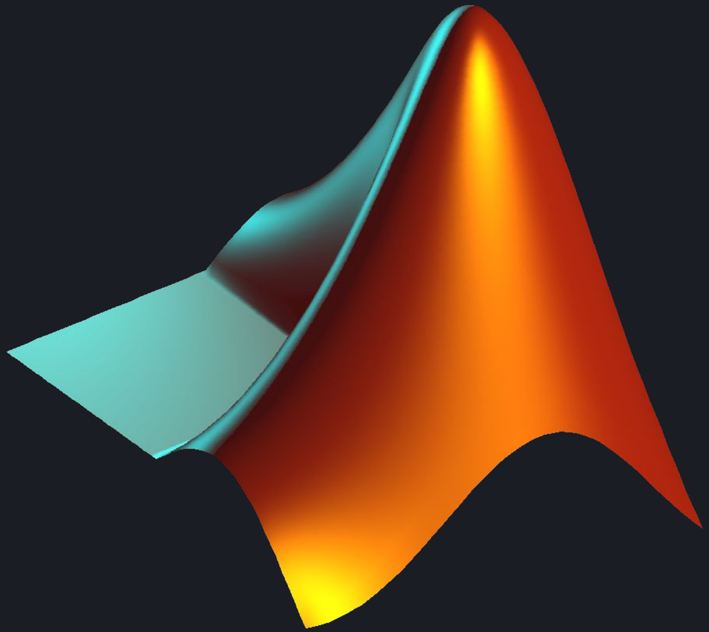
Matlab Scripting Module



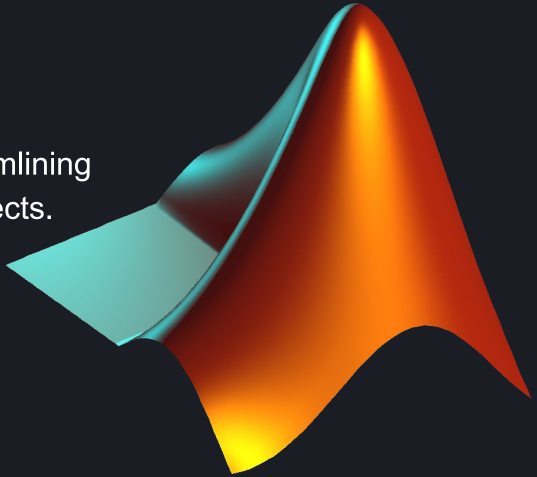
OS-Academy

Model-Based Development Program

Matlab Scripting Module

Objectives

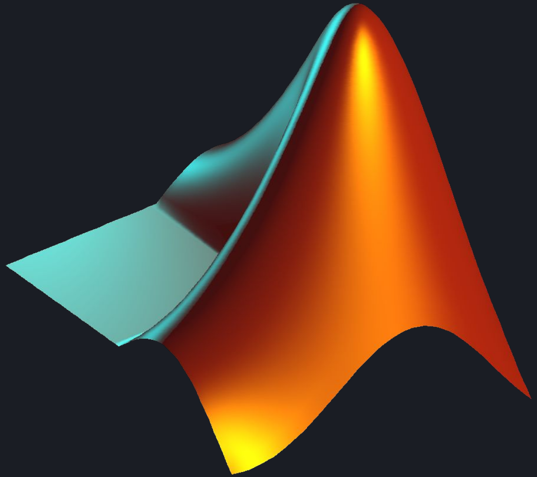
- Develop proficiency in creating MATLAB scripts and functions to enhance programming skills and facilitate efficient code organization.
- Acquire the ability to automate repetitive modeling and simulation tasks, streamlining workflow processes and improving overall productivity in MATLAB-based projects.



Matlab Scripting Module

Module Content

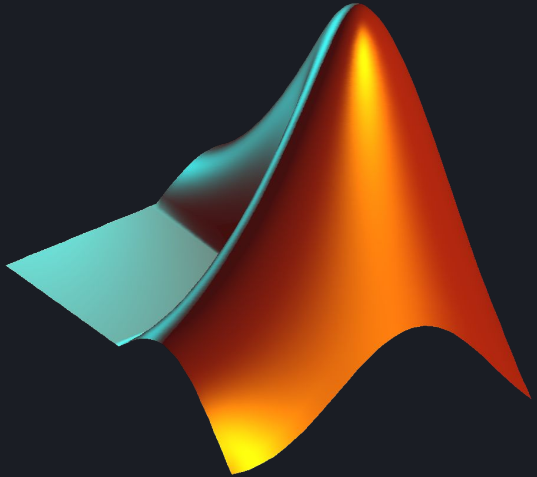
- Automate repetitive Modeling and Simulation Tasks
- Text Manipulation
- File handling



Matlab Scripting Module

Session Content

- Automate repetitive Modeling and Simulation Tasks

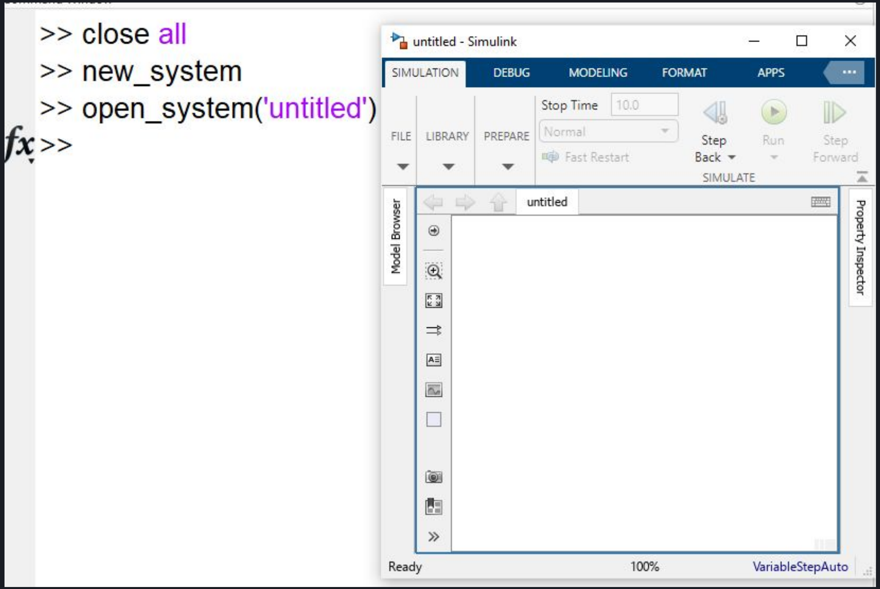


Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

new_system()

- `new_system` creates a model named `untitled`
- The function may return a handle or identifier to the newly created system or object, depending on its implementation.
- The `new_system` function does not open the new model.
- This function creates the model in memory.
- To save the model, use `save_system`, or open the model with `open_system` and then save it using the Simulink Editor.



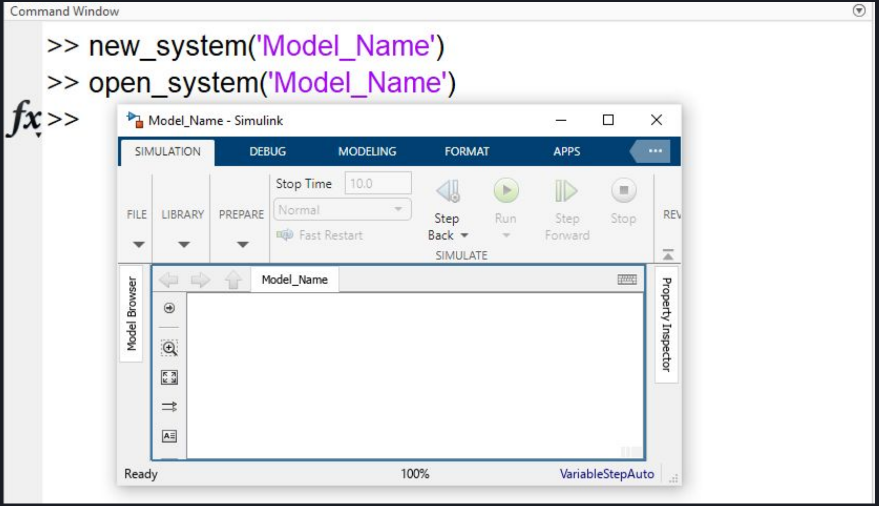
```
>> close all
>> new_system
>> open_system('untitled')
fx >>
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

new_system(name)

- creates a model based on your default model template and gives the new model the specified name.
- The `new_system` function does not open the new model.
- This function creates the model in memory.
- To save the model, use `save_system`, or open the model with `open_system` and then save it using the Simulink Editor.
- To avoid shadowing, if `name` is empty, the `new_system` function checks loaded models and files on the path and creates a model with the next available name `untitled`, `untitled1`, `untitled2`, and so on.
- This function returns the new model's numeric handle.



```
>> new_system('Model_Name')
>> open_system('Model_Name')
>>
```

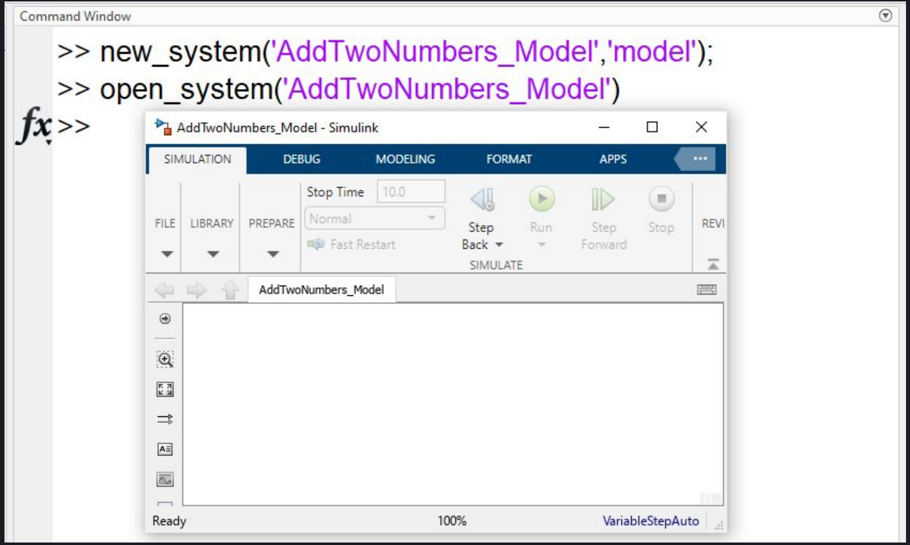
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

new_system(name,model)

- creates an empty model based on the Simulink default model and returns the new model's numeric handle. The Simulink default model is also known as the root block diagram and has the numeric handle `0`. If `name` is empty, the function creates a model or library named `untitled`, `untitled1`, `untitled2`, and so on.
- The `new_system` function does not open the new model. This function creates the model in memory. To save the model, use `save_system`, or open the model with `open_system` and then save it using the Simulink Editor.



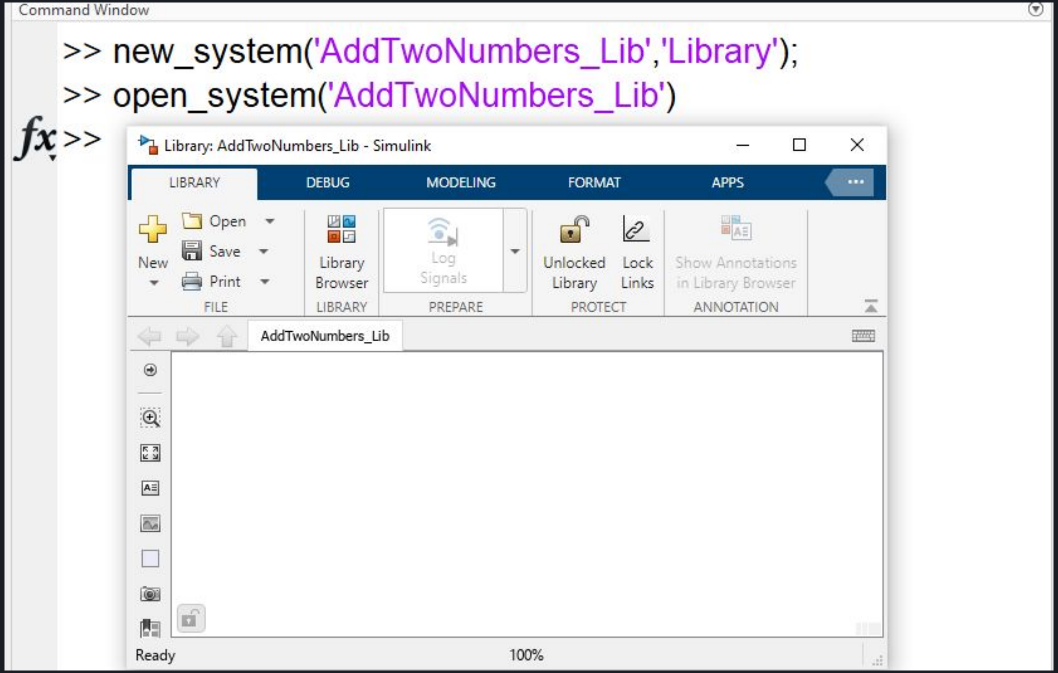
```
>> new_system('AddTwoNumbers_Model','model');
>> open_system('AddTwoNumbers_Model')
>>
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

new_system(name,library)

- creates an empty library that has the specified name and returns a numeric handle.



```
>> new_system('AddTwoNumbers_Lib','Library');
>> open_system('AddTwoNumbers_Lib')
fx >>
```

Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise

Create a MATLAB script that allows users to create either a Simulink model or a Simulink library using the `new_system` function. The script should prompt the user to choose between creating a model or a library, ask for the name of the model or library to be created, and then display a message confirming the creation.

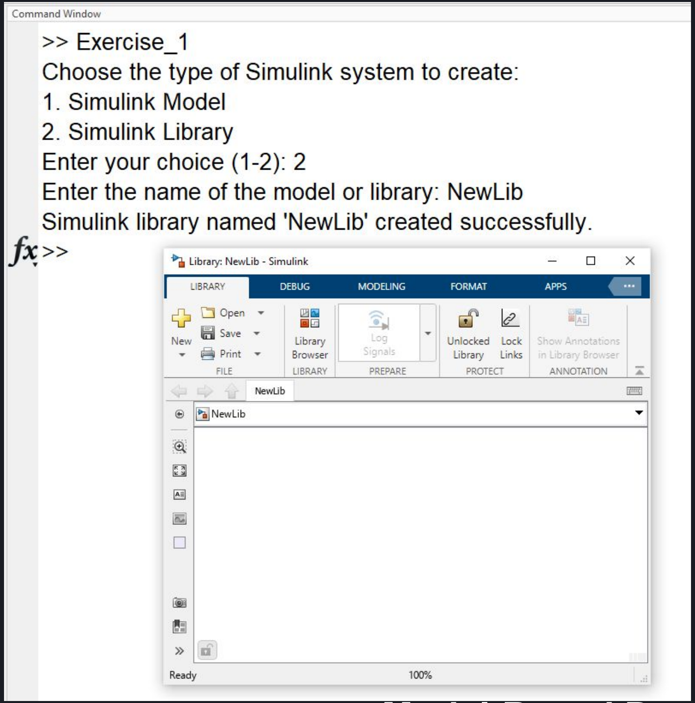
Here's how you can structure the exercise:

- Prompt the user to choose between creating a model or a library.
- Based on the user's choice, ask for the name of the model or library to be created.
- Use the `new_system` function to create the specified model or library.
- Display a message confirming the creation of the model or library.

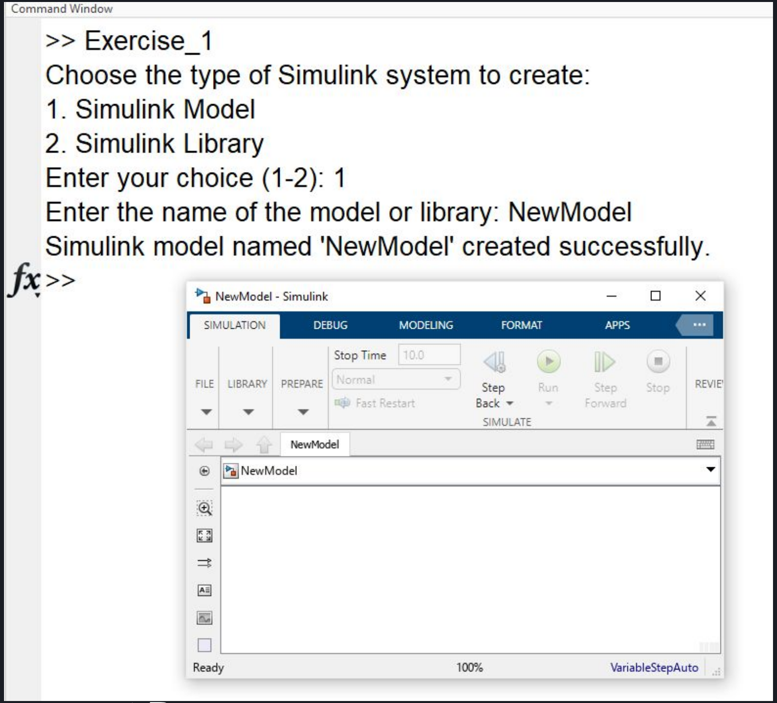
Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise



```
>> Exercise_1
Choose the type of Simulink system to create:
1. Simulink Model
2. Simulink Library
Enter your choice (1-2): 2
Enter the name of the model or library: NewLib
Simulink library named 'NewLib' created successfully.
>>
```



```
>> Exercise_1
Choose the type of Simulink system to create:
1. Simulink Model
2. Simulink Library
Enter your choice (1-2): 1
Enter the name of the model or library: NewModel
Simulink model named 'NewModel' created successfully.
>>
```

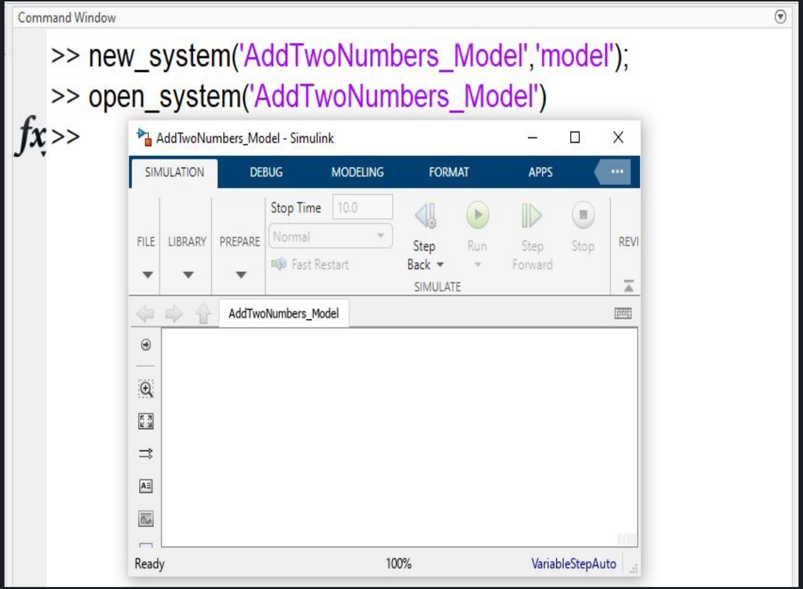
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

open_system(‘modelName’)

- The `open_system` function in MATLAB is used to open Simulink models or subsystems. It is particularly useful when you want to programmatically open a Simulink model for further inspection or modification.
- `'modelName'`: The name of the Simulink model to open. If the model is not currently open, `open_system` will open it.
- If `'modelName'` is provided, `open_system` will open the specified Simulink model in the Simulink editor, displaying its contents.
- If the model or block is successfully opened, `open_system` returns a numeric handle to the model or block. This handle can be used for further programmatic interactions with the model or block.



```
>> new_system('AddTwoNumbers_Model','model');
>> open_system('AddTwoNumbers_Model')
>>
```

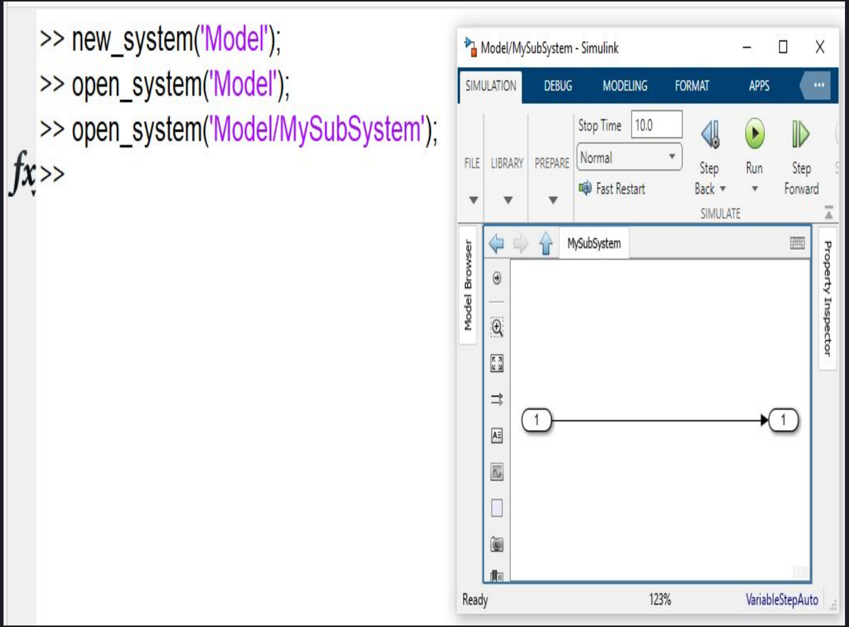
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

open_system('modelName/BlockName')

- Here, the `open_system` function is used to open a specific subsystem ('MySubsystem') within the 'Model''. This is useful when you want to focus on a particular part of a large model.
- `'blockName'` (optional): The name of a specific block within the Simulink model to open. This allows you to focus on a particular block within the model.
- If `'blockName'` is provided along with the model name, `open_system` will open the specified block within the model, focusing on that block while displaying the rest of the model.
- If the model or block is successfully opened, `open_system` returns a numeric handle to the model or block. This handle can be used for further programmatic interactions with the model or block.



```
>> new_system('Model');
>> open_system('Model');
>> open_system('Model/MySubSystem');
fx >>
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise

Create a MATLAB script that utilizes the `open_system` function to open Simulink models and specific blocks within those models. The script should allow users to choose between opening an entire Simulink model or focusing on a specific block within the model. After opening the model or block, the script should display a message confirming the action.

Here's how you can structure the exercise:

- Prompt the user to choose between opening an entire Simulink model or a specific block within a model.
- If the user chooses to open a specific block, prompt them to enter the name of the block.
- Use the `open_system` function to open the selected Simulink model or block.
- Display a message confirming the action.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_block('blockType','modelName/blockName')

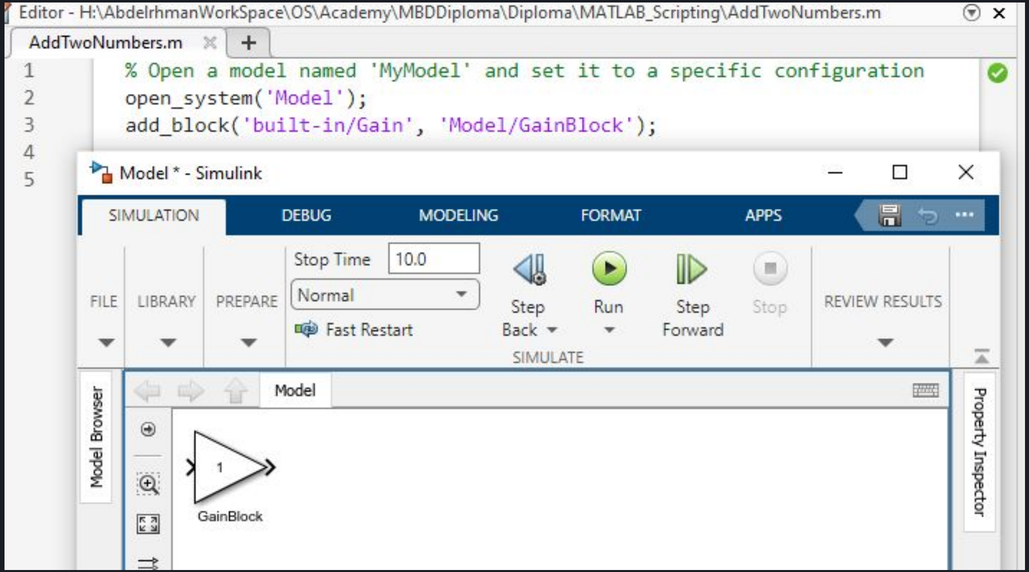
- In MATLAB Simulink environment, the `add_block` function is used to programmatically add blocks to a Simulink model. This function is particularly useful when you want to automate the process of building or modifying Simulink models.
- `'blockType'`: A string specifying the type of block to be added. This can be any valid Simulink block type, such as 'Gain', 'Sum', 'Scope', etc.
- `'modelName/blockName'`: The name of the Simulink model to which the block will be added, followed by the name to assign to the block within the model.
- The `add_block` function adds the specified block to the Simulink model specified by `'modelName'`, creating it if it doesn't already exist.
- If the block name is not specified, Simulink automatically assigns a name to the block based on its type and position within the model.
- The function returns a handle to the added block, which can be used for further programmatic interactions with the block.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_block('blockType','modelName/blockName')

In this example, the `add_block` function is used to add a Gain block to an existing Simulink model named 'Model'. The block is placed in the model and is named 'GainBlock'.



Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\AddTwoNumbers.m

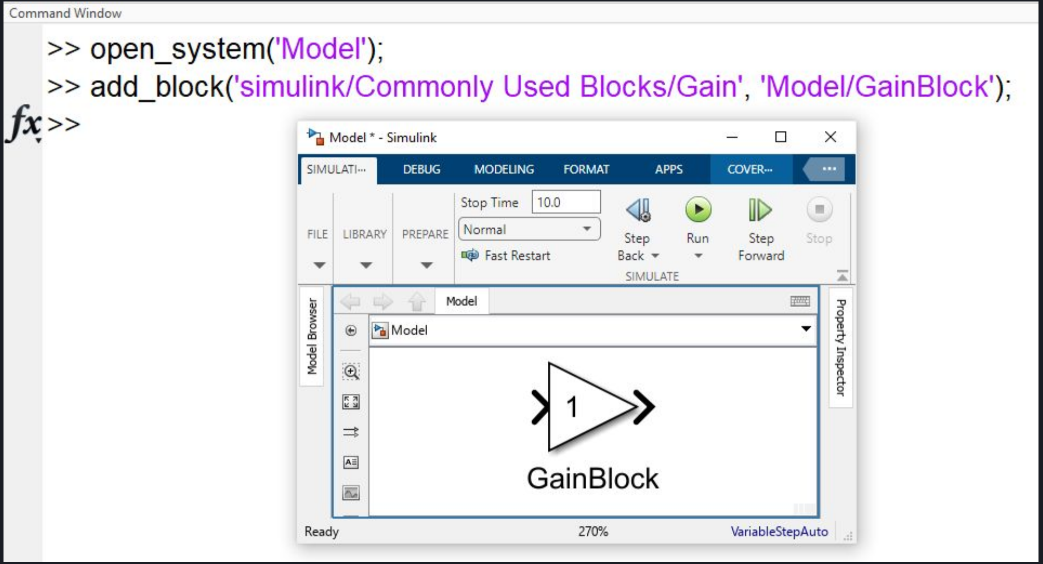
```
% Open a model named 'MyModel' and set it to a specific configuration
open_system('Model');
add_block('built-in/Gain', 'Model/GainBlock');
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_block('blockType','modelName/blockName')

In this example, the `add_block` function is used to add a Gain block to an existing Simulink model named 'Model'. The block is placed in the model and is named 'GainBlock'.



Command Window

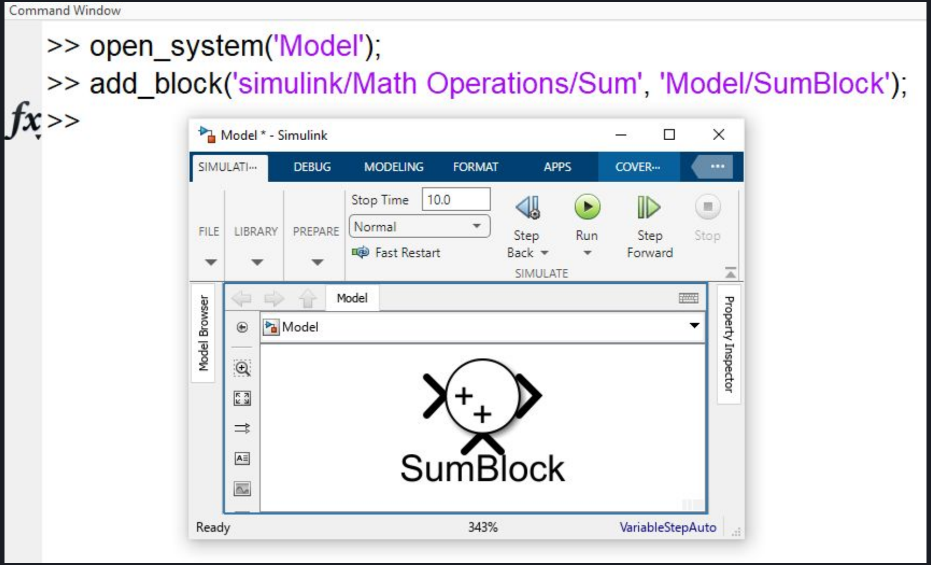
```
>> open_system('Model');
>> add_block('simulink/Commonly Used Blocks/Gain', 'Model/GainBlock');
>>
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_block(‘blockType’,’modelName/blockName’)

In this example, the `add_block` function is used to add a Sum block to an existing Simulink model named 'Model'. The block is placed in the model and is named 'SumBlock'.



```
>> open_system('Model');
>> add_block('simulink/Math Operations/Sum', 'Model/SumBlock');
>>
```

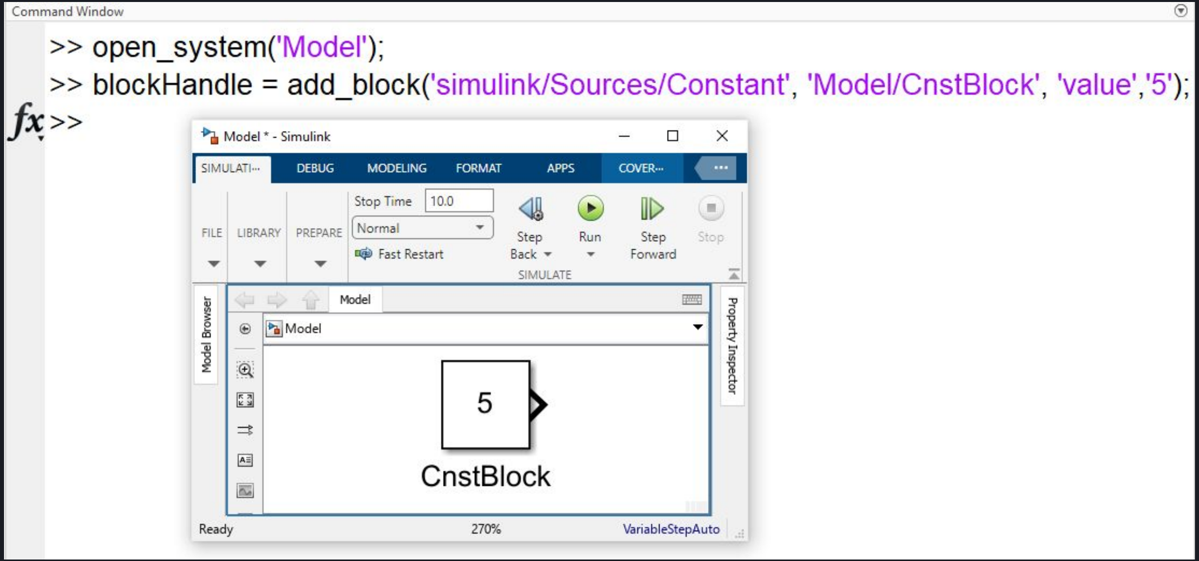
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_block(‘blockType’,’modelName/blockName’)

In this example, the `add_block` function is used to add a constant block to an existing model ('Model'). The block parameters (value) is customized during block creation.



```
>> open_system('Model');
>> blockHandle = add_block('simulink/Sources/Constant', 'Model/CnstBlock', 'value','5');
fx >>
```

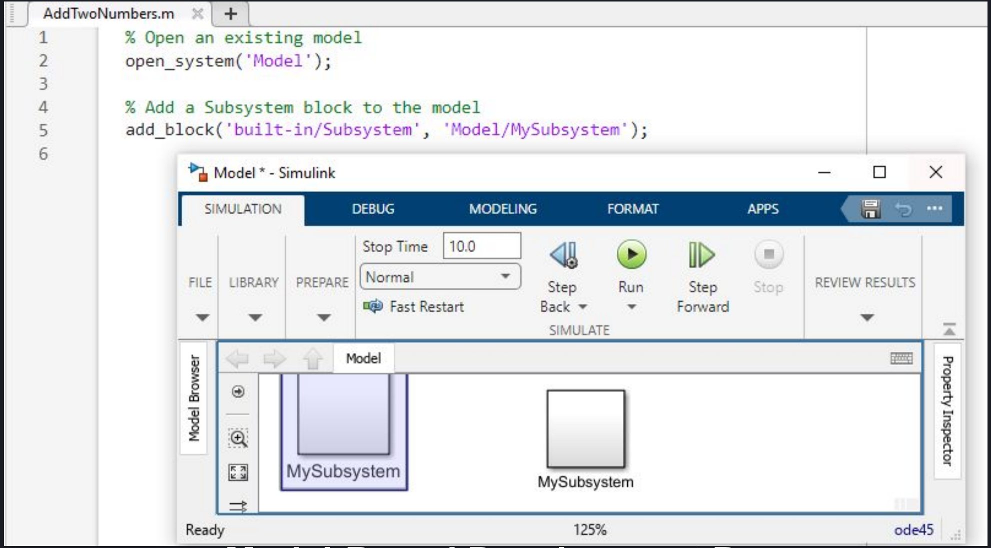
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_block('blockType','modelName/blockName')

In this example, the `add_block` function is used to add a Subsystem block to an existing model ('Model'). This Subsystem block can be populated with additional blocks to create a modular and organized structure within the model.



AddTwoNumbers.m

```
% Open an existing model
open_system('Model');

% Add a Subsystem block to the model
add_block('built-in/Subsystem', 'Model/MySubsystem');
```

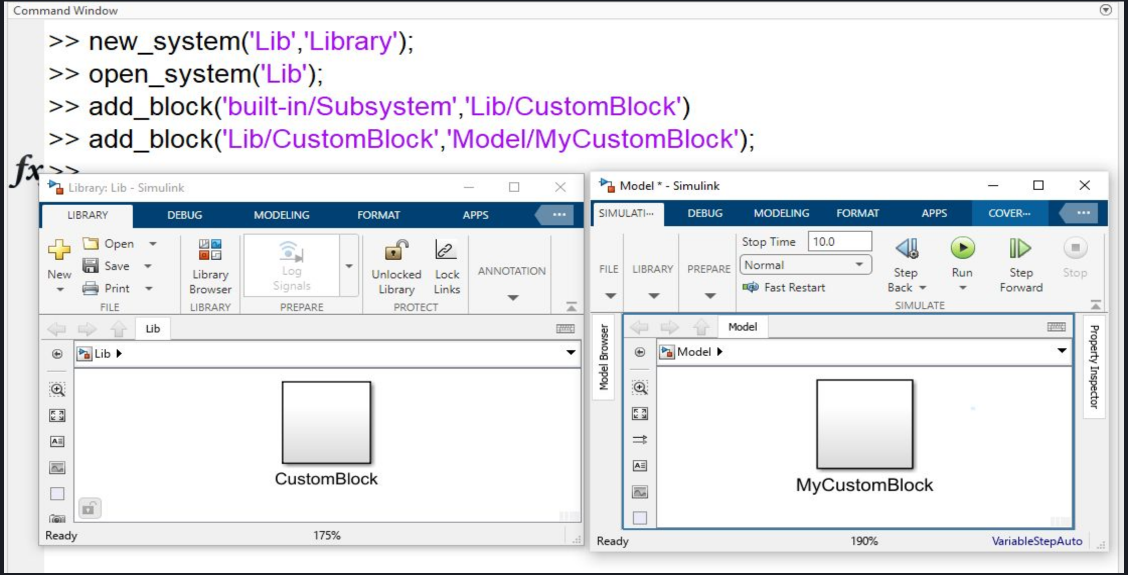
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_block(‘blockType’,’modelName/blockName’)

If 'Lib' is a user-defined Simulink library, this command adds a custom block named 'CustomBlock' from that library to the Simulink model 'Model'.



Command Window

```
>> new_system('Lib','Library');
>> open_system('Lib');
>> add_block('built-in/Subsystem','Lib/CustomBlock')
>> add_block('Lib/CustomBlock','Model/MyCustomBlock');
fx >>
```

OS-Academy

Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise

Create a MATLAB script that utilizes the `add_block` function to programmatically build a simple Simulink model. The script should allow users to specify the types of blocks to be added to the model. Users should also provide the name of an existing Simulink model where the blocks will be added. After adding the blocks, the script should open the Simulink model and display a message confirming the creation.

Here's how you can structure the exercise:

Prompt the user to specify the name of an existing Simulink model where the blocks will be added.
Prompt the user to specify the types of blocks they want to add to the model.
Use the `add_block` function to add each block to the specified Simulink model.
Open the Simulink model specified by the user.
Display a message confirming the addition of blocks to the model and the name of the model opened.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_line(‘modelName’,’sourceBlock’,’destinationBlock’)

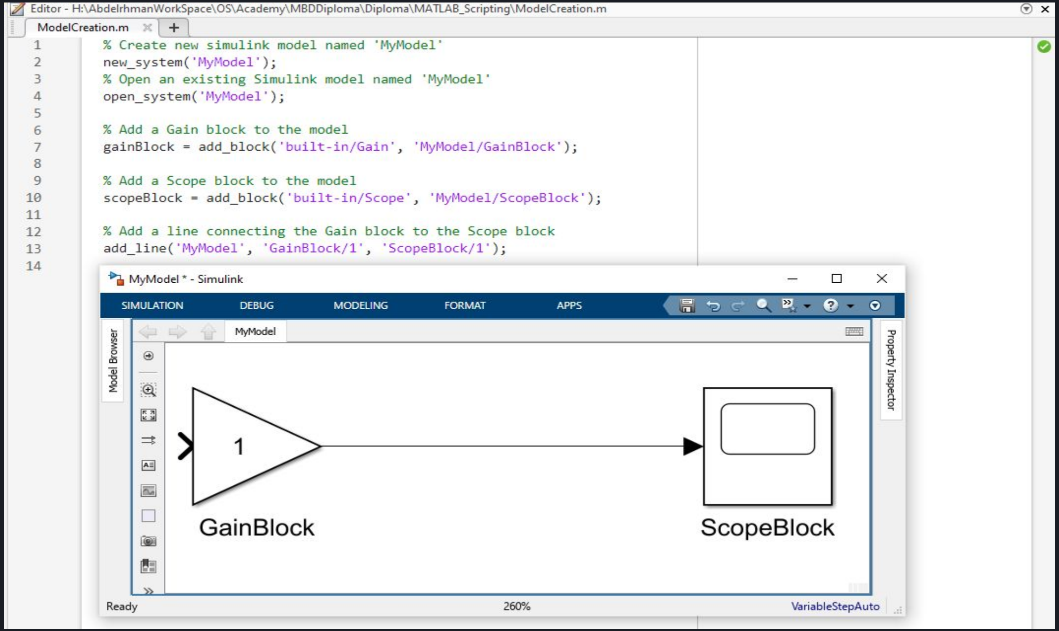
- In MATLAB's Simulink environment, the `add_line` function is used to programmatically connect blocks within a Simulink model with lines, representing signal flow or data flow between them. This function is useful for automating the process of building or modifying Simulink models.
- `'modelName'`: The name of the Simulink model where the blocks are located.
- `'sourceBlock'`: The name of the source block from which the line originates.
- `'destinationBlock'`: The name of the destination block to which the line connects.
- The `add_line` function creates a connection (line) between the specified source and destination blocks within the Simulink model.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_line('modelName','sourceBlock','destinationBlock')

In this example, the `add_block` function is used to add a Gain block and a Scope block to an existing Simulink model ('MyModel'). Then, the `add_line` function is used to connect the output port of the Gain block ('GainBlock/1') to the input port of the Scope block ('ScopeBlock/1').



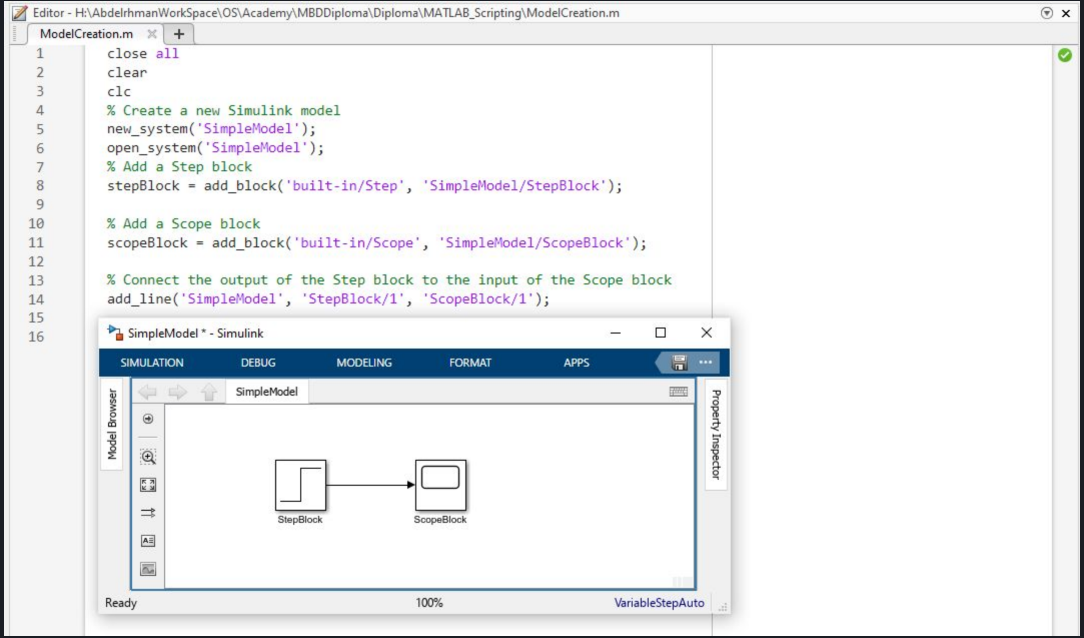
```
% Create new simulink model named 'MyModel'
new_system('MyModel');
% Open an existing Simulink model named 'MyModel'
open_system('MyModel');

% Add a Gain block to the model
gainBlock = add_block('built-in/Gain', 'MyModel/GainBlock');

% Add a Scope block to the model
scopeBlock = add_block('built-in/Scope', 'MyModel/ScopeBlock');

% Add a line connecting the Gain block to the Scope block
add_line('MyModel', 'GainBlock/1', 'ScopeBlock/1');
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_line(‘modelName’,’sourceBlock’,’destinationBlock’)

In this example, a Step block and a Scope block are added to a new Simulink model, and the `add_line` function is used to connect the output of the Step block to the input of the Scope block.



Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

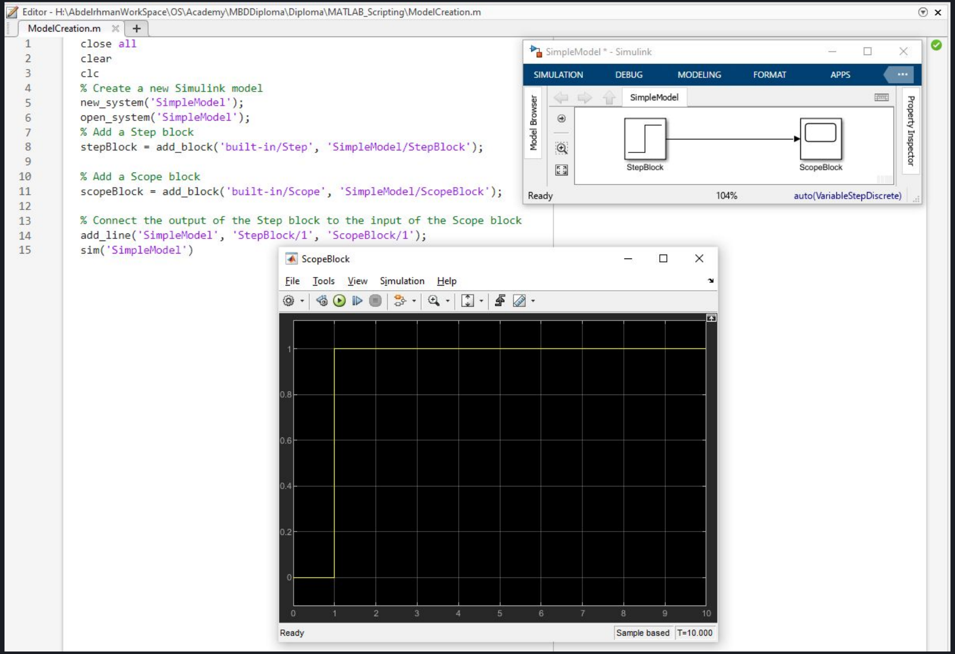
```matlab
close all
clear
clc
% Create a new Simulink model
new_system('SimpleModel');
open_system('SimpleModel');
% Add a Step block
stepBlock = add_block('built-in/Step', 'SimpleModel/StepBlock');

% Add a Scope block
scopeBlock = add_block('built-in/Scope', 'SimpleModel/ScopeBlock');

% Connect the output of the Step block to the input of the Scope block
add_line('SimpleModel', 'StepBlock/1', 'ScopeBlock/1');
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_line('modelName','sourceBlock','destinationBlock')

In this example, a Step block and a Scope block are added to a new Simulink model, and the `add_line` function is used to connect the output of the Step block to the input of the Scope block.



```
close all
clear
clc
% Create a new Simulink model
new_system('SimpleModel');
open_system('SimpleModel');
% Add a Step block
stepBlock = add_block('built-in/Step', 'SimpleModel/StepBlock');

% Add a Scope block
scopeBlock = add_block('built-in/Scope', 'SimpleModel/ScopeBlock');

% Connect the output of the Step block to the input of the Scope block
add_line('SimpleModel', 'StepBlock/1', 'ScopeBlock/1');
sim('SimpleModel')
```

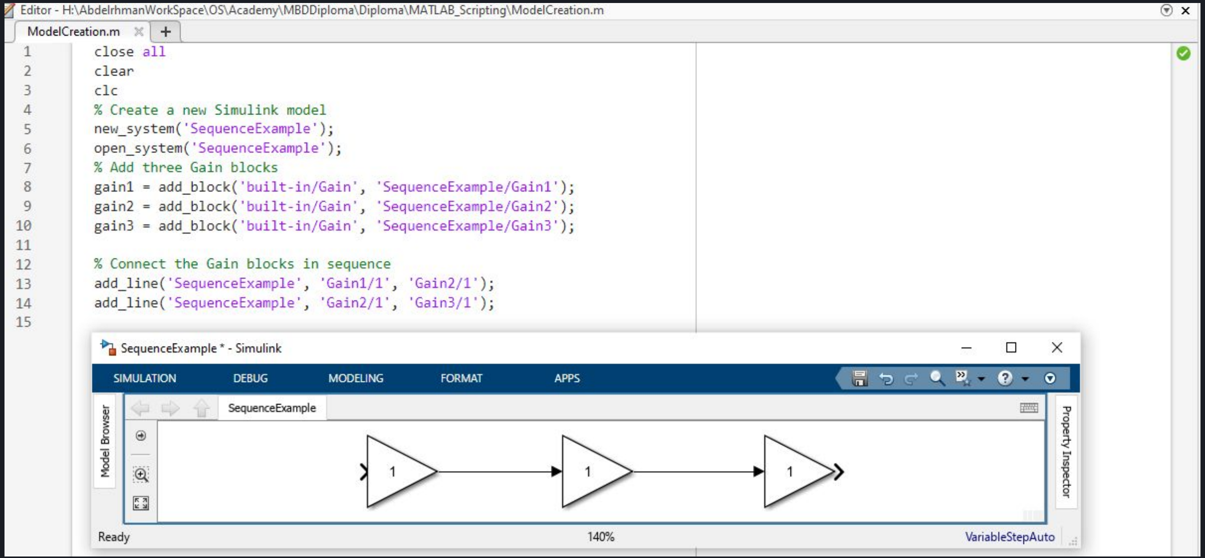
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

add_line('modelName','sourceBlock','destinationBlock')

This example shows how to connect multiple blocks in a sequence. Three Gain blocks are added, and the `add_line` function is used to connect them in a linear sequence.



Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

```
close all
clear
clc
% Create a new Simulink model
new_system('SequenceExample');
open_system('SequenceExample');
% Add three Gain blocks
gain1 = add_block('built-in/Gain', 'SequenceExample/Gain1');
gain2 = add_block('built-in/Gain', 'SequenceExample/Gain2');
gain3 = add_block('built-in/Gain', 'SequenceExample/Gain3');

% Connect the Gain blocks in sequence
add_line('SequenceExample', 'Gain1/1', 'Gain2/1');
add_line('SequenceExample', 'Gain2/1', 'Gain3/1');
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise

Create a MATLAB script that demonstrates the use of various Simulink functions to programmatically create a simple Simulink model. The script should cover the following functions: `new_system`, `open_system`, `add_block`, and `add_line`.

Here's how you can structure the exercise:

Use the `new_system` function to create a new Simulink model.
Use the `add_block` function to add a Step Input block and a Scope block to the model.
Use the `add_line` function to connect the Step Input block to the Scope block within the model.
Use the `open_system` function to open the created Simulink model.
Run the Simulink model.
Open the Scope block to view the result.
Display a message confirming the creation of the model, the connections between blocks, and instructing the user to check the result in the Scope block.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

gcb

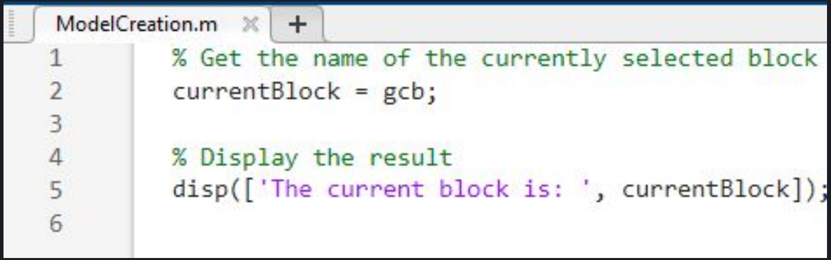
- The `gcb` function in MATLAB's Simulink environment stands for "get current block." It is used to retrieve the full hierarchical name of the currently executing block within a Simulink model.
- Purpose: Retrieve the full hierarchical name of the currently executing block within a Simulink model.
- `Output:blockName`: A character vector containing the full hierarchical name of the currently executing block.
- The `gcb` function is typically used within Simulink callback functions or within a MATLAB Function block to obtain information about the block's location within the model hierarchy.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

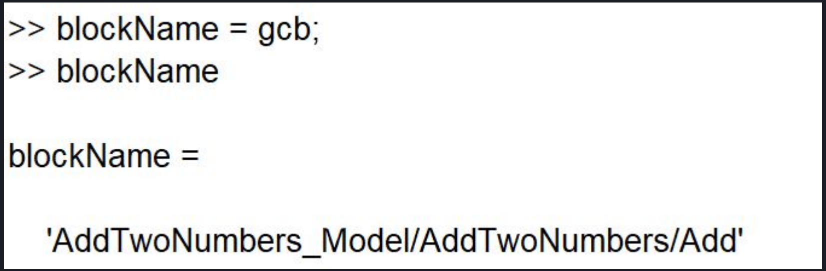
gcb

The `gcb` function in Simulink, which stands for "get current block," is a MATLAB command used to obtain the name of the currently selected block within the Simulink environment. This function is particularly useful when you need to programmatically reference or interact with the block that is currently selected by the user in the Simulink model.



ModelCreation.m

```
% Get the name of the currently selected block
currentBlock = gcb;

% Display the result
disp(['The current block is: ', currentBlock]);

```



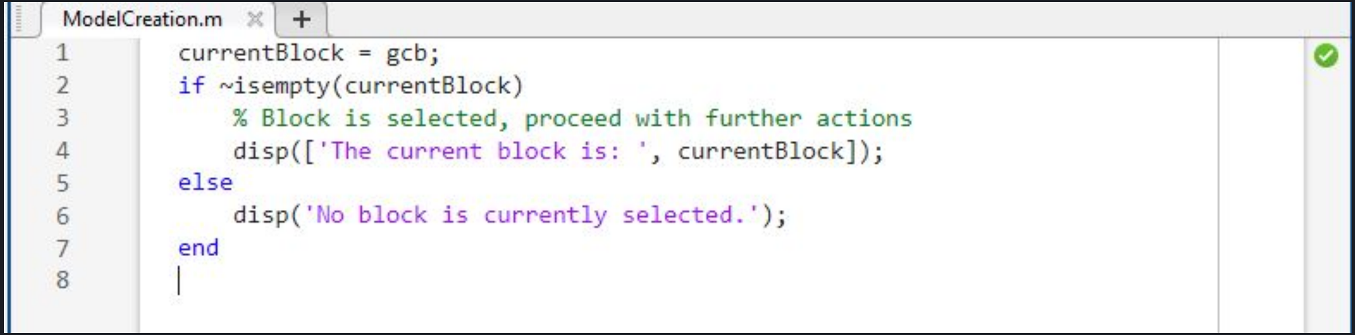
```
>> blockName = gcb;
>> blockName

blockName =

    'AddTwoNumbers_Model/AddTwoNumbers/Add'
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

gcb

Keep in mind that `gcb` returns an empty string (`''`) if no block is currently selected, so it's a good practice to check whether a block is selected before attempting to use its name.



```
currentBlock = gcb;
if ~isempty(currentBlock)
    % Block is selected, proceed with further actions
    disp(['The current block is: ', currentBlock]);
else
    disp('No block is currently selected.');
end
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

gcbh

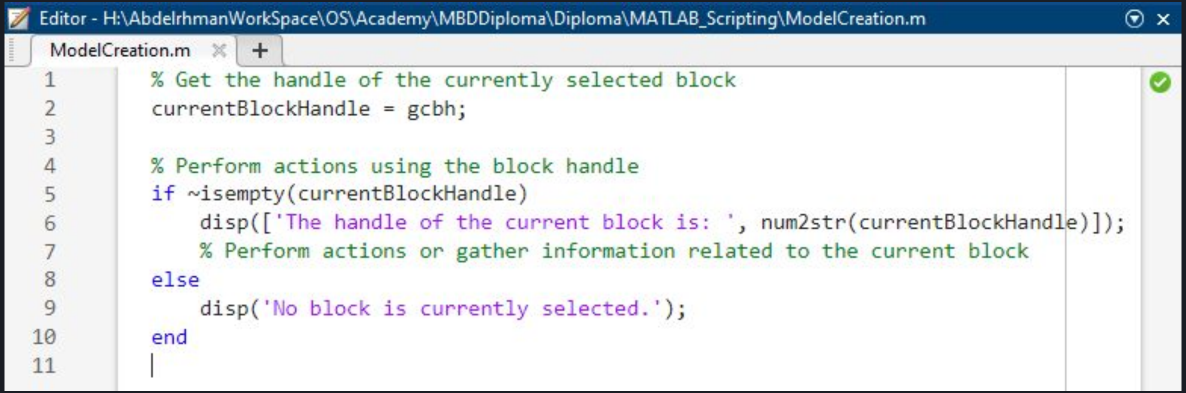
- In MATLAB's Simulink environment, the `gcbh` function stands for "get current block handle." It is used to retrieve the handle of the currently executing block within a Simulink model.
- Purpose: Retrieve the handle of the currently executing block within a Simulink model.
- `Output:blockHandle`: A numeric handle representing the currently executing block within the Simulink model.
- The `gcbh` function is typically used within Simulink callback functions or within a MATLAB Function block to obtain a handle to the block. This handle can then be used to access and manipulate properties of the block programmatically.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

gcb

The `gcbh` function in Simulink stands for "get current block handle." It is used to obtain the handle of the currently selected block within the Simulink environment. This function returns the Simulink block handle, which can be used for programmatically manipulating or interacting with the selected block.


Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m
ModelCreation.m

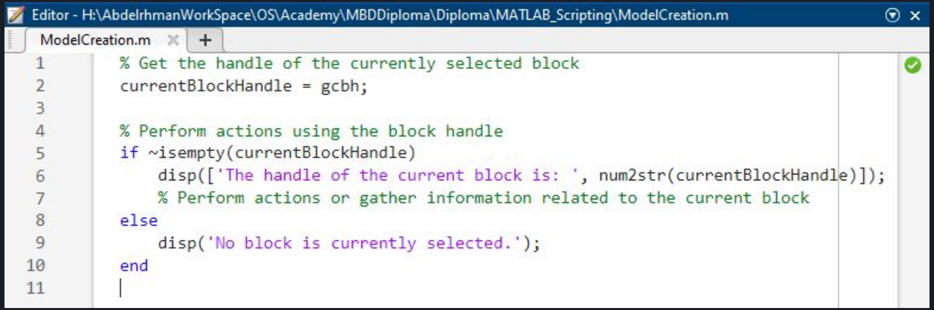
```
% Get the handle of the currently selected block
currentBlockHandle = gcbh;

% Perform actions using the block handle
if ~isempty(currentBlockHandle)
    disp(['The handle of the current block is: ', num2str(currentBlockHandle)]);
    % Perform actions or gather information related to the current block
else
    disp('No block is currently selected.');
end
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

gcb

- Keep in mind that, similar to `gcb`, `gcbh` also returns an empty array (`[]`) if no block is currently selected. Therefore, it's good practice to check whether a block is selected before attempting to use its handle.
- The block handle is a unique identifier that Simulink uses to represent the selected block in the model. It allows you to programmatically modify block parameters, connect or disconnect blocks, or perform various other actions on the selected block.
- This function is particularly useful when you need to dynamically reference or manipulate the properties of the currently selected block in Simulink through MATLAB scripting.


Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

```matlab
% Get the handle of the currently selected block
currentBlockHandle = gcbh;

% Perform actions using the block handle
if ~isempty(currentBlockHandle)
    disp(['The handle of the current block is: ', num2str(currentBlockHandle)]);
    % Perform actions or gather information related to the current block
else
    disp('No block is currently selected.');
end
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

inspect

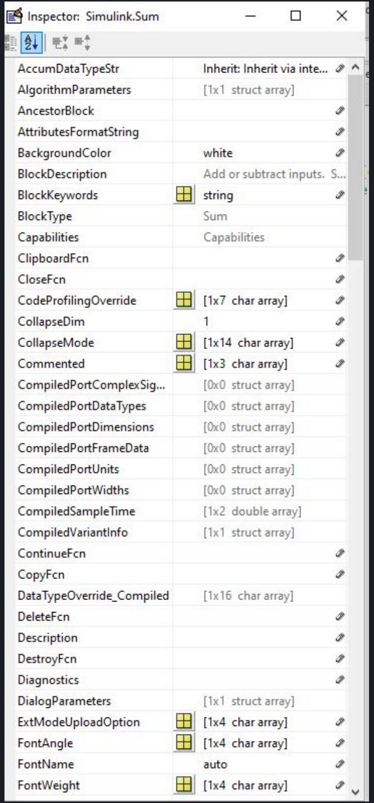
- In MATLAB's Simulink environment, the `inspect(gcbh)` function combination is used to inspect the properties of the currently executing block within a Simulink model.
- Purpose: The `inspect()` function opens the Simulink Property Inspector, allowing users to view and modify the properties of a Simulink block or model.
- Argument: `gcbh`
- `gcbh`: This function retrieves the handle of the currently executing block within a Simulink model.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

inspect

- To inspect the properties of a certain block, use the command inspect (BlockHandle)
- Important Parameters:
  - Handle, Block Type, Value, Position, Path
- inspect(gcbh)
- inspect(CnstBlockHandle)



Inspector: Simulink.Sum

AccumDataTypeStr		Inherit: Inherit via inte...
AlgorithmParameters		[1x1 struct array]
AncestorBlock		
AttributesFormatString		
BackgroundColor		white
BlockDescription		Add or subtract inputs. S...
BlockKeywords	⊞	string
BlockType		Sum
Capabilities		Capabilities
ClipboardFcn		
CloseFcn		
CodeProfilingOverride	⊞	[1x7 char array]
CollapseDim		1
CollapseMode	⊞	[1x14 char array]
Commented	⊞	[1x3 char array]
CompiledPortComplexSig...		[0x0 struct array]
CompiledPortDataTypes		[0x0 struct array]
CompiledPortDimensions		[0x0 struct array]
CompiledPortFrameData		[0x0 struct array]
CompiledPortUnits		[0x0 struct array]
CompiledPortWidths		[0x0 struct array]
CompiledSampleTime		[1x2 double array]
CompiledVariantInfo		[1x1 struct array]
ContinueFcn		
CopyFcn		
DataTypeOverride_Compiled		[1x16 char array]
DeleteFcn		
Description		
DestroyFcn		
Diagnostics		
DialogParameters		[1x1 struct array]
ExtModeUploadOption	⊞	[1x4 char array]
FontAngle	⊞	[1x4 char array]
FontName		auto
FontWeight	⊞	[1x4 char array]

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

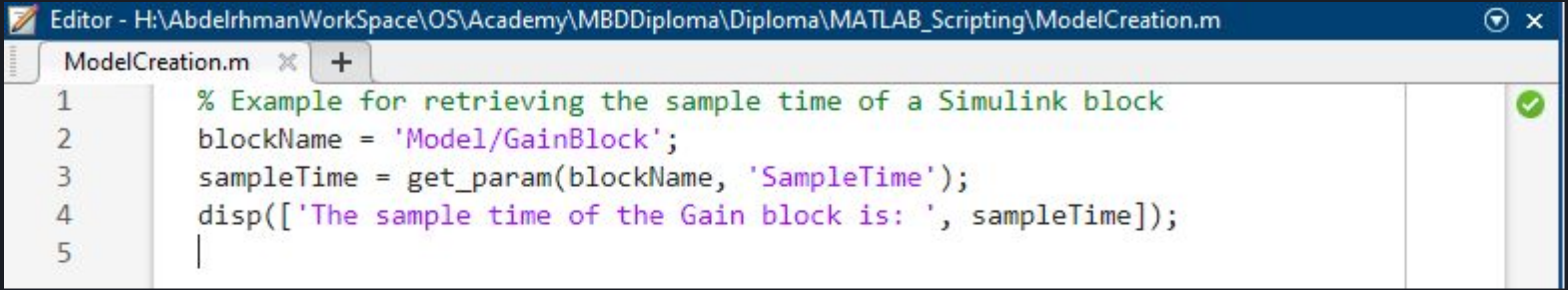
- In MATLAB's Simulink environment, the `get_param` function is used to retrieve the value of a parameter for a specified block within a Simulink model. This function allows you to programmatically access various properties and parameters of Simulink blocks.
- Purpose: Retrieve the value of a parameter for a specified Simulink block.
- `block`: The name or handle of the Simulink block whose parameter you want to access.
- `parameterName`: The name of the parameter you want to retrieve.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

- In this example, `get_param` is used to retrieve the sample time parameter of a Simulink block named 'GainBlock' within the model 'Model'.



Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

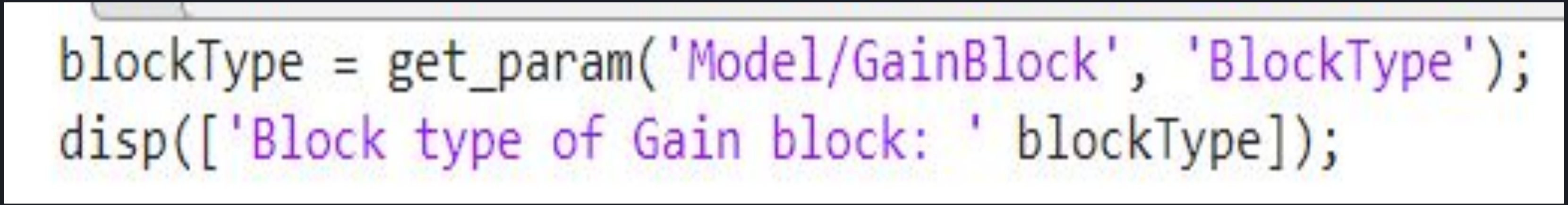
```
% Example for retrieving the sample time of a Simulink block
blockName = 'Model/GainBlock';
sampleTime = get_param(blockName, 'SampleTime');
disp(['The sample time of the Gain block is: ', sampleTime]);
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

- In this example, `get_param` is used to retrieve the Block Type parameter of a Simulink block named 'GainBlock' within the model 'Model'.



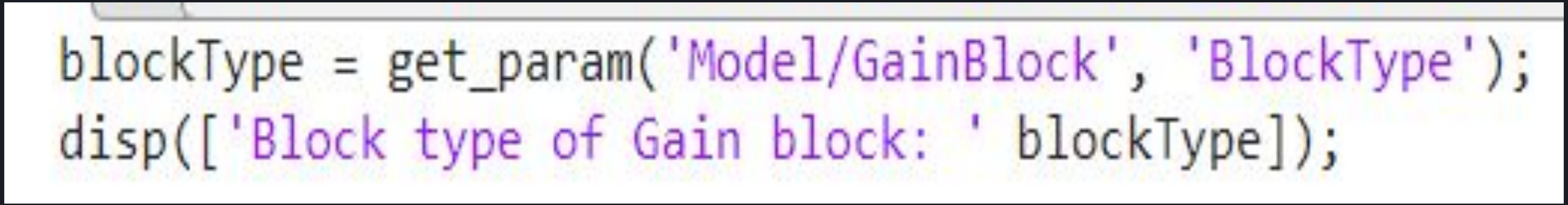
```
blockType = get_param('Model/GainBlock', 'BlockType');
disp(['Block type of Gain block: ' blockType]);
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

- In this example, `get_param` is used to retrieve the Block Type parameter of a Simulink block named 'GainBlock' within the model 'Model'.



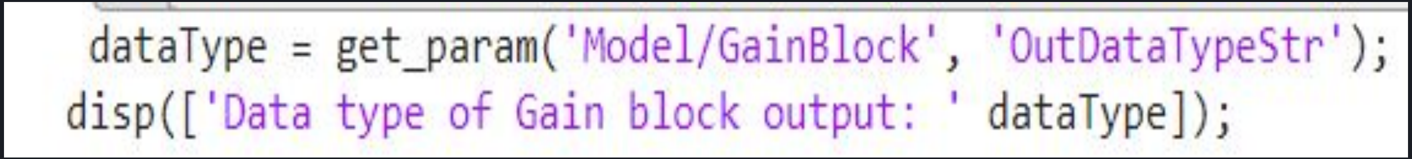
```
blockType = get_param('Model/GainBlock', 'BlockType');
disp(['Block type of Gain block: ' blockType]);
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

- In this example, `get_param` is used to retrieve the Data Type parameter of a Simulink block named 'GainBlock' within the model 'Model'.



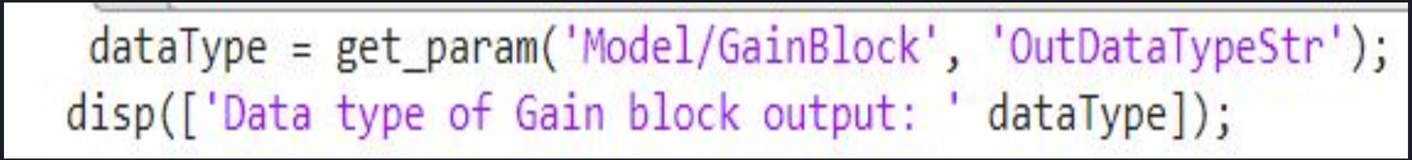
```
 dataType = get_param('Model/GainBlock', 'OutDataTypeStr');
disp(['Data type of Gain block output: ' dataType]);
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

- In this example, `get_param` is used to retrieve the Data Type parameter of a Simulink block named 'GainBlock' within the model 'Model'.



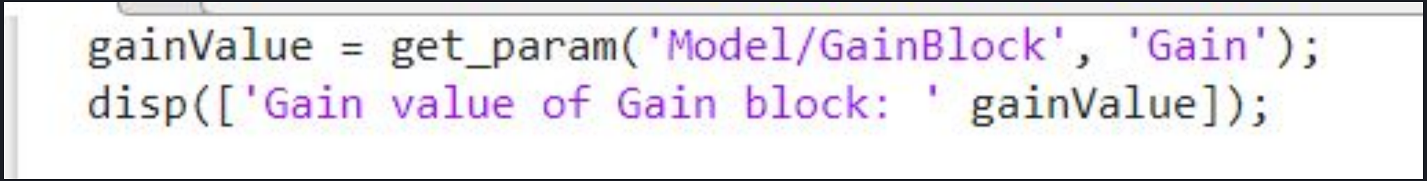
```
 dataType = get_param('Model/GainBlock', 'OutDataTypeStr');
disp(['Data type of Gain block output: ' dataType]);
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

- In this example, `get_param` is used to retrieve the value parameter of a Simulink block named 'GainBlock' within the model 'Model'.

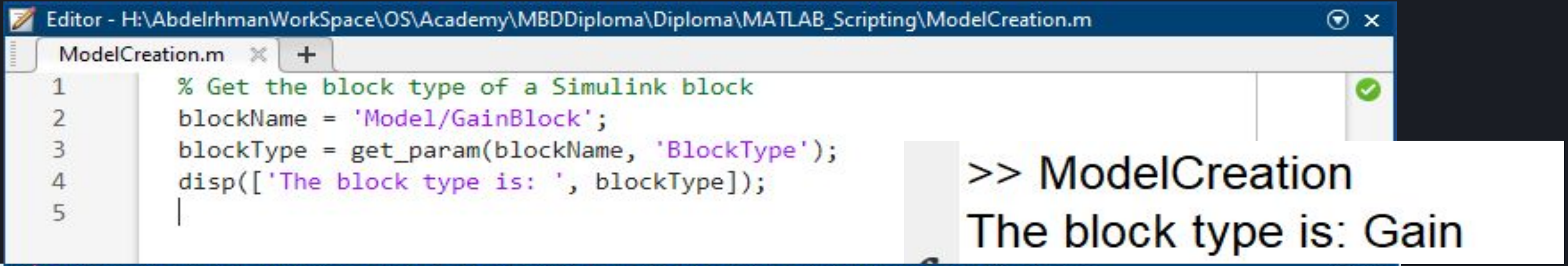


```
gainValue = get_param('Model/GainBlock', 'Gain');
disp(['Gain value of Gain block: ' gainValue]);
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

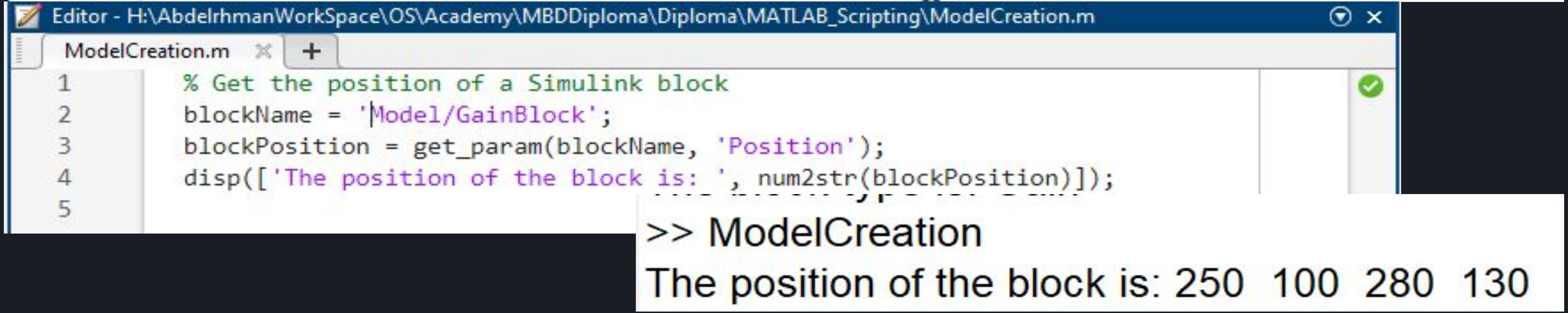
get_param(blockName, parameterName)


Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

```
% Get the block type of a Simulink block
blockName = 'Model/GainBlock';
blockType = get_param(blockName, 'BlockType');
disp(['The block type is: ', blockType]);
```

```
>> ModelCreation
The block type is: Gain
```


Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

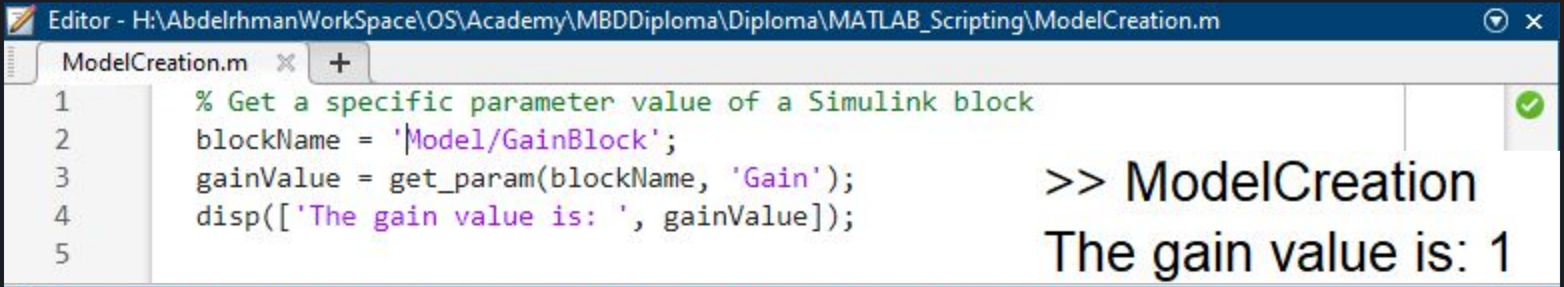
```
% Get the position of a Simulink block
blockName = 'Model/GainBlock';
blockPosition = get_param(blockName, 'Position');
disp(['The position of the block is: ', num2str(blockPosition)]);
```

```
>> ModelCreation
The position of the block is: 250  100  280  130
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

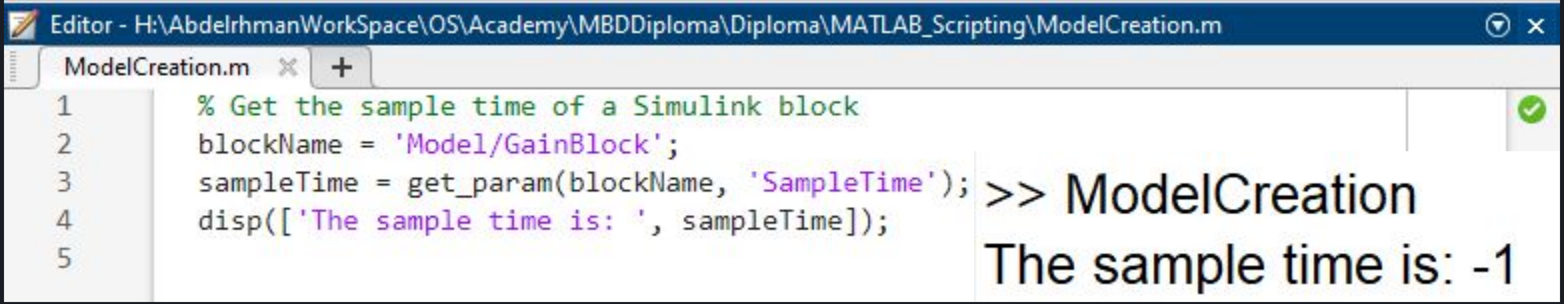


Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

```matlab
% Get a specific parameter value of a Simulink block
blockName = 'Model/GainBlock';
gainValue = get_param(blockName, 'Gain');
disp(['The gain value is: ', gainValue]);
```

```
>> ModelCreation
The gain value is: 1
```



Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

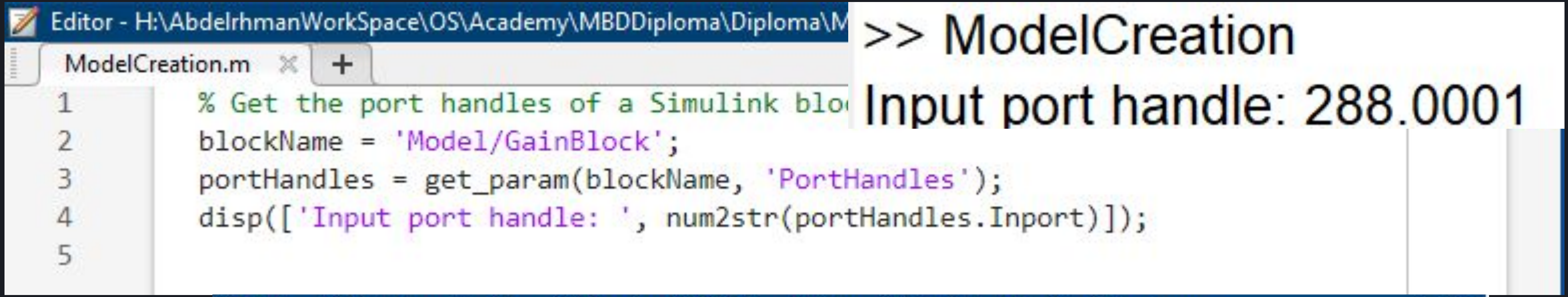
```matlab
% Get the sample time of a Simulink block
blockName = 'Model/GainBlock';
sampleTime = get_param(blockName, 'SampleTime');
disp(['The sample time is: ', sampleTime]);
```

```
>> ModelCreation
The sample time is: -1
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

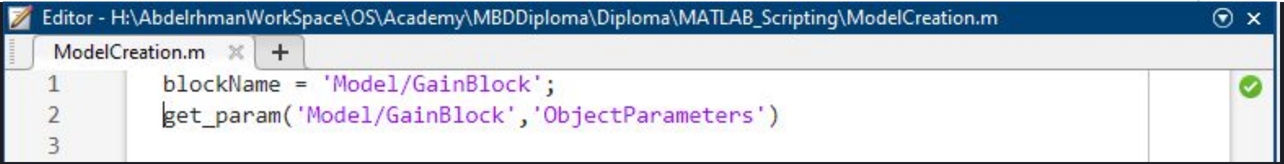


Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\M

ModelCreation.m

```
% Get the port handles of a Simulink bloc
blockName = 'Model/GainBlock';
portHandles = get_param(blockName, 'PortHandles');
disp(['Input port handle: ', num2str(portHandles.Inport)]);
```

```
>> ModelCreation
Input port handle: 288.0001
```



Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

```
blockName = 'Model/GainBlock';
get_param('Model/GainBlock','ObjectParameters')
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

get_param(blockName, parameterName)

Common Parameters:

Here are some common parameters that you might retrieve using `get_param`:

- `Name`: Name of the block.
- `BlockType`: Type of the block (e.g., 'Gain', 'Scope', etc.).
- `Value`: Value of a specific block parameter (e.g., gain value for a Gain block).
- `SampleTime`: Sample time of the block.
- `PortHandles`: Handles to the block's input and output ports.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

set_param(blockName, parameterName,paramterValue)

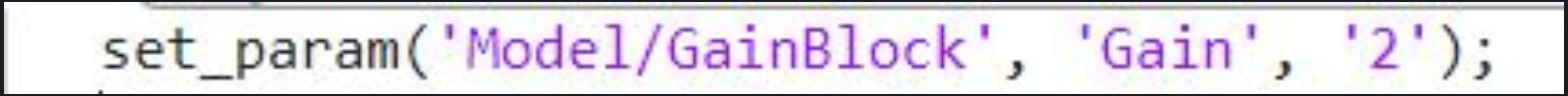
- In MATLAB's Simulink environment, the `set_param` function is used to set the value of a parameter for a specified block within a Simulink model. This function allows you to programmatically modify various properties and parameters of Simulink blocks.
- Purpose: Set the value of a parameter for a specified Simulink block.
- Arguments:
  - `block`: The name or handle of the Simulink block whose parameter you want to modify.
  - `parameterName`: The name of the parameter you want to set.
  - `parameterValue`: The value you want to assign to the parameter.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

set_param(blockName, parameterName,paramterValue)

- This code sets the gain value parameter of the 'Gain' block in the Simulink model named 'Model' to '2'.



```
set_param('Model/GainBlock', 'Gain', '2');
```

FOS-Academy

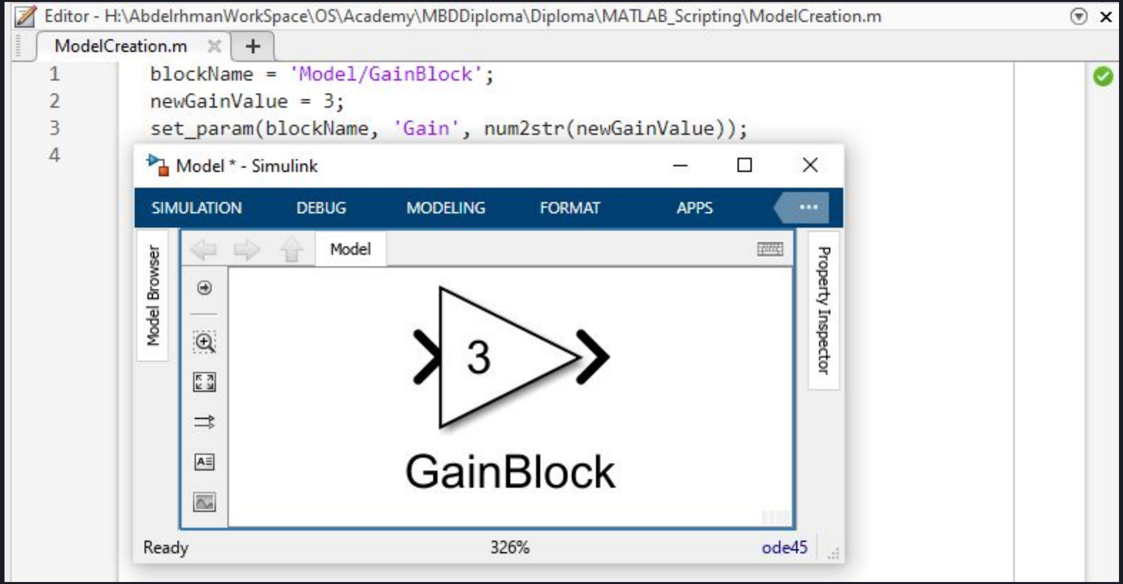
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

set_param(blockName, parameterName,paramterValue)

- This code sets the gain value parameter of the 'Gain' block in the Simulink model named 'Model' to '3'.



Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

```
blockName = 'Model/GainBlock';
newGainValue = 3;
set_param(blockName, 'Gain', num2str(newGainValue));
```

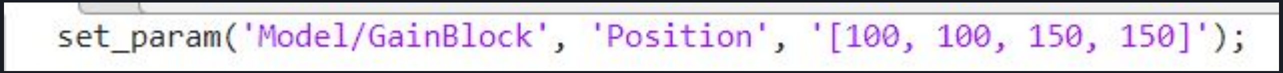
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

set_param(blockName, parameterName,paramterValue)

- This code sets the gain value parameter of the 'Gain' block in the Simulink model named 'Model' to '2'.



```
set_param('Model/GainBlock', 'Position', '[100, 100, 150, 150]');
```

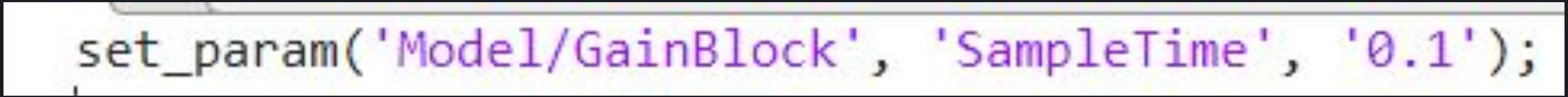
Model-Based Development Program

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

set_param(blockName, parameterName,paramterValue)

- This code sets the gain value parameter of the 'Gain' block in the Simulink model named 'Model' to '2'.



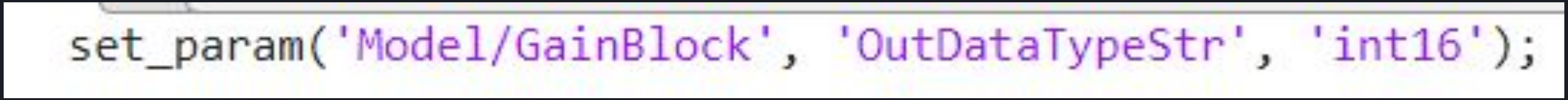
```
set_param('Model/GainBlock', 'SampleTime', '0.1');
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

set_param(blockName, parameterName,paramterValue)

- This code sets the gain value parameter of the 'Gain' block in the Simulink model named 'Model' to '2'.

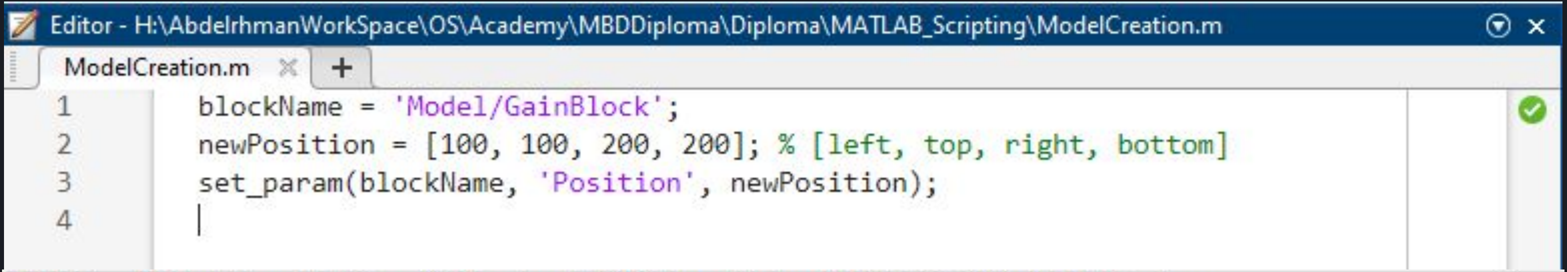


```
set_param('Model/GainBlock', 'OutDataTypeStr', 'int16');
```

Matlab Scripting Module

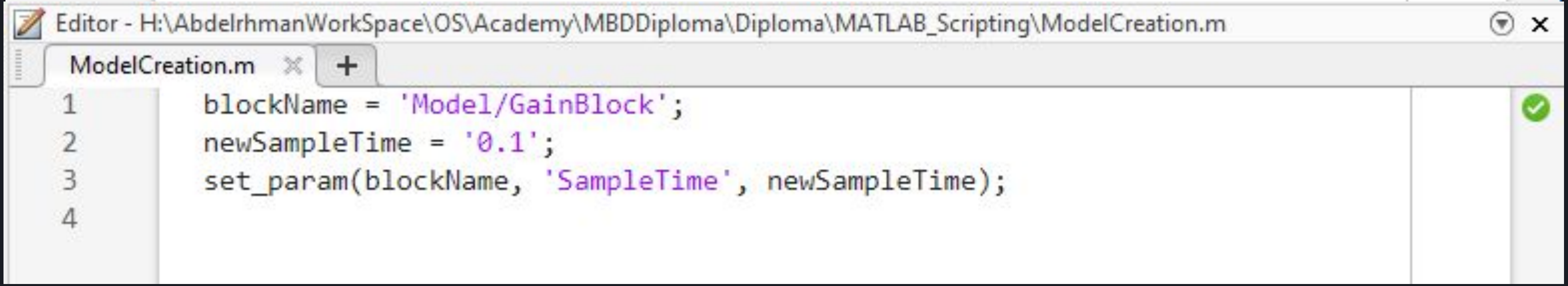
Automate repetitive Modeling and Simulation Tasks

set_param(blockName, parameterName,paramterValue)


Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

```
blockName = 'Model/GainBlock';
newPosition = [100, 100, 200, 200]; % [left, top, right, bottom]
set_param(blockName, 'Position', newPosition);
```


Editor - H:\AbdelrhmanWorkSpace\OS\Academy\MBDDiploma\Diploma\MATLAB_Scripting\ModelCreation.m

ModelCreation.m

```
blockName = 'Model/GainBlock';
newSampleTime = '0.1';
set_param(blockName, 'SampleTime', newSampleTime);
```

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

set_param(blockName, parameterName,paramterValue)

- For parameters that expect numeric values, you may need to convert the values to strings using `num2str`.
- Some parameters may have constraints or valid value ranges, so make sure the assigned values are within the valid range.
- Changes made using `set_param` take effect during the simulation or when the model is updated.
- `set_param` is a powerful tool for automating and customizing Simulink models through MATLAB scripts, making it

  easier to manage and manipulate models programmatically.

Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise

Create a MATLAB script that covers the usage of various Simulink functions to create, modify, simulate, and save a Simulink model using Gain, Step, and Scope blocks.

Here's how you can structure the exercise:

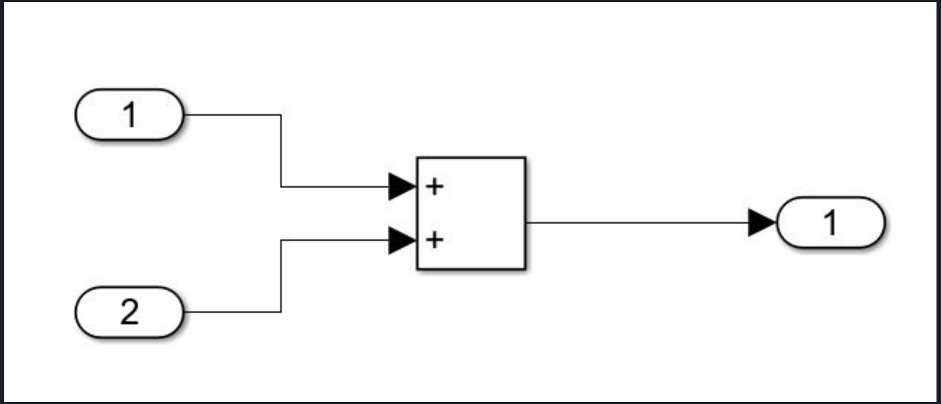
- Create a new Simulink model named "SimulationExample".
- Add a Gain block, a Step Input block, and a Scope block to the model.
- Connect the Step Input block to the Gain block, and then connect the Gain block to the Scope block.
- Set parameters for the blocks using `set_param`, such as the gain value for the Gain block and the step time and amplitude for the Step Input block.
- Retrieve parameters for the blocks using `get_param`, such as the sample time for the Scope block, and display them.
- Simulate the model using the `sim()` function.
- Open the model using `open_system`.
- Save the model using `save_system`.
- Close the model using `close_system`.

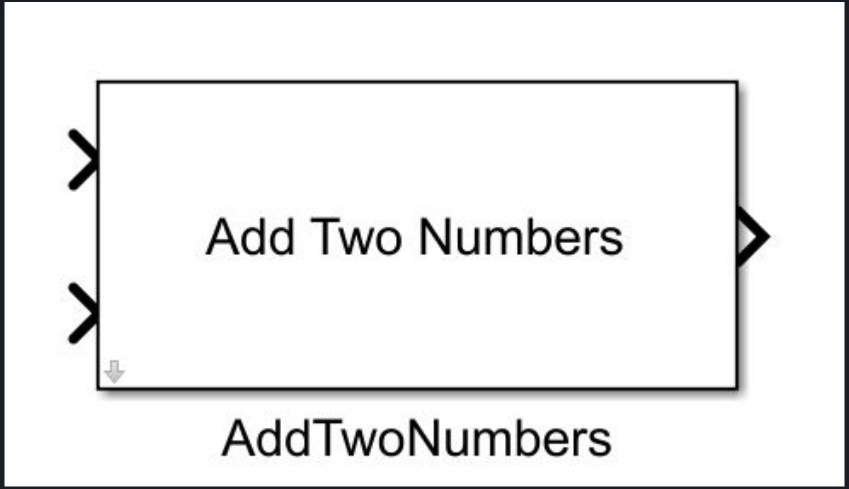
Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise

It is required to programmatically create a model that contains “Add Two Numbers” block



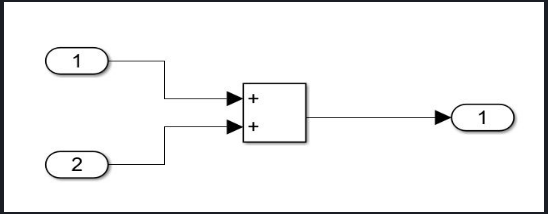


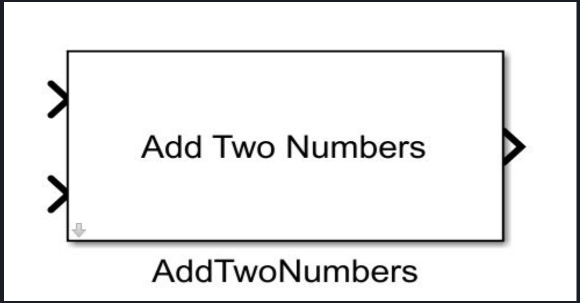
Matlab Scripting Module

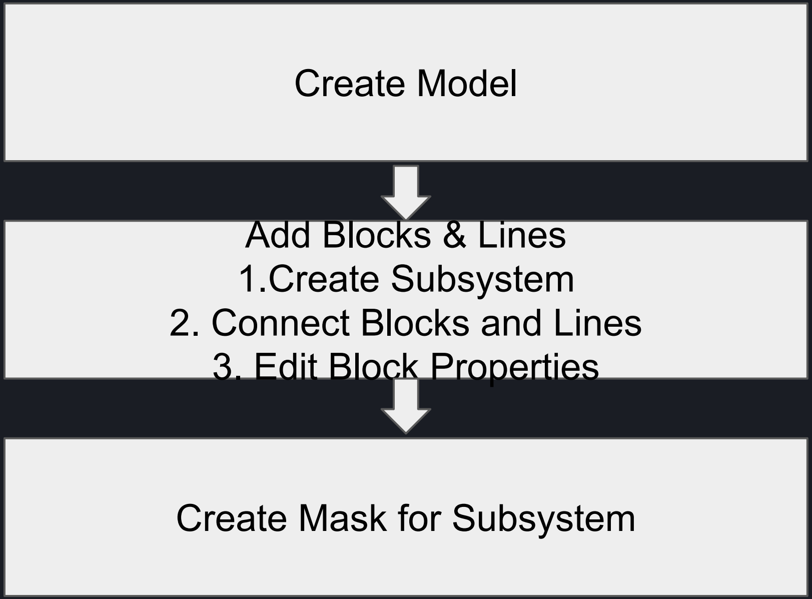
Automate repetitive Modeling and Simulation Tasks

Exercise

It is required to programmatically create a model that contains “Add Two Numbers” block





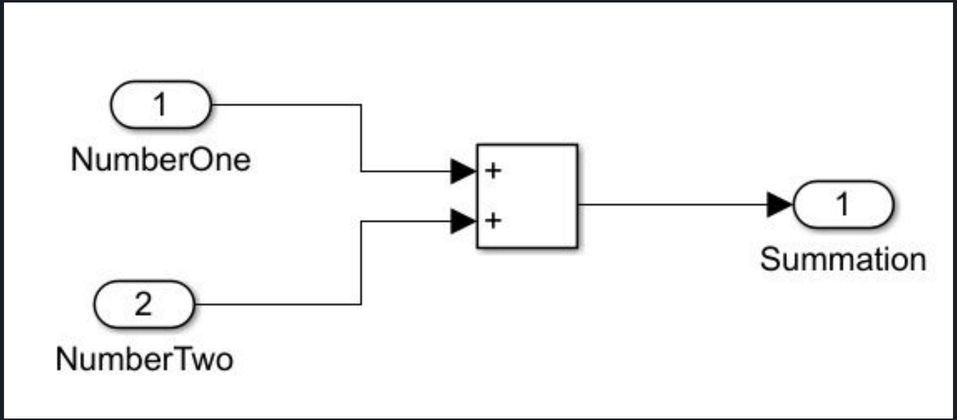


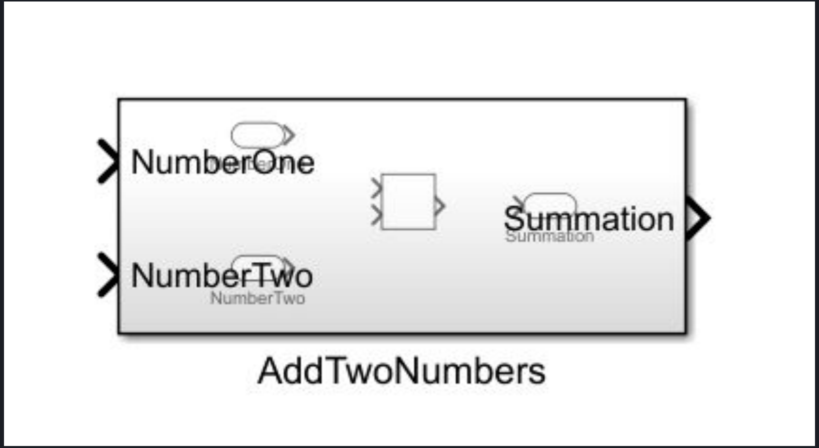
Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise

It is required to programmatically create a model that contains “Add Two Numbers” block



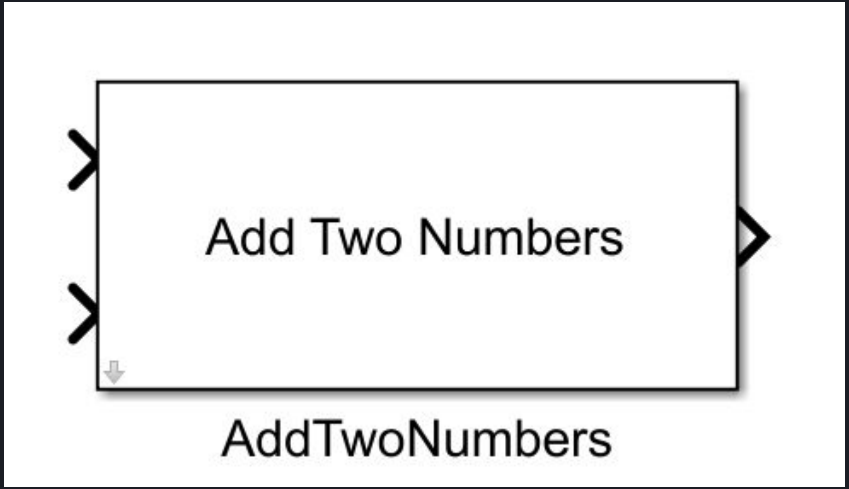


Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise

It is required to programmatically create a model that contains “Add Two Numbers” block

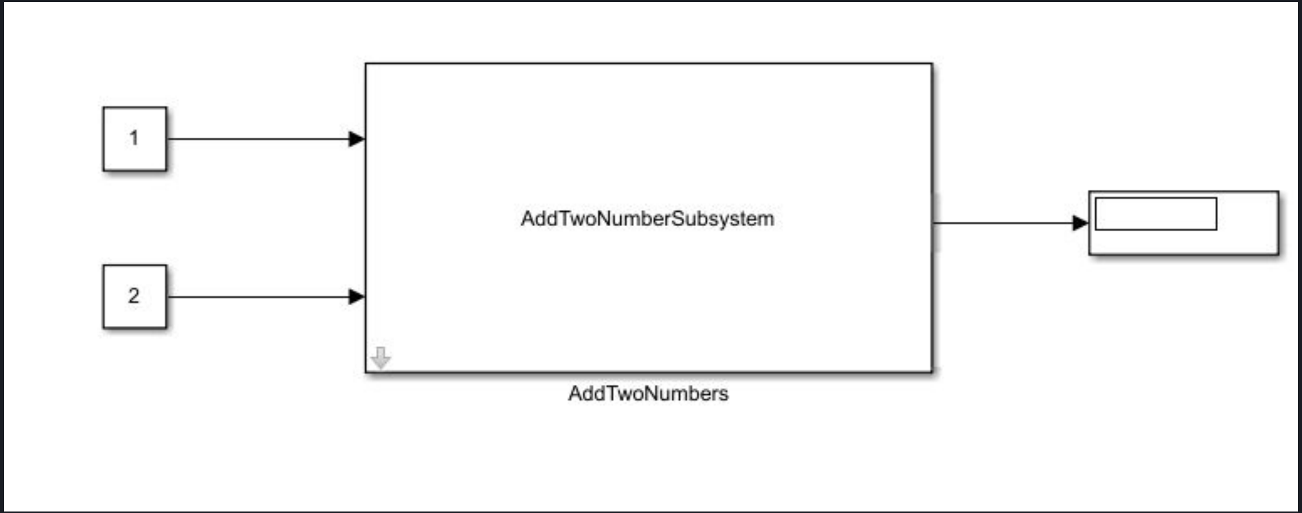


Matlab Scripting Module

Automate repetitive Modeling and Simulation Tasks

Exercise

Now try to test your work programmatically



Thank You!