(actual_is	15 - % getLongProcessingDurations
24 -	end	
25 -	end	<pre>17 expected_numberOfDurations = length(durations);</pre>
26 L	end	18
		19 % Exercise the function getLongProcessingDurations
		<pre>20 [actual_numberOfDurations, ~] = getLongProcessingDurations(durations, durationThreshold);</pre>
		21
		<pre>22 testCase.verifyEqual(actual numberOfDurations, expected numberOfDurations);</pre>
		23 - end



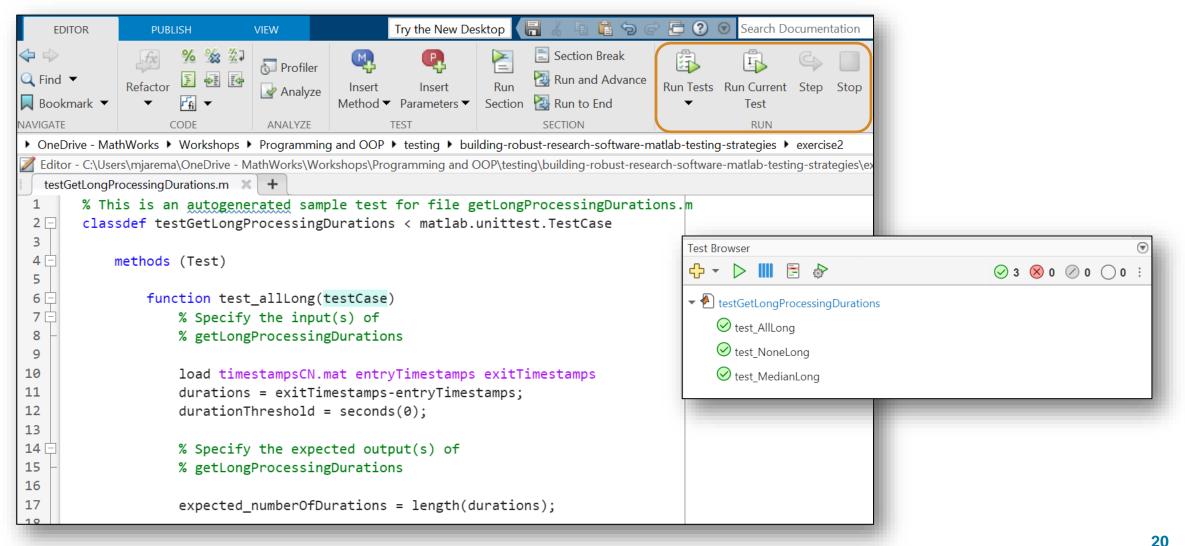
#### We can complete the test and add further test functions

1	% This is an autogenerated sample test for file getLongPro	cessingDurations.m
2 🖵	<pre>classdef testGetLongProcessingDurations &lt; matlab.unittest.</pre>	TestCase
3		
4 🗇	methods (Test)	
5		<pre>function test_NoneLong(testCase)</pre>
6 🗖	<pre>function test_allLong(testCase)</pre>	% Specify the input(s) of
7 🖨	% Specify the input(s) of	% getLongProcessingDurations
8 -	% getLongProcessingDurations	
9		<pre>load timestampsCN.mat entryTimestamps exitTimestamps</pre>
10	<pre>load timestampsCN.mat entryTimestamps exitTime</pre>	<pre>durations = exitTimestamps-entryTimestamps;</pre>
11	<pre>durations = exitTimestamps-entryTimestamps;</pre>	<pre>durationThreshold = max(durations)+seconds(1);</pre>
12	<pre>durationThreshold = seconds(0);</pre>	
13		% Specify the expected output(s) of
14 🖨	% Specify the expected output(s) of	% getLongProcessingDurations
15 -	% getLongProcessingDurations	
16		<pre>expected_numberOfDurations = 0;</pre>
17	<pre>expected_numberOfDurations = length(durations)</pre>	
18		[actual_numberOfDurations,~] = getLongProcessingDurations(durations,
19	% Exercise the function getLongProcessingDurat	durationThreshold);
20	[actual_numberOfDurations, ~] = getLongProcess	
21		% Exercise the function getLongProcessingDurations
22	<pre>testCase.verifyEqual(actual_numberOfDurations,</pre>	<pre>testCase.verifyEqual(actual_numberOfDurations, expected_numberOfDurations);</pre>
23 –	end	end





# We can run tests interactively in the Editor or in the Test Browser app







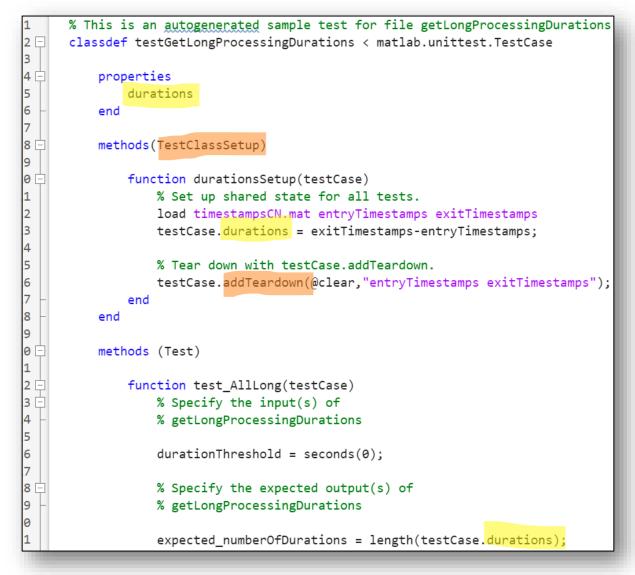
# With the Test Browser we can also interactively collect code coverage

Test Browser       ☑         ↓ ▼       ▶       III       E       III       E       III       IIII       III       IIII       IIIII       IIIII       IIIII       IIIII       IIIII       IIIII       IIIII       IIIII       IIIII       IIIIIIIIII       IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Code Covera The code coverage reputests.			ysis of the s	ource code cov	vered by the		
Enable coverage reporting  Metrics	Overall Coverage Summary of the code cover	-	Il source files.					
Statement Decision Condition MC/DC	Total Files	Coverage	Executable	Missed	Code Cov			
Source	4	Function	1	0	100%			
Add Files Add Folder Remove All orkshops\Programming and OOP\clean-code-practices\exercise6\getLongProcessingDurations	- ·	Statement	2	0	100%			
Currently viewing: Statement Covered Missed Partially Covered								
Report  Open coverage report after run	Breakdown by So Code coverage metrics per	source file.						





# **Tip:** We can extract setup/teardown code in test fixtures and avoid duplicated code





#### Your turn: Add test fixtures for all test functions

• Go to your version of CleanCodePractices>exercise6 and add test fixtures.



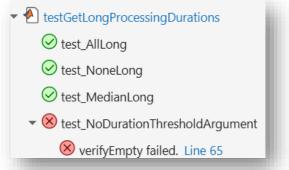
Imagine we'd like to call our function getLongProcessingDurations without a durationThreshold input argument ...

 Let's add a test that should pass if we can run the function without a durationThreshold input argument and get no errors ...

```
function test_NoDurationThresholdArgument(testCase)
    actualException = [];
    try
        getLongProcessingDurations(testCase.durations);
    catch actualException
    end
    testCase.verifyEmpty(actualException);
end
```

! You can find the test in CleanCodePractices>exercise8

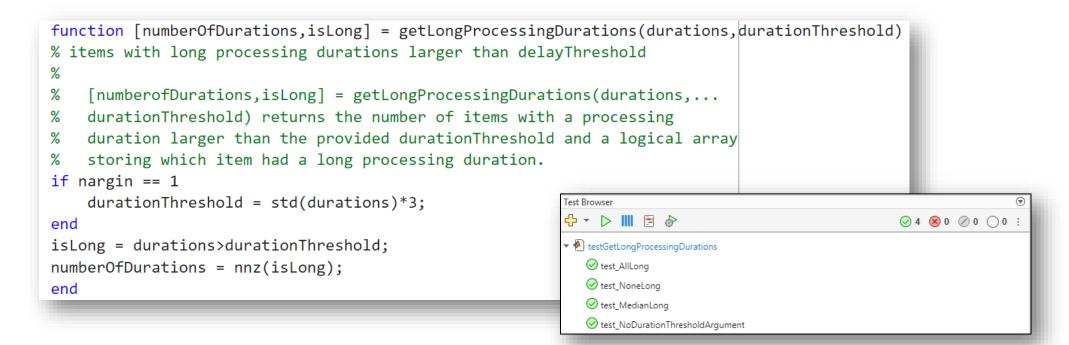
 The new test should fail (because we have yet to change the behavior of our function ...).





# Imagine we'd like to call our function getLongProcessingDurations without a durationThreshold input argument ...

 We could add function code to pass the failed test by checking the number of input arguments and computing a durationThreshold if none is inputted.

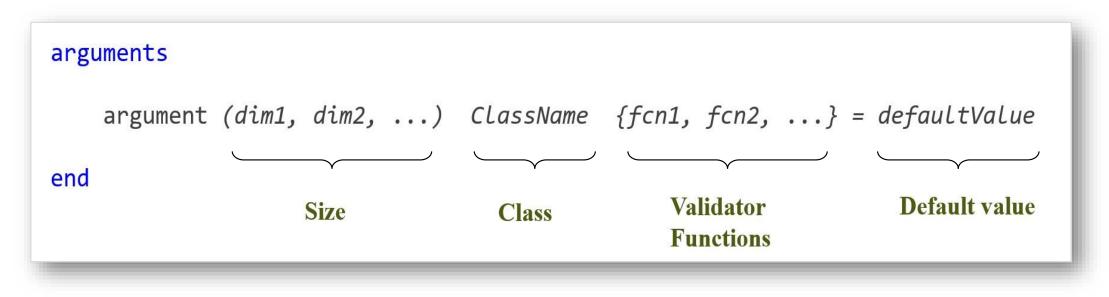


But writing code in the function body to verify input arguments are valid before running function code could make our code harder to read and maintain ...



Imagine we'd like to call our function getLongProcessingDurations without a durationThreshold input argument ...

 We can use function argument validation to declare specific restrictions on input arguments without writing code in the body of the function to check these arguments.





Imagine we'd like to call our function getLongProcessingDurations without a durationThreshold input argument...

 We can use function argument validation to declare specific restrictions on input arguments without writing code in the body of the function to check these arguments.

```
function [numberOfDurations, isLong] = getLongProcessingDurations(durations, durationThreshold)
 1日
       % items with long processing durations larger than delayThreshold
 2 🕀
       %
 3
           [numberOfDurations, isLong] = getLongProcessingDurations(durations,...
       %
 4
           durationThreshold) returns the number of items with a processing
 5
           duration larger than the provided durationThreshold and a logical array
 6
       %
           storing which item had a long processing duration.
       %
 8
       %
           [numberOfDurations, isLong] = getLongProcessingDurations(durations)
 9
       %
           uses a default value for the durationThreshold equal to three times the
10
11
       %
           standard deviation of the durations.
12
       arguments
           durations (:,1) duration
13
           durationThreshold (1,1) duration = std(durations)*3
14
15
       end
16
17
       isLong = durations>durationThreshold;
       numberOfDurations = nnz(isLong);
18
19
       end
```



#### **Organizing and Collaborating – Project Management**



### How can we organize our code and data so everybody can easily continue where we left off?

 We shouldn't assume others have the same hierarchical folder structure or the same operating system, but generate path and file names so they are valid for others and on other platforms:

```
>> f = fullfile("myfolder", "mysubfolder", "myfile")
```

f =

Windows

'myfolder\mysubfolder\myfile.m'

f =
'myfolder/mysubfolder/myfile.m' Linux

```
>> fun_dir = "toolbox" + filesep + "matlab" + filesep + "ionfun"
```

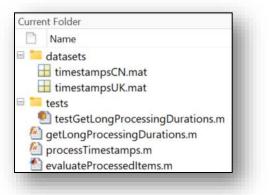
fun\_dir =

"toolbox\matlab\ionfun"



### How can we organize our code and data so everybody can easily continue where we left off?

• We should structure our project code so that it is easy to maintain and build on, e.g., extract datasets and tests in their own folders ...



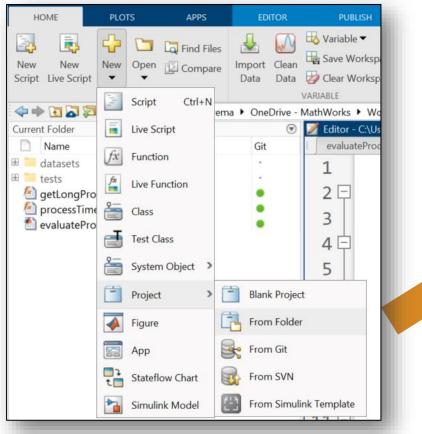
• For more complex projects, we can pick a general standard to follow, e.g.,

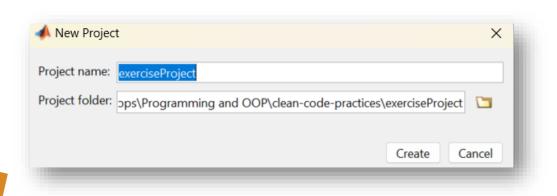
MATLAB Toolbox Best Practices							
version v0.9.1 license CC-BY-4.0							
You have a MATLAB® toolbox that you want to share with the world. We want to help. To do that, we want to convince you to use the MathWorks Toolbox best practices. It's a little bit of extra sector is the sector of the sector.							



### How can we organize our code and data so everybody can easily continue where we left off?

We can create MATLAB Projects from an existing folder to automatically set up the working environment for anyone using the project ...







# We can organize our work with MATLAB Projects so that it is easy to maintain and build on for us and others using our code ...

We can add any required folders to the project path, so any user can access the files within these folders without having to set up anything.

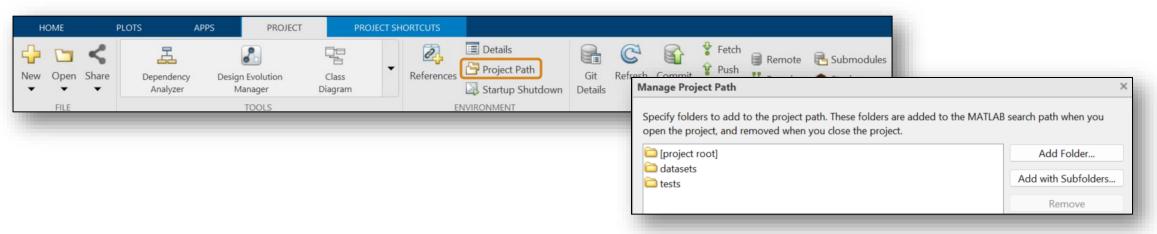
	Welcome to your project	ct		×		
		Let's get start	ed!			
		et up your project to mana ata, and path in one place.				
	6					
	Set	t Paths	Run at Startup			
		Set Up Project				
Set Up Project (Step 1 of 2)	×		S	et Up Project	(Step 2 of 2)	×
Specify folders to add to the project path. These folders are added to the MATLAB se open the project, and removed when you close the project.	arch path when you				t files to automate startup tasks. Startup files automatically run (.m and .p nulink models) when you open the project.	p files), load (.mat files),
C [project root]	Add Folder			Startup files:		
🗀 datasets	Add with Subfolders					•
Contract tests						+
	Remove			Add	Remove	



# We can organize our work with MATLAB Projects so that it is easy to maintain and build on for us and others using our code ...

#### A MATLAB Project manages all our MATLAB files, data files, etc. in one place:

 All relevant project folders have already been added to the project path, so we and all users can access the files within them without having to set up anything. The project path is modifiable later on as well.

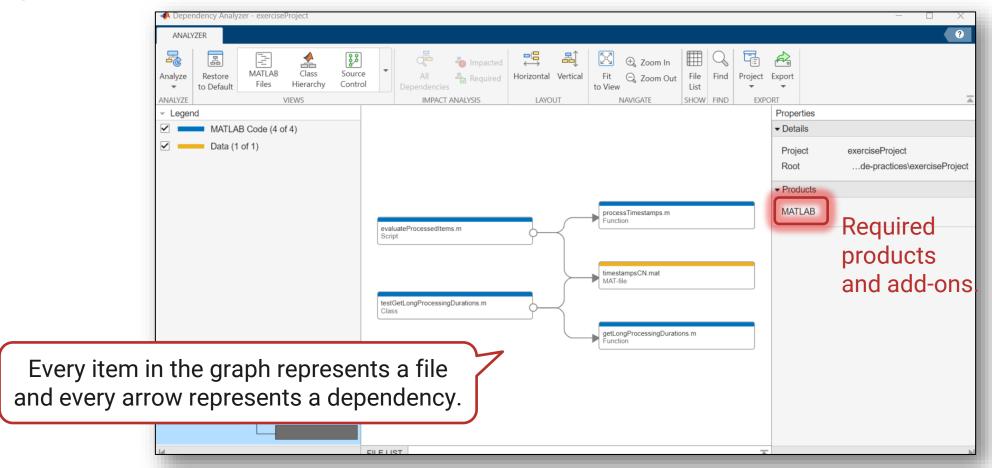


 MATLAB adds the folders to the search path whenever we open the project and removes them from the path when we close the project, so we're always good to go!



# We can organize our work with MATLAB Projects so that it is easy to maintain and build on for us and others ...

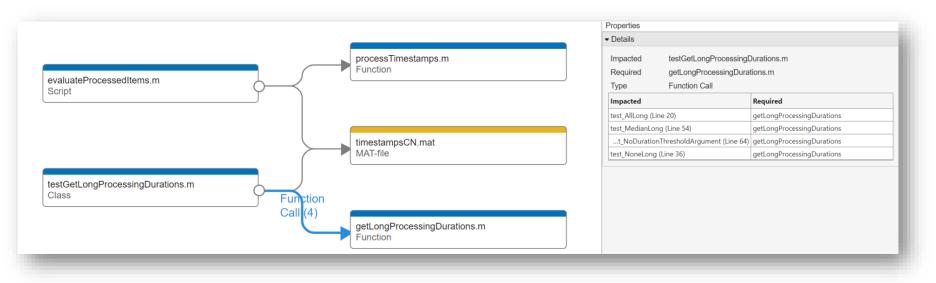
We can also use the **Dependency Analyzer** to see the project structure, dependencies, and how files relate to each other:





# We can organize our work with MATLAB Projects so that it is easy to maintain and build on for us and others ...

File dependencies help us understand better the impact of a code change and identify the tests we need to run to validate the change:



Select dependency arrow to see:

- dependency type (e.g., Function Call)
- where dependency is introduced (e.g., Line 20, etc.)



# We can organize our work with MATLAB Projects so that it is easy to maintain and build on for us and others ...

File dependencies help us understand better the impact of a code change and identify the tests we need to run to validate the change:

Say we refactor our code and think we can safely remove a file. When we **rename**, delete, or remove files or folders in a project, the project runs a dependency analysis to check for effects on other files:

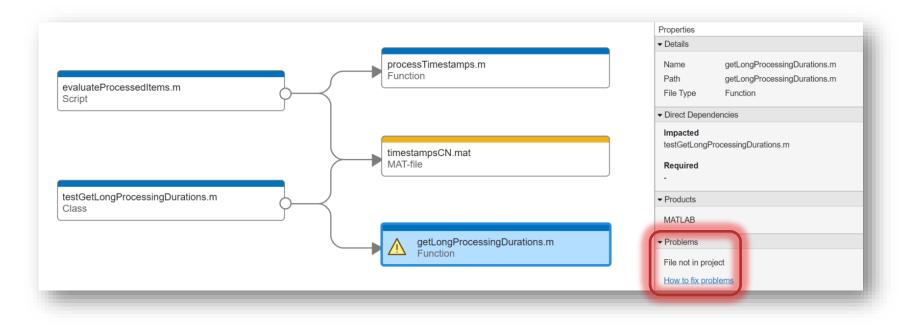
! When **renaming** files, the project offers to automatically update references to the file, which prevents errors that result from changing names or paths manually and overlooking or mistyping the name.

Remove from Project	—		$\times$
1 file depends on the removed files.			
If you remove these files, then the impacted file wil	I no longer function correctly.		
File	Dependencies		
testGetLongProcessingDurations.m	4		
Line 20	MATLAB File		
Line 36	MATLAB File		
Line 54	MATLAB File		
Line 64	MATLAB File		
	Update Remove	Can	cel



File dependencies help us understand better the impact of a code change and identify the tests we need to run to validate the change:

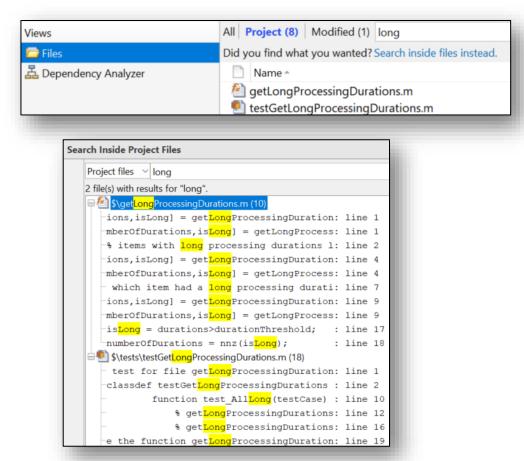
Say we refactor our code and remove a file (despite the warnings). The Dependency Analyzer automatically updates the graph and the **Problems** section:





MATLAB Projects can help in many other ways:

• We can **search** for project files with a specific name:



or content:



MATLAB Projects can help in many other ways:

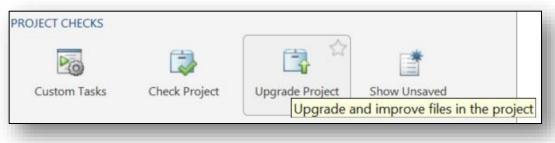
• We can specify startup/shutdown files to run when we open/close our project.



We can use shortcuts to load data, run frequently used files/tests, etc.



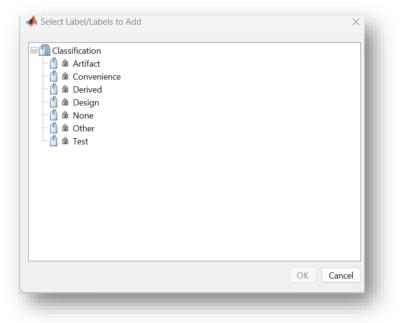
 We can run checks, e.g., to check for compatibility issues or upgrade our project to the current MATLAB release.





MATLAB Projects can help in many other ways:

 We can use one or more labels to organize project files and communicate information to project users:



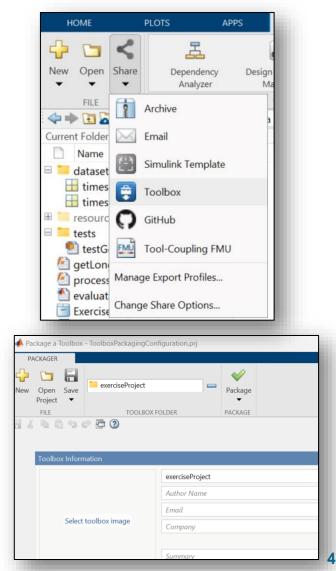
Class-based unit tests are automatically associated with a Test label!



# We can organize our work with MATLAB Projects so that it is easy to maintain and share with others ...

When we are ready to share, we can use source control to collaborate, but there are also other ways:

- We can make our project publicly available on GitHub.
- Suppose we're sharing with somebody who doesn't have access to a source control tool, we can share our project as an **archive**.
- Or we can create a toolbox to share, this also puts everything into one file.





### Your turn: Create a MATLAB project from an existing folder

 Go to your version of CleanCodePractices>exerciseProjectDraft and create a project from the folder.



### **Organizing and Collaborating – Version Control**



### How do we keep track of our code as it changes?

### We can use **source control** instead of maintaining several copies of our code versions:

#### Use source control with Projects:



Jse Git in MATLAE	3:
Git in MATLAB – Functions	\$
Create Repository	
Create Repository	

#### Use source control in Current Folder:

Name	Git	Views			All P
🗉 📒 datasets	1 A A A A A A A A A A A A A A A A A A A	🗁 Files	🧰 Files		Did yo
timestampsCN.mat timestampsUK.mat	:	ឝ Depend	dency A	nalyzer	
E 📁 resources	Open Current Folder	in Explorer			🥂 🙆 g
<ul> <li>tests</li> <li>testGetLongProcessin</li> <li>getLongProcessingDura</li> <li>processTimestamps.m</li> </ul>	New Compare Selected Files/Folders Compare Against		>		
			>		
ExerciseProject.prj	Paste	Ctrl+V		View Details	
	Add to Path		> 🗳	View and Commi	t Changes
	Remove from Path		> 🐇	Fetch	
	Indicate Files Not on	Path	Ŷ	Push	

#### Use source control in Command Window:

#### >> !git status

On branch rse

Your branch is ahead of 'origin/rse' by 3 commits. (use "git push" to publish your local commits)

Changes to be committed:



#### How do we keep track of our project code as it changes?

We can use projects to work with files under source control\*:

APPS     PROJECT     PROJECT SHORTCUTS       Class     Model Testing     Image: Class model Testing     Image: Class model Testing       Diagram     Dashboard     Image: Class model Testing     Image: Class model Testing       TOOLS     Environment     Image: Class model Testing     Image: Class model Testing	Use Source Control SOURCE CONTROM	
Source Control Information		×
Integration: None		
Repository location: N/A		
No source control system detected for the p C:\Users\mjarema\MATLAB Drive\CleanCo	-	raft
Add Project to Source Control	Project - Add to Source Contr	ol
	Source control tool:	Git
	Project root:	C:\Users\mjarema\MATLAB Drive\CleanCodePractices\exercisesProjectDraft
	Source control information:	JGit version 5.0.1 and libgit2 version 1.3.0

\*In this case, local source control, ready to store locally a history of changes we make to our repository



#### How do we keep track of our project code as it changes?

We can use projects to work with files under source control and perform source control operations, such as **committing**<sup>\*</sup> added project files:

ass Model Testing ram Dashboard	<ul> <li>✓ References</li> <li>✓ Environment</li> <li>✓ Environment</li> </ul>	tdown Details	ush Branche	E Submodules	;	
CodePractices    exercise	sProjectDraft > Project - exercisesProjectDraft					
	Views	All Project (16) Modified (78)				
	🗁 Files	🗋 Name 🔺	Status	Classification	Git	
	L Dependency Analyzer	<ul> <li>datasets</li> <li>datevectorsUK.mat</li> <li>timestampsCN.mat</li> <li>timestampsUK.csv</li> <li>timestampsUK.mat</li> </ul>	<ul> <li>Enter</li> </ul>	Design Design a Comment	+ Modified + 4 Added files	
			First con	nmit to local reposi	tory.	
	t snapshot of ou					Submit



#### Your turn: Put your project under local source control

 Go to your version of CleanCodePractices>exerciseProjectDraft and put your project under local source control. If you have a GitHub account, you can share your project to GitHub instead.



#### Bonus: Use GitHub Actions to automate a build and test pipeline

 In your GitHub repository, create a .github/workflows directory and in it a YAML file to define a workflow that will be triggered by a push. The workflow should check out the repository and run tests using the Run MATLAB Tests action.

≡ 🔘 mihaelajarema / awesome-res	earch A
<> Code <ul> <li>Issues</li> <li>Pull requests</li> </ul>	🕞 Actions 🖽 Projects 🕛 Security 🗠 Insights 🕸 Settings
• Files	awesome-research / .github / workflows / generate-test-artifacts.yml
ৃঃ main    +	() mihaelajarema Modify trigger in generate test artifacts workflow
Q Go to file t	Code Blame 16 lines (15 loc) · 467 Bytes 🔀 Code 55% faster with GitHub Cop
generate-test-artifacts.yml	1 name: generate-test-artifacts
generate test artifices.ymi	<pre>2 run-name: Test artifacts generated by \${{ github.actor }}</pre>
> 📄 datasets	3 on: push
> resources	4 jobs:
/ Tesources	5 my-job:
🗋 .gitattributes	6 name: Run MATLAB tests
.gitignore	<pre>7 runs-on: [self-hosted, Windows, X64]</pre>
gragnore	8 steps:
Awesomeresearch.prj	9 - name: Check out repository
	10 uses: actions/checkout@v4
	11 - name: Run tests 12 uses: matlab-actions/run-tests@v2
	12 uses: matlab-actions/run-tests@v2 13 with:
	13 With: 14 test-results-junit: test-results/results.xml
	14 test-results-junit: test-results/results/results.xml 15 code-coverage-cobertura: code-coverage/coverage.xml
	16

• If your repo is public and without transformation products, and you're using a selfhosted runner machine, you can do all this without needing to set up a license!

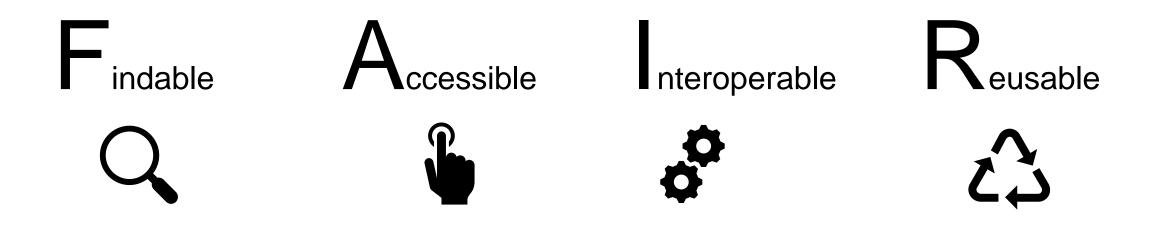


### What's next?



#### So our code is more sustainable, what's next?

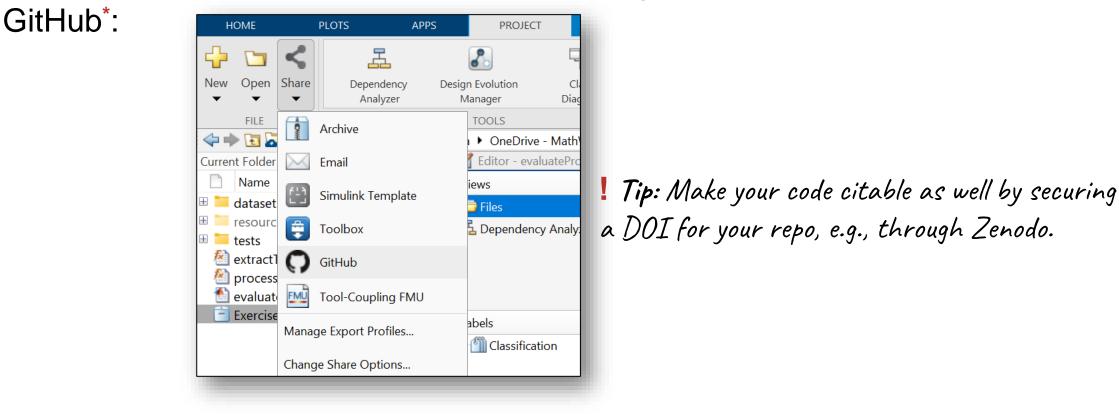
Clean code is easier to understand and **reuse**, provided others can find it and access it irrespective of their programming environments.





#### Findable – the quest for visibility

- State: Our Git repo is potentially local and thus not findable by others.
- Actions to take: Share our project by making it publicly available on



\*Requires a GitHub account



#### Findable – the quest for visibility

- **State:** Research software is available on public GitHub repo.
- Actions to take: Connect GitHub repo to MATLAB Central File Exchange\*
  - + No need to maintain code in two different locations.
  - + Community can discover & install code directly in MATLAB via Add-On Explorer.



#### Accessible – ensuring easy access

- State: Research software is available on public GitHub repo and linked to File Exchange.
- Actions to take: <u>Create link to GitHub repo</u> so anyone can open and view repo in <u>MATLAB Online</u> with just one click.

Author and repository name or URL*	
author-name/repo-name or URL	
File path (optional)	Line number (optional)
Path to a file	#
Project path (optional)	
Path to a project (.prj file)	
Copy the URL below to share a link that opens this repository in MATLAB O	nline
Fill in the fields above to create link	
Copy the markdown below and paste it into your README to display this bu	itton: 🔦 Open in MATLAB Online



#### Access in MATLAB Online

	MATLAB Online (basic)	MATLAB Online
Cost	Free	License*
Version	Online only	Online and desktop
Products included	10 commonly used products Please see FAQ below	All products on your license available in MATLAB Online
Hours per month	20 hours per calendar month	Unlimited
Storage	5 GB	20 GB
Continuous compute time	15 minutes	90 minutes
Inactive idle timeout	15 minutes	60 minutes



# Interoperable – collaboration across formats & programming languages

- **State:** Software is MATLAB-based, but we do leverage MATLAB's support for a wide range of <u>standard file formats</u> to ensure easy data import/export.
- Actions to take: Leverage MATLAB's interoperability with other programming languages to call, e.g., Python code for AI shared by our peers directly in our MATLAB code.

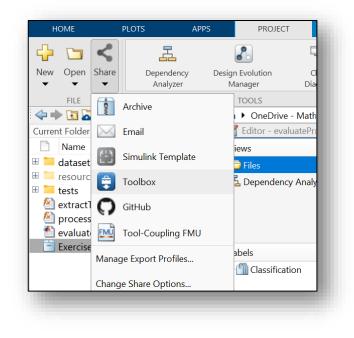


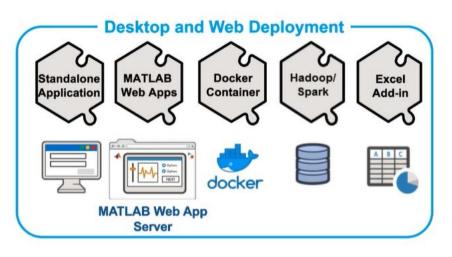




#### Reusable – How else can we improve impact and usability?

- Adopt appropriate open-source licensing for their MATLAB code, encouraging reuse and modification by others.
- Package code into a standalone application so any end user can run code royalty-free using MATLAB Runtime.

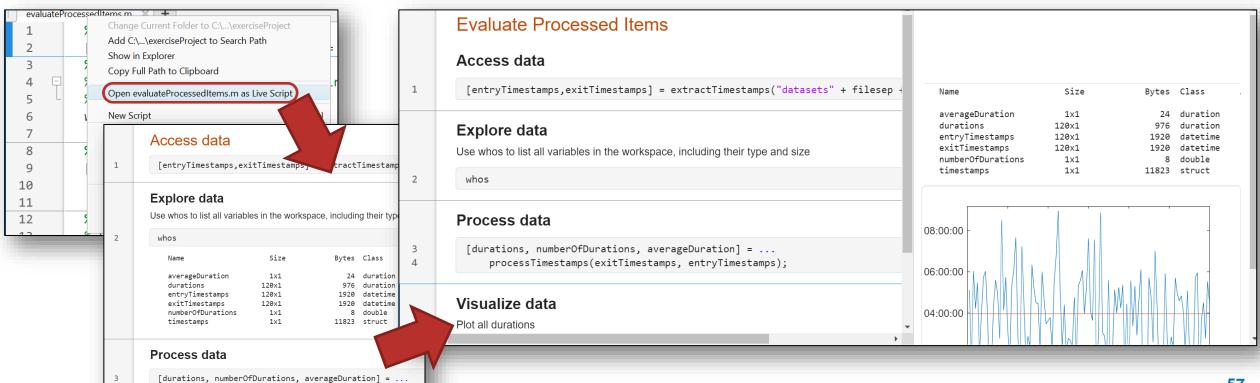






### Reusable – How else can we improve impact and usability?

Enhance documentation using <u>Live Scripts</u> (computational notebooks) by creating interactive tutorials of typical workflows, and sharing them as PDF, Jupyter notebooks, etc. with non-MATLAB users.



processTimestamps(exitTimestamps, entryTimestamps);



#### Open call for working together to support your use cases

Reach out to:

#### Dr. Mihaela Jarema

Academia Group <u>mjarema@mathworks.com</u> <u>https://www.linkedin.com/in/mihaelajarema/</u>

