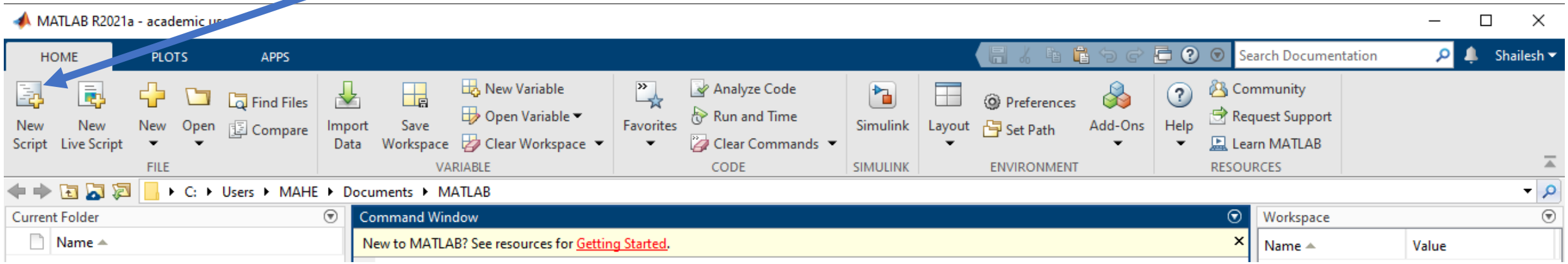
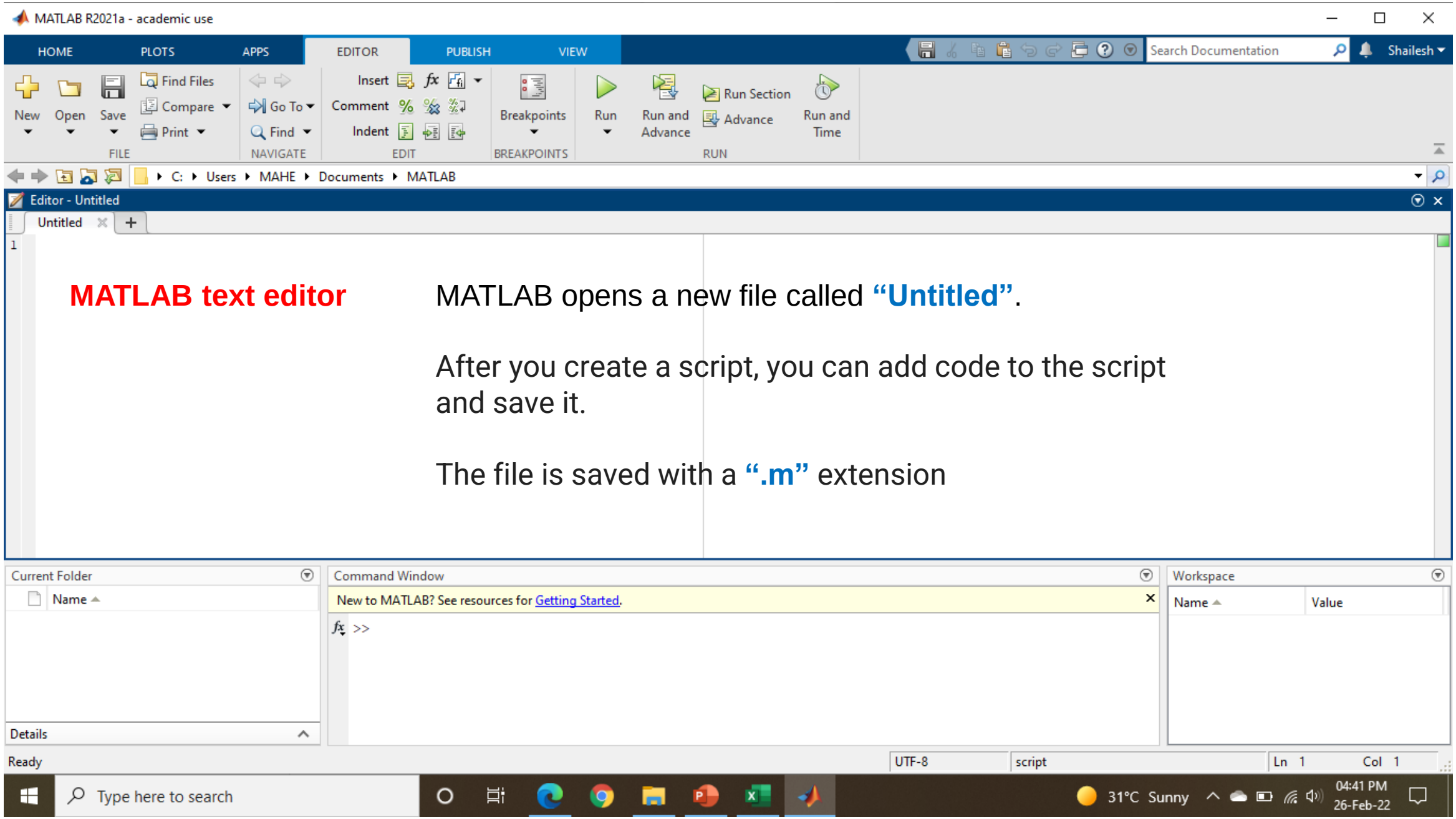
Scripts and Functions

Programming in MATLAB

Scripts (.m files)

- **Scripts** are program (code) files.
- Scripts contain a series of **sequential MATLAB statements and function calls**.
- In command window type >> **edit**
- Or click on "**New Script**" on MATLAB toolbar





MATLAB text editor

MATLAB opens a new file called **"Untitled"**.

After you create a script, you can add code to the script and save it.

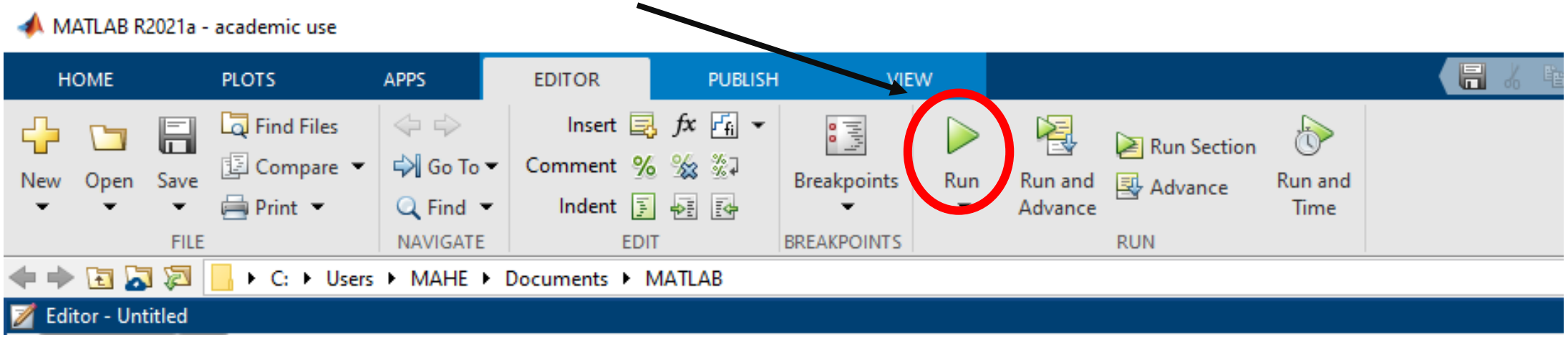
The file is saved with a **".m"** extension

Running the script

- Save your script and run the code using either of these methods:

- Type the script name on the **command line** and press **Enter**.
- For example,
  - to run the **numGenerator.m** script, type >> **numGenerator**

- On the **Editor** tab, click the **Run** button.



Comments in MATLAB

- Comments allow **others** to **understand** code and one can **refresh** their memory when they **return to it later**.

- During code **development** and testing, one can use **comments** to **comment** out **any code** that **does not need to run**.

- To add comments to MATLAB code, use the percent (**%**) symbol.
  - Comment lines can appear anywhere in a code file, and one can append comments to the end of a line of code.

Comments in MATLAB

- To comment out multiple lines of code, use the block comment operators, %{ and %}.
- The %{ and %} operators must appear alone on the lines that immediately precede and follow the block of help text.
- Do not include any other text on these lines.

Example code :

```
a = magic(3);

%{
sum(a)
diag(a)
sum(diag(a))
%}

sum(diag(fliplr(a)))
```

Effective Use of Script Files

- Follow the MA TLAB convention for **naming variables**.
- Avoid giving **script files** the **same name** as a **variable** it computes.
- Do not give a **script file** the **same name** as a MATLAB **command** or **function**.
- Use **exist** command – try >> help **exist**

Debugging Script Files

- **Debugging** a program is the process of **finding** and removing **errors.**
- **Syntax errors** - MATLAB usually detects and displays a message describing the error and its location.
- **Always test the code** with a simple version of the problem, whose **answers** can be checked by **hand calculations**.
- **Display** any **intermediate calculations** by **removing semicolons** at the end of statements.

Programming Style

- **Comments section** – About the input, output variables and user defined functions or any other relevant information.
- **Input section** - About input data and/or the input functions that enable data to be entered.
- **Calculation section –** Functions and algorithms required for calculations in this section.
- **Output section –** Functions necessary to deliver the output in whatever form required.

Input/output commands

- **disp**(A) Displays the contents, but not the name, of the array A.

- **disp**(‘text’) Displays the text string enclosed within single quotes.

- x = **input**(‘text’) Displays the text in quotes, waits for user input from the keyboard, and stores the value in x.

- x = **input**(‘text’,’s’) Displays the text in quotes, waits for user input from the keyboard, and stores the input as a string in x.

Example : Type

```
>> disp(‘The predicted speed is:’)
>> disp(Speed)
```

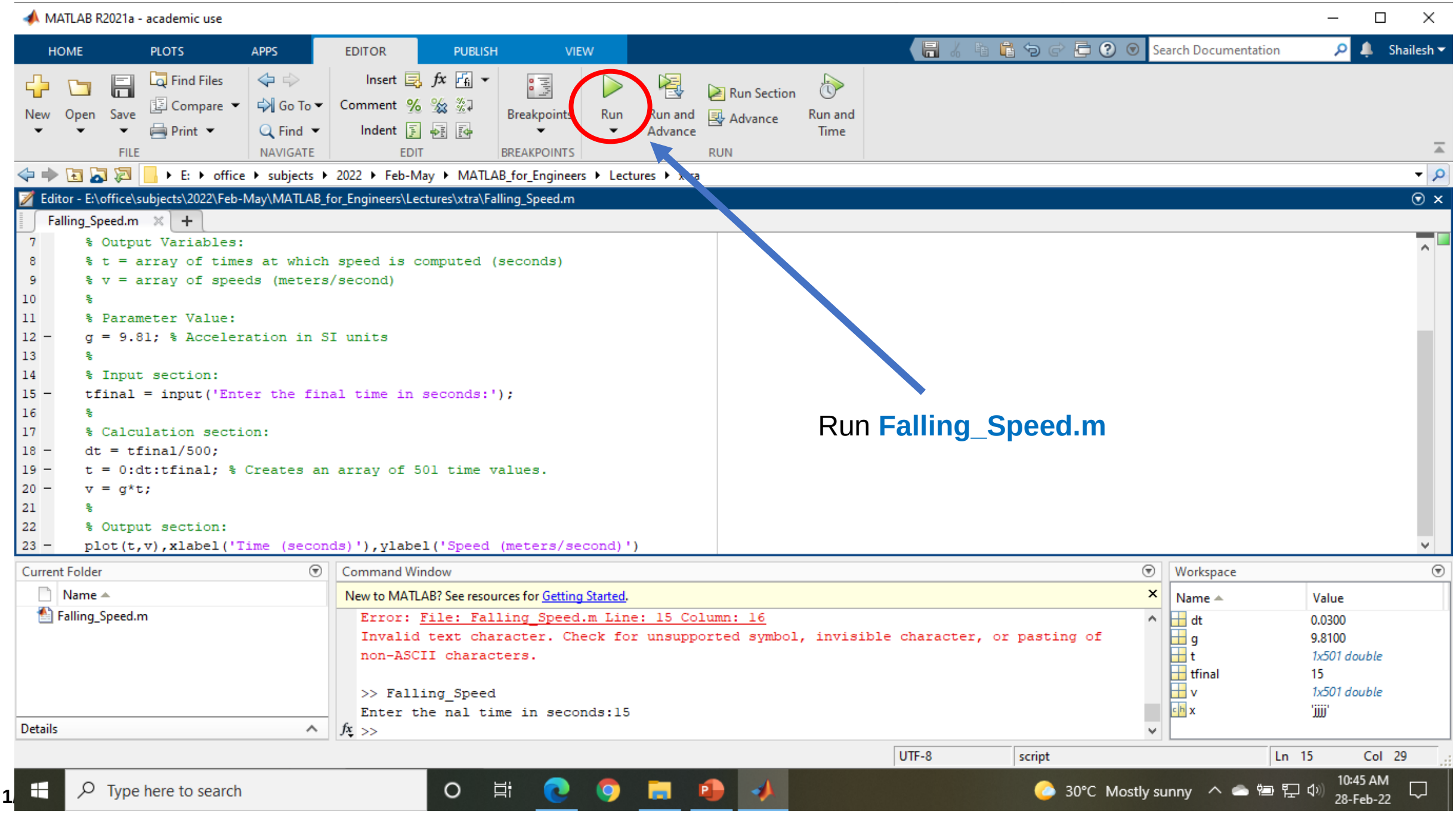
Example of a Script File

- % **Program Falling_Speed.m: plots speed of a falling object.**
- % **Created on** March 1, 2009 by W. Palm III
- %
- % **Input Variable:**
- % tfinal = final time (in seconds)
- %
- % **Output Variables:**
- % t = array of times at which speed is computed (seconds)
- % v = array of speeds (meters/second)
- %
- % **Parameter Value:**
- g = 9.81; % Acceleration in SI units

% **Input section:**
tfinal = input('Enter the final time in seconds:');
%
% **Calculation section:**
dt = tfinal/500;
t = 0:dt:tfinal; % Creates an array of 501 time values.
v = g*t;
%
% **Output section:**
plot(t,v),xlabel('Time (seconds)'),ylabel('Speed (meters/second)')

Save and Run **Falling_Speed.m**

- After creating this file, you save it with the name **Falling_Speed.m**.
- To run it, you type **Falling_Speed** (without the .m) in the Command window at the prompt.
- >> **Falling_Speed**
- You will then be asked to enter a value for **tfinal** .
- After you enter a value and press **Enter**, you will see the plot on the screen.

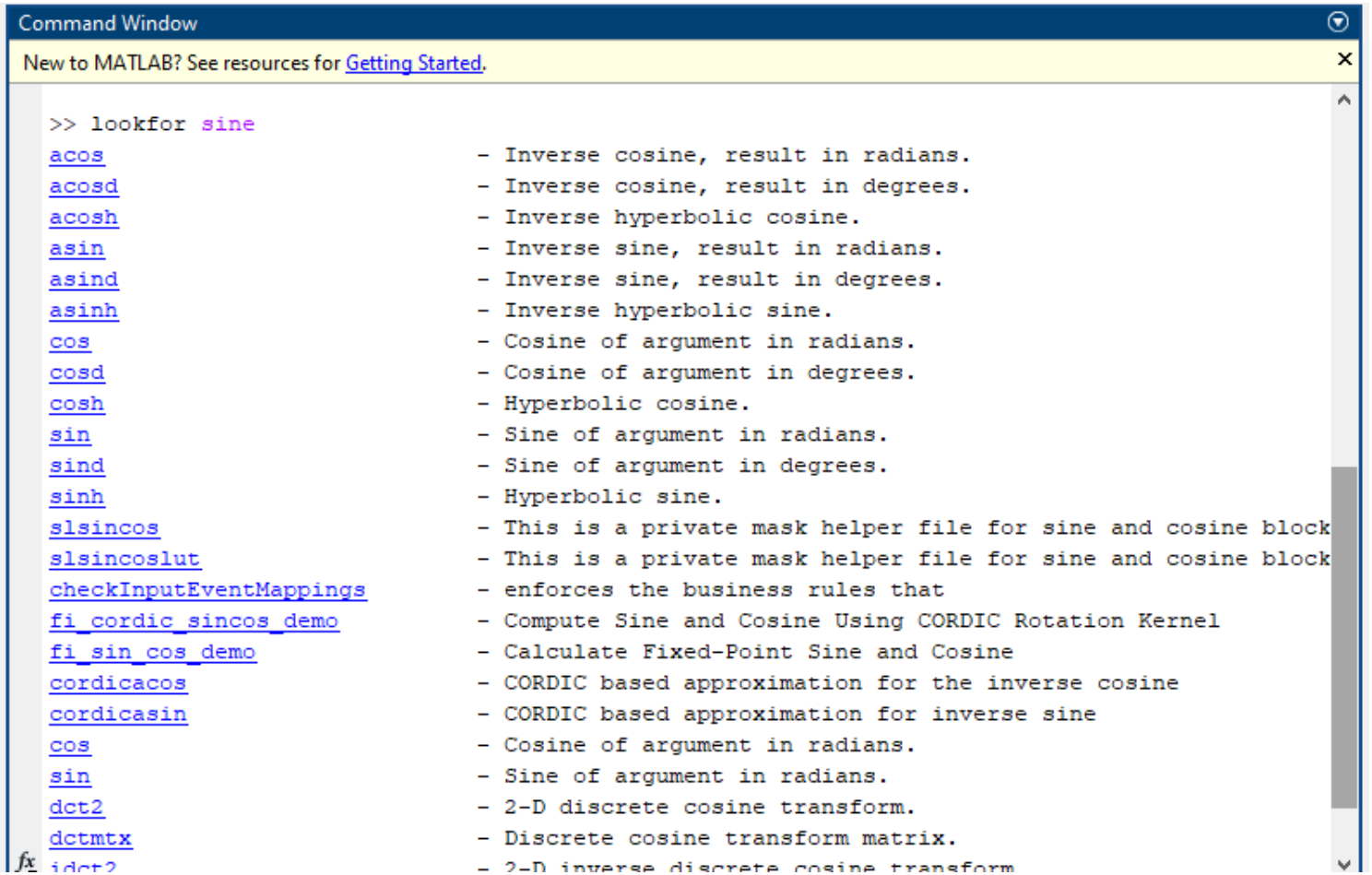

MATLAB R2021a - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Shailesh
New
Open
Save
Find Files
Compare
Print
FILE
Go To
Find
NAVIGATE
Insert
Comment
Indent
EDIT
Breakpoints
BREAKPOINTS
Run
Run and Advance
Run Section
Advance
Run and Time
RUN
E: ▸ office ▸ subjects ▸ 2022 ▸ Feb-May ▸ MATLAB_for_Engineers ▸ Lectures ▸ xtra
Editor - E:\office\subjects\2022\Feb-May\MATLAB_for_Engineers\Lectures\xtra\Falling_Speed.m
Falling_Speed.m
7 % Output Variables:
8 % t = array of times at which speed is computed (seconds)
9 % v = array of speeds (meters/second)
10 %
11 % Parameter Value:
12 g = 9.81; % Acceleration in SI units
13 %
14 % Input section:
15 tfinal = input('Enter the final time in seconds:');
16 %
17 % Calculation section:
18 dt = tfinal/500;
19 t = 0:dt:tfinal; % Creates an array of 501 time values.
20 v = g*t;
21 %
22 % Output section:
23 plot(t,v),xlabel('Time (seconds)'),ylabel('Speed (meters/second)')
Run Falling_Speed.m
Current Folder
Name
Falling_Speed.m
Details
Command Window
New to MATLAB? See resources for Getting Started.
Error: File: Falling_Speed.m Line: 15 Column: 16
Invalid text character. Check for unsupported symbol, invisible character, or pasting of non-ASCII characters.
>> Falling_Speed
Enter the nal time in seconds:15
>>
Workspace
Name
Value
dt 0.0300
g 9.8100
t 1x501 double
tfinal 15
v 1x501 double
x 'jjjjj'
UTF-8
script
Ln 15
Col 29
Type here to search
30°C Mostly sunny
10:45 AM
28-Feb-22

The lookfor Function

- **lookfor** - Search all MATLAB files for keyword.

- Type >> help sine.m
- >> sine.m not found

- Type >> lookfor sine



Problem-Solving Methodologies

- Steps in Engineering Problem Solving –
- 1. **Understand** the purpose of the **problem**.

- 2. **Collect** the known and relevant **information - INPUT**.
  - Realize that some of it might later be found **unnecessary**.

- 3. **Determine** what **information** you must **find - OUTPUT**.

- 4. **Simplify** the **problem** only enough to obtain the required information. State any **assumptions** you make.

- 5. Draw a **sketch** and **label** any necessary **variables**.

Problem-Solving Methodologies

- Steps in Engineering Problem Solving –

- 6. Determine which **fundamental principles (Example : Newton's laws)** are applicable.

- 7. Think generally about your **proposed solution** approach and consider **other approaches** before proceeding with the details.

- 8. Label each step in the solution process.

Problem-Solving Methodologies

- Steps in Engineering Problem Solving –

- 9. If you solve the problem with a program, **hand check the results using a simple version of the problem**. Print the results of intermediate steps.

- 10. Perform a reality check and precision check on the answer.
  - For example : Height cannot be negative or large.

Function files

- **Functions** − functions files are also program files with .m extension.

- Functions can accept inputs and return outputs.

- Internal variables are local to the function.

function **f** = fact(**n**)

End

Save as **fact.m**

Function call

>> X = fact(Y)

Syntax for Function Definition

- **Function name** - Valid function names follow the same rules as **variable names**.
  - They must start with a **letter**, and can **contain letters**, **digits**, or **underscores**.

- function y = myFunction(one,two,three) - Input arguments

- function [one, two, three] = myFunction(x) - Output arguments

- If there is no output, you can omit it.
  - function myFunction(x) or function [] = myFunction(x)

Live script editor

- Watch the demo.