THE JOY OF MATLAB

A Presentation by Emilie Saucier

Matlab - Part 1

- *Installing Matlab*
- Matlab Interface
- Data Types
- Scripts
- Importing
- Tables
- Exporting
- Plots
- Useful functions
- Searching Documentation
- *Questions / Topic Request for Part 2*

INSTALLING MATLAB

For a McGill Student

It's free!

McGill has a MATLAB licence, which means you can install it on your own computer for free!

https://www.mathworks.com/academia/tah-portal/mcgill-university-30521249.html

You can Install MATLAB by following this link. You will be asked to create an account.

Use your McGill email when prompted as it is associated with the free licence.

You can follow this video for detailed instructions: https://www.mathworks.com/videos/how-to-install-matlab-1525083586145.html

When Installing, you might be prompted to add more products and Add-On, then the default.

You can go through the list and select more if you want, but it will extend the installation time

There are always way to install the Add-On at a later time, as needed

MATLAB INTERFACE

Command Window, Workspace, Current Folder, …

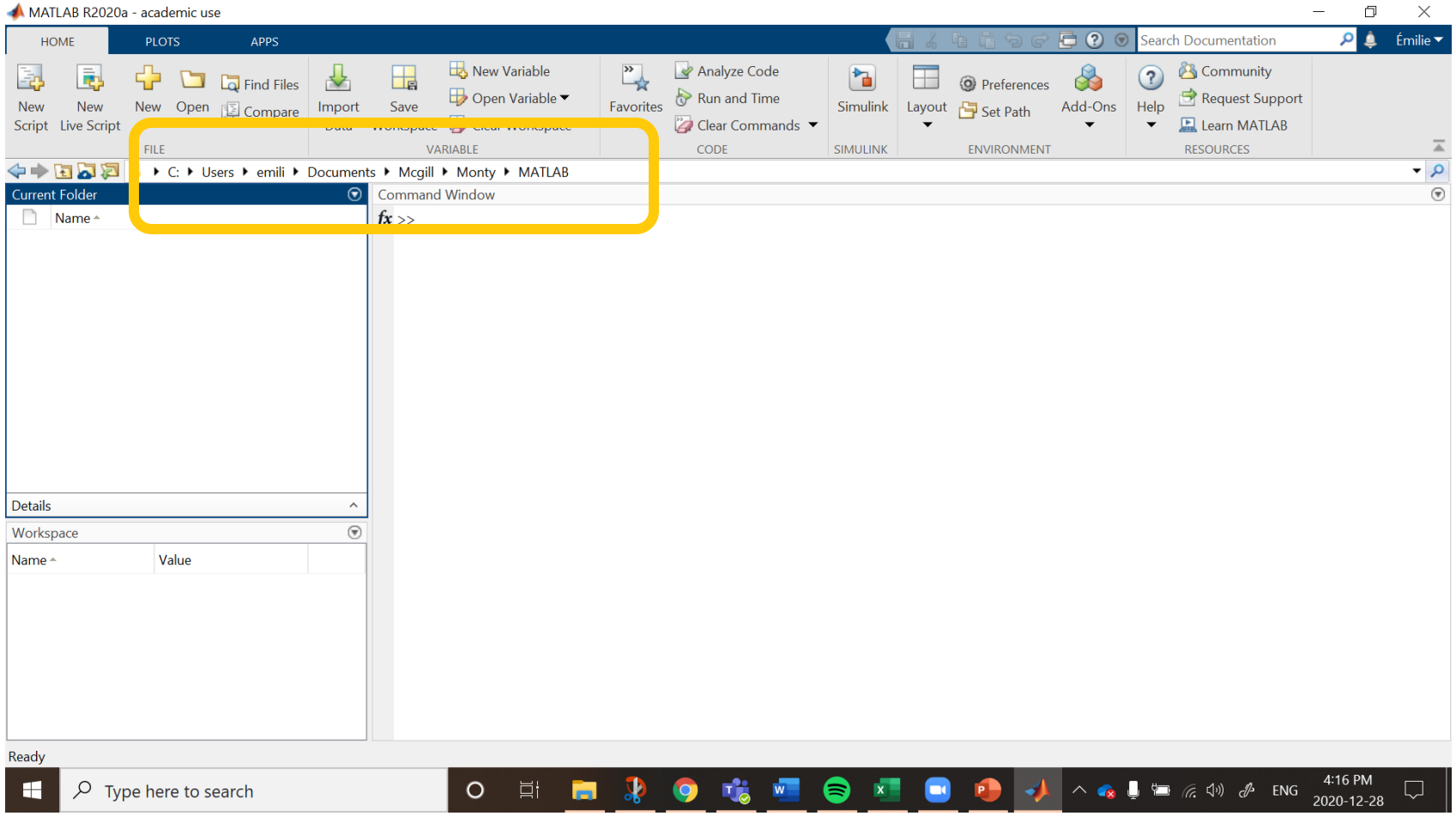
Current Folder

Every Project on MATLAB is contained within a folder.

This folder can be as large or specific as you want.

I recommend having a specific folder to contain a project and only that project.

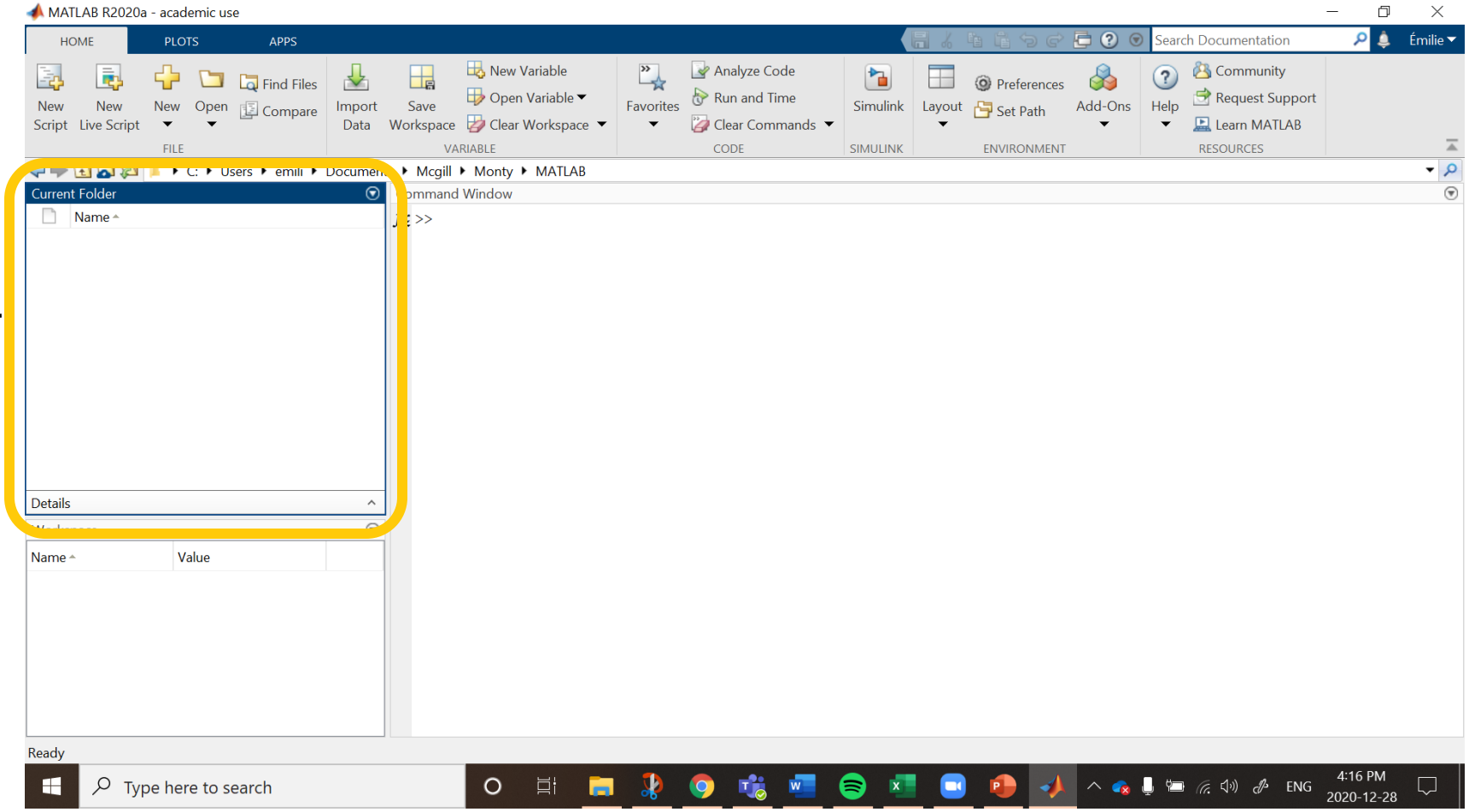
The Yellow Box outlines where that folder is located on your computer.



Current Folder

I now have the MATLAB folder open, and the Current Folder window is where you can see what is currently contained in your folder.

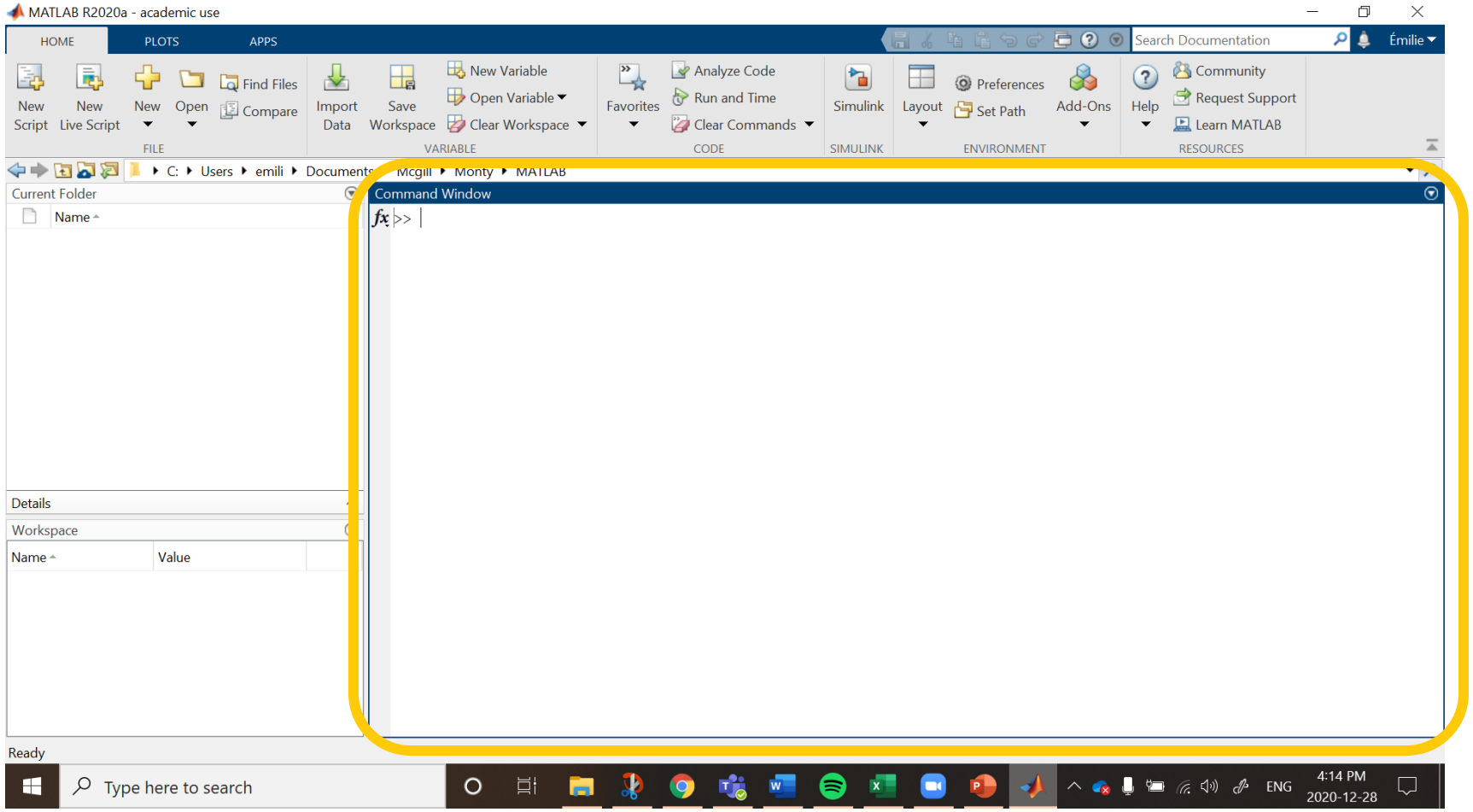
My MATLAB folder is currently empty, so nothing is displayed



Command Window

The Command Window is where you can directly type and submit instructions/commands one line at a time.

This is also where answers will be outputted.

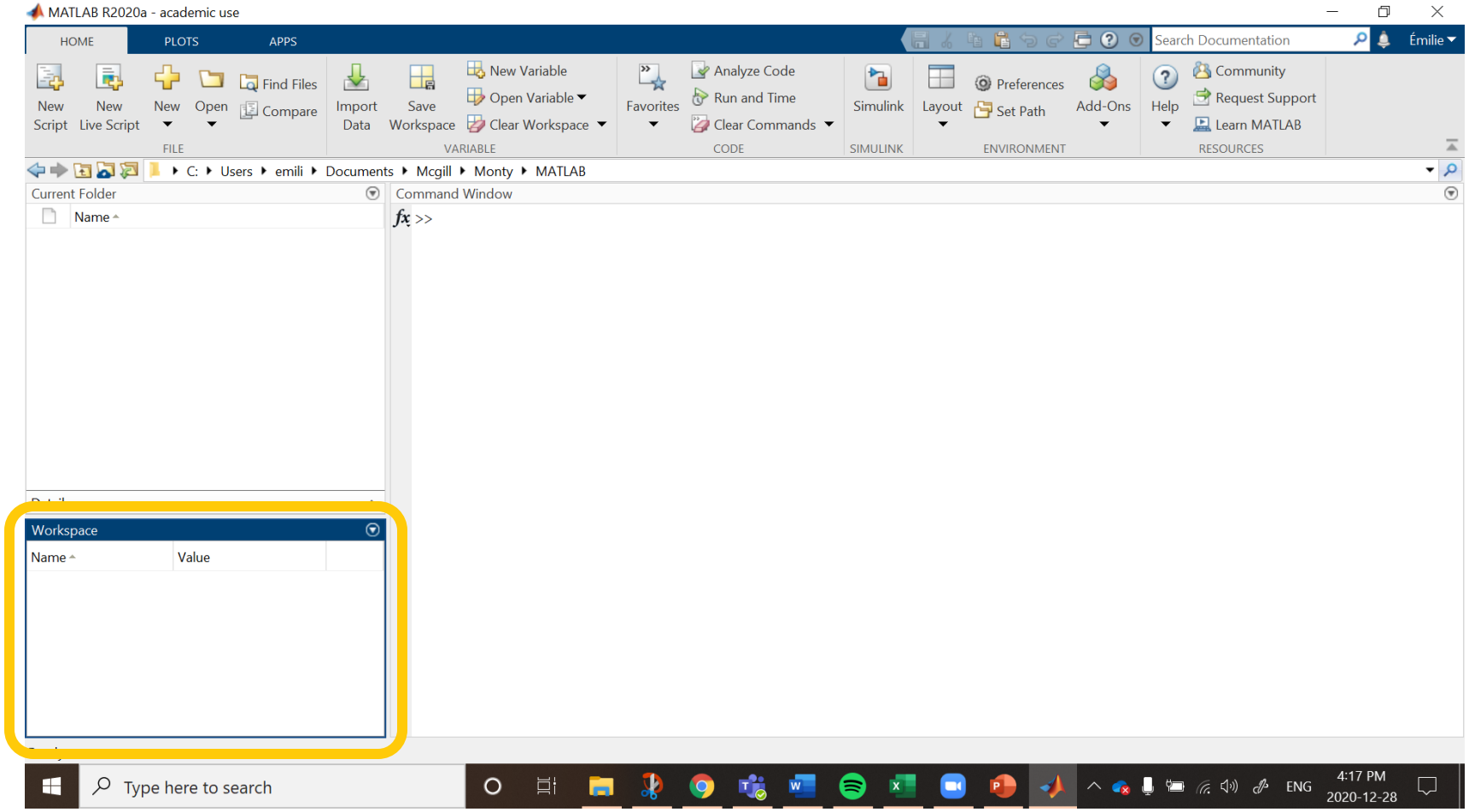


Workspace

The Workspace is where your variables will be displayed, with their name and values.

If the values would take to much place to display, the type of the value will be displayed instead.

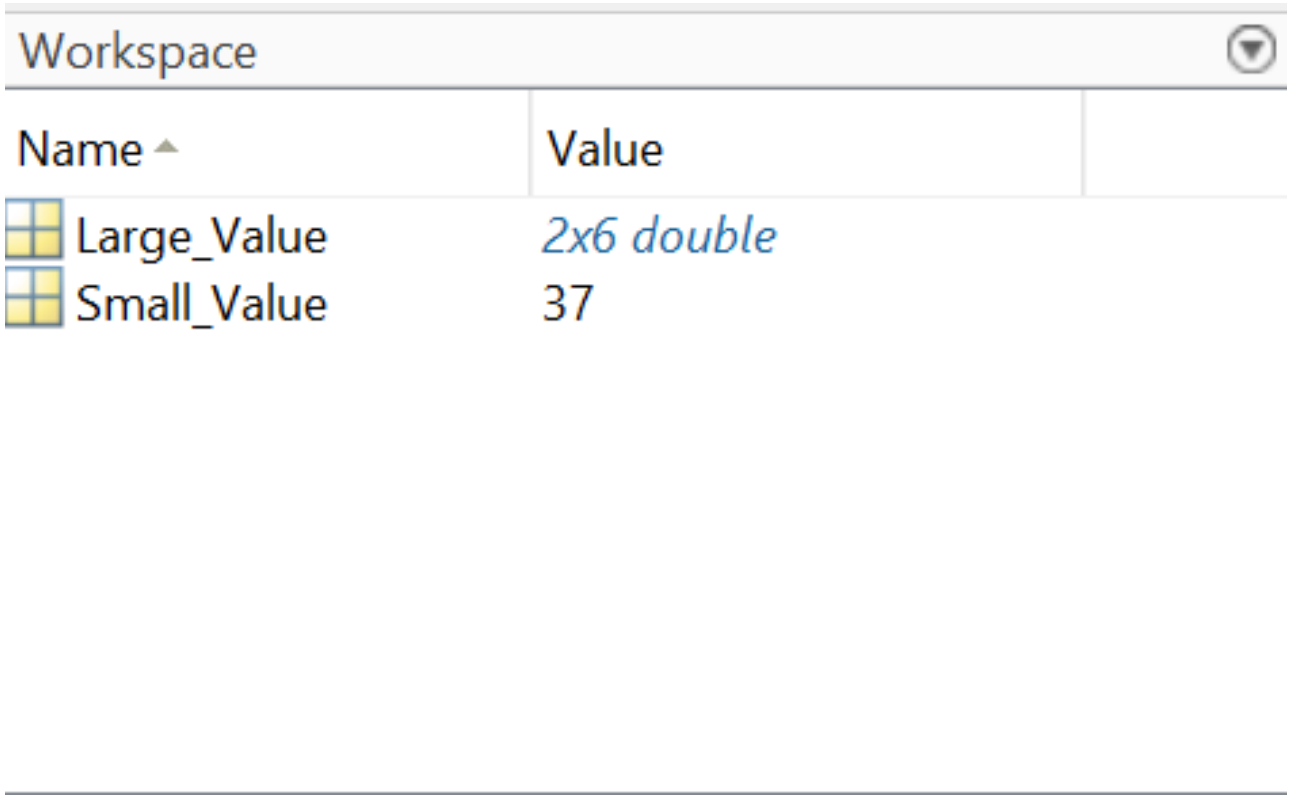
In that case, to visualise the value you would have to double click on the name and a new window would open



Workspace Example

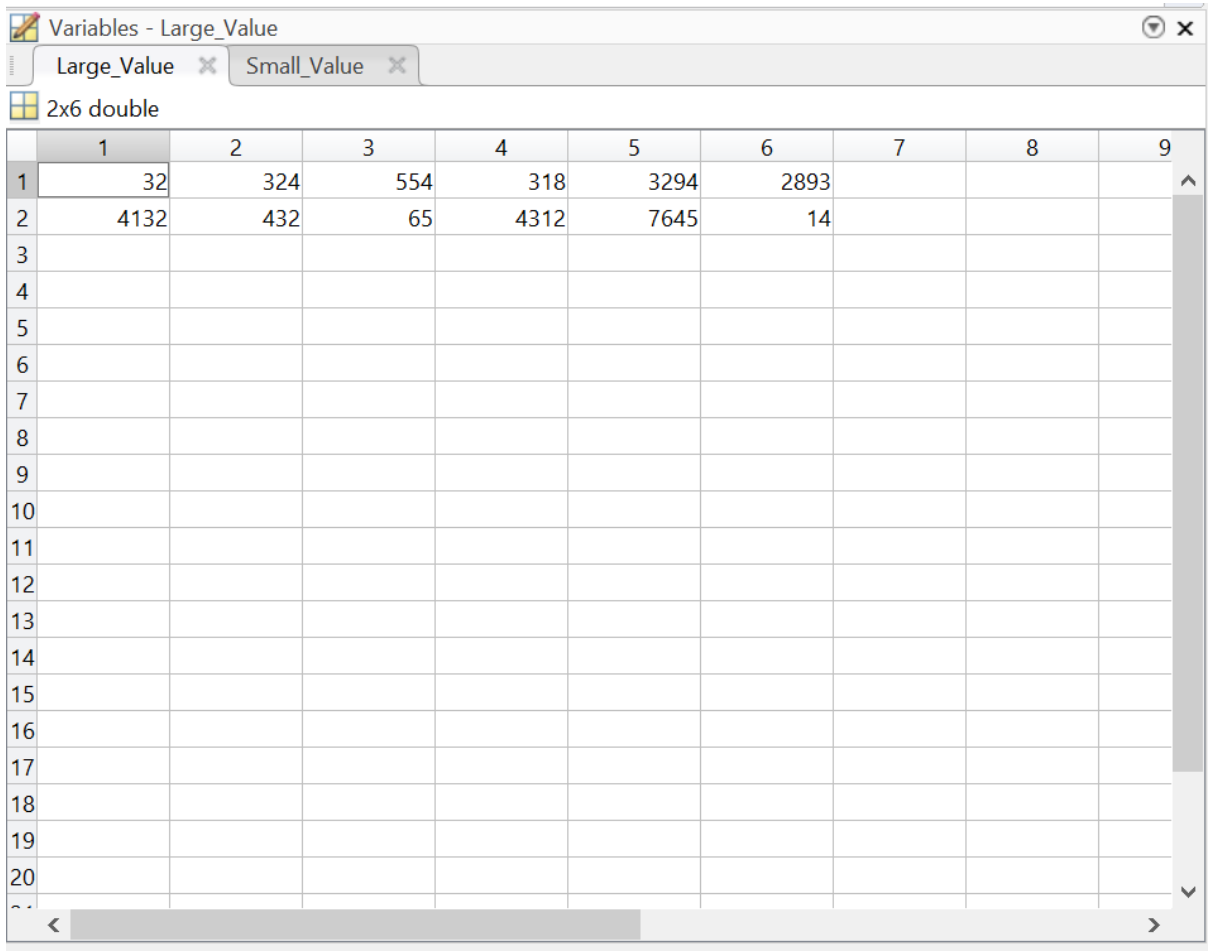
I have created two different values here, one Small_Value, that can be displayed, and one Large_Value of which the type (2x6 double) is displayed

By Double Clicking on the variable Name, the "Variables" window opens



Variables Window

In the Variable Window, you can investigate the content of various variables, such as this 2x6 double for example



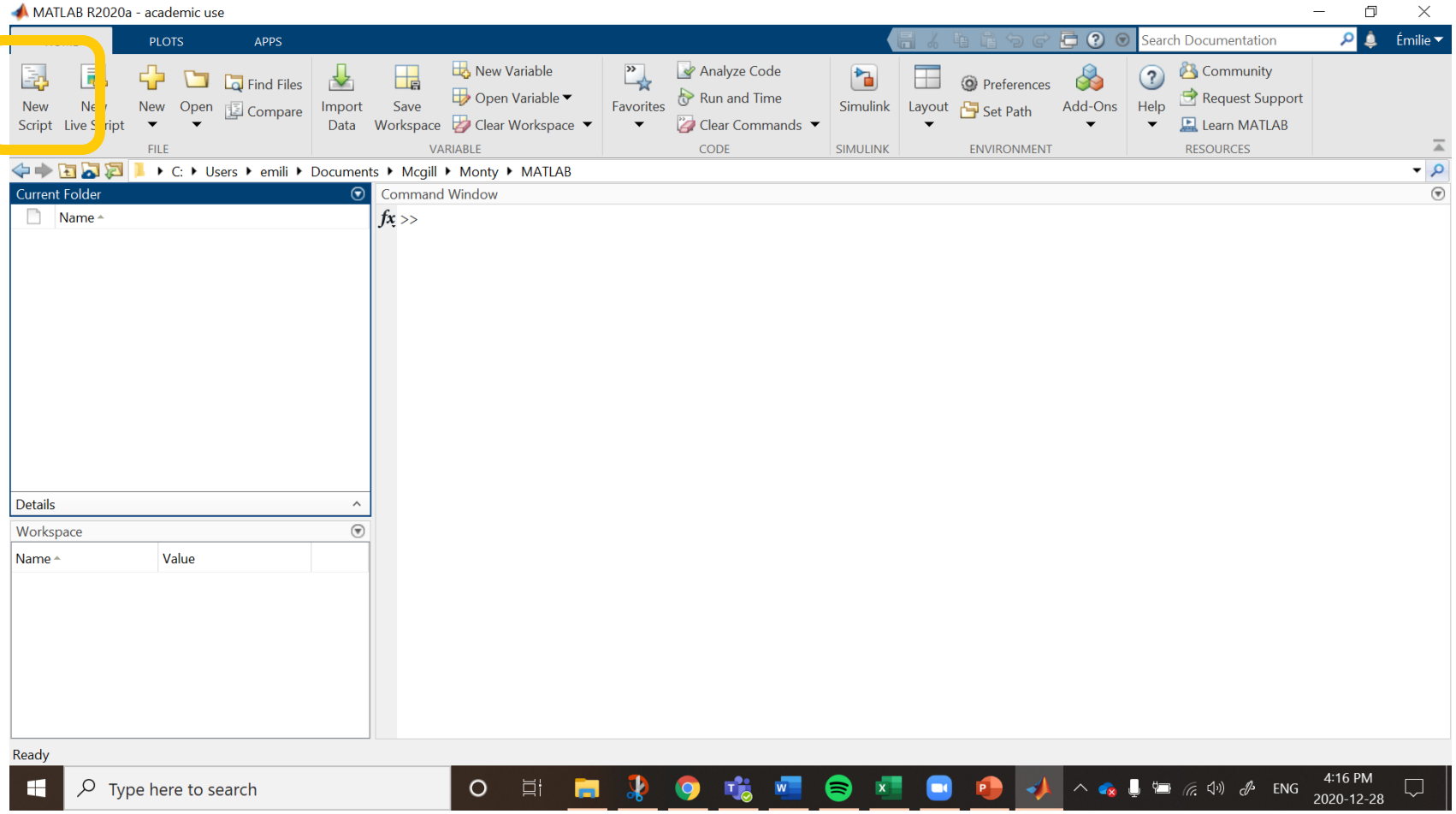
	1	2	3	4	5	6	7	8	9
1	32	324	554	318	3294	2893			
2	4132	432	65	4312	7645	14			
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									

Script & Editor Window

Most of the things that you do in MATLAB will be done in a Script

To start a Script, you can click on the New Script

This will open the Editor Window

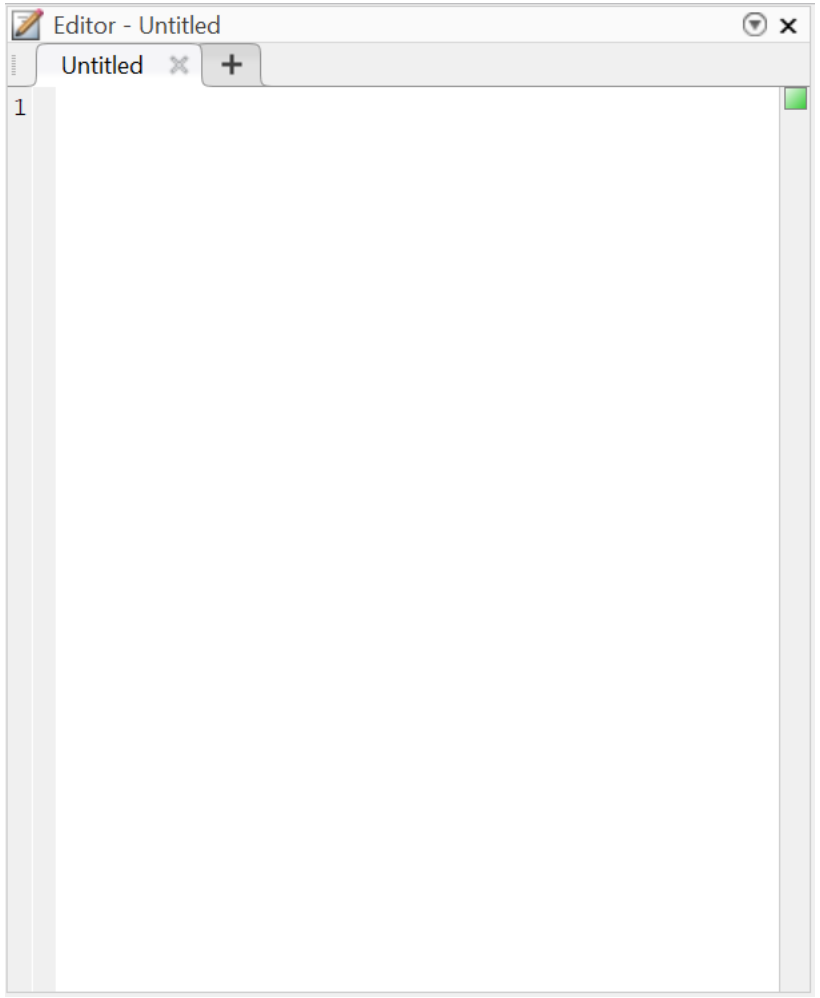


Editor Window

The editor window is where you can let your imagination go wild. Here you can write multiple line at a time to create a code that you can save and run as you wish.

The editor window is used to write both Scripts and Functions.

The line numbers of your code will be displayed on the left of the window.

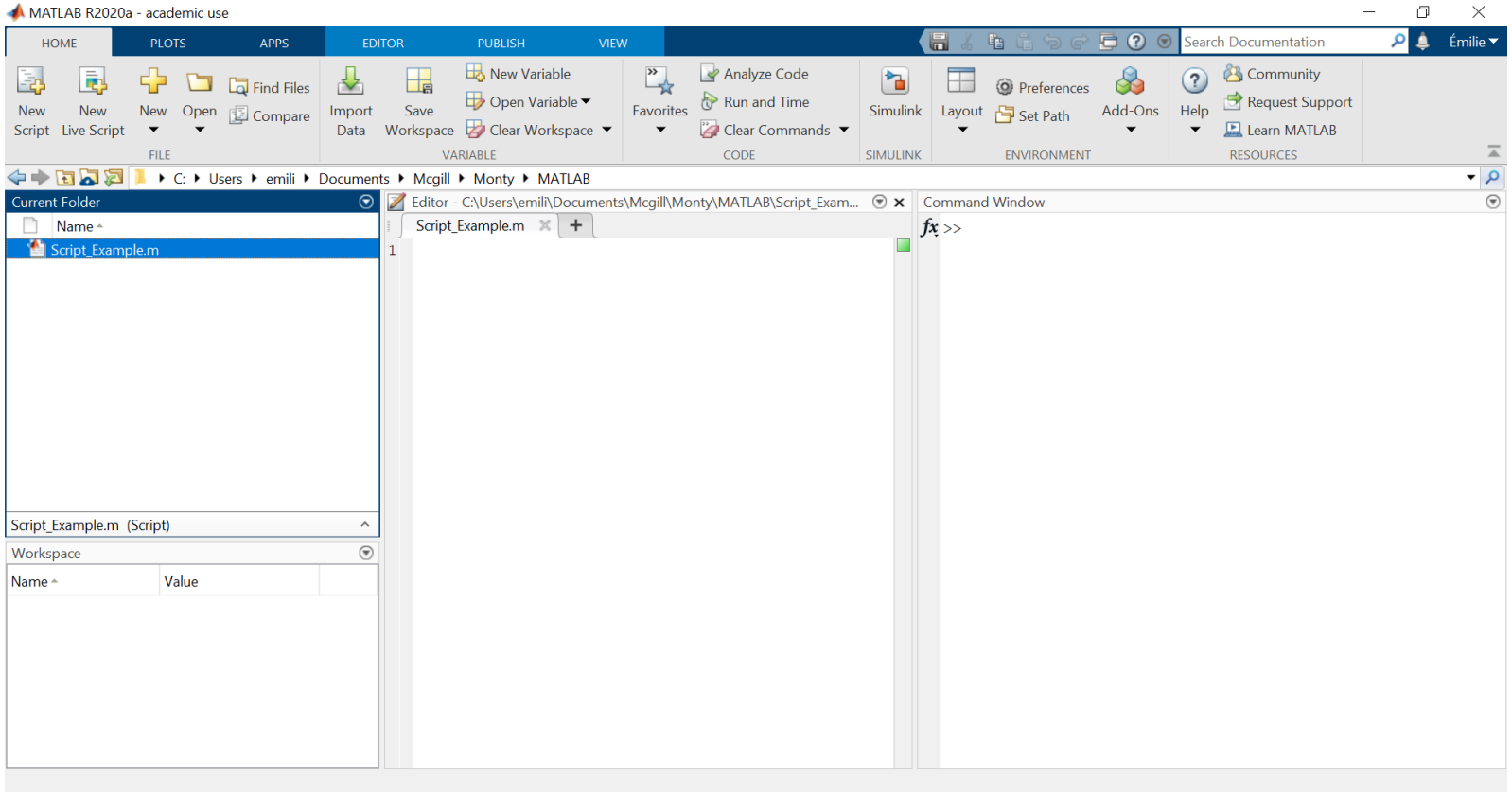


Window Layout

You can change the Window layout by dragging them around.

Just find what works best for you.

Personally, I like this layout, where my editor window is in the middle and my command window is on the right.



DATA TYPES

Array, String, Char, Cell Array, String Array, Struct

Array [ ]

- When to use?
  - To store data for calculations
- What type of array can you have
  - Row
  - Column
  - M x N matrix
- Syntax
  - Envelop in square bracket [ ]
  - Use **coma** or **space** for different **column** entries
  - Use **semicolon** for different **row**

```
%% Array
clear all
clc
x = [1 2 3 4 5 6]
y = [1,2,3,4,5,6]
z = [1;2;3;4;5;6]
M = [1,2,3;4,5,6;7,8,9]
```

Command Window

```
x =
     1     2     3     4     5     6
y =
     1     2     3     4     5     6
z =
     1
     2
     3
     4
     5
     6
M =
     1     2     3
     4     5     6
     7     8     9
```

String " "

- When to use?
  - To store and combine text
- Syntax
  - Envelop in double quotes " "
- Operation
  - You can add stings together
  - Other…

```
%% String
clear all
clc
x = "this"
y = "is"
z = " fun"
txt = x+y+z
```

```
x =
    "this"
y =
    "is"
z =
    " fun"
txt =
    "thisis fun"
```

Char ‘’

- When to use?
  - To store characters, … aka never?
- Syntax
  - Envelop in single quotes ‘’
- Operation
  - You can do operation on the individual characters

```
%% Char
clc
x = 'this '
y = '  is '
z = 'weird'
char = x+y+z
```

```
Command Window
x =
    'this '
y =
    '  is '
z =
    'weird'
char =
   267   237   315   344   164
fx >>
```

Cell Array { } and String Array [ ]

- When to use?
  - To store strings and other types (cell)
- What type of array can you have
  - Row (both)
  - Column (both)
  - M x N matrix (cell array)
- Syntax
  - Envelop in square bracket [ ] for string array and curly bracket { } for cell array
  - Use **coma** or **space** for different **column** entries
  - Use **semicolon** for different **row**

```
%% Cell Array
clear all
clc
x = {'hello'}
z = [1 2 3]
y = {'same','separation';z 'here'}
```

Command Window

```
x =
  1×1 cell array
    {'hello'}
z =
     1     2     3
y =
  2×2 cell array
    {'same'    }    {'separation'}
    {1×3 double}    {'here'      }
fx >>
```

```
%% String Array
clear all
clc
x = ["string" "array"]
```

Command Window

```
x =
  1×2 string array
    "string"    "array"
fx >>
```

Struct

- When to use?
  - To store different data files under one subject
- Syntax
  - Create with the **subject = struct()** function
  - Add to it by referring in the following way:
    - subject.object

```
%% Struct
clear all
clc
x = struct()
x.num = 1
x.arr = [1 2 3 3 4 4 5]
x.str = "hello"
```

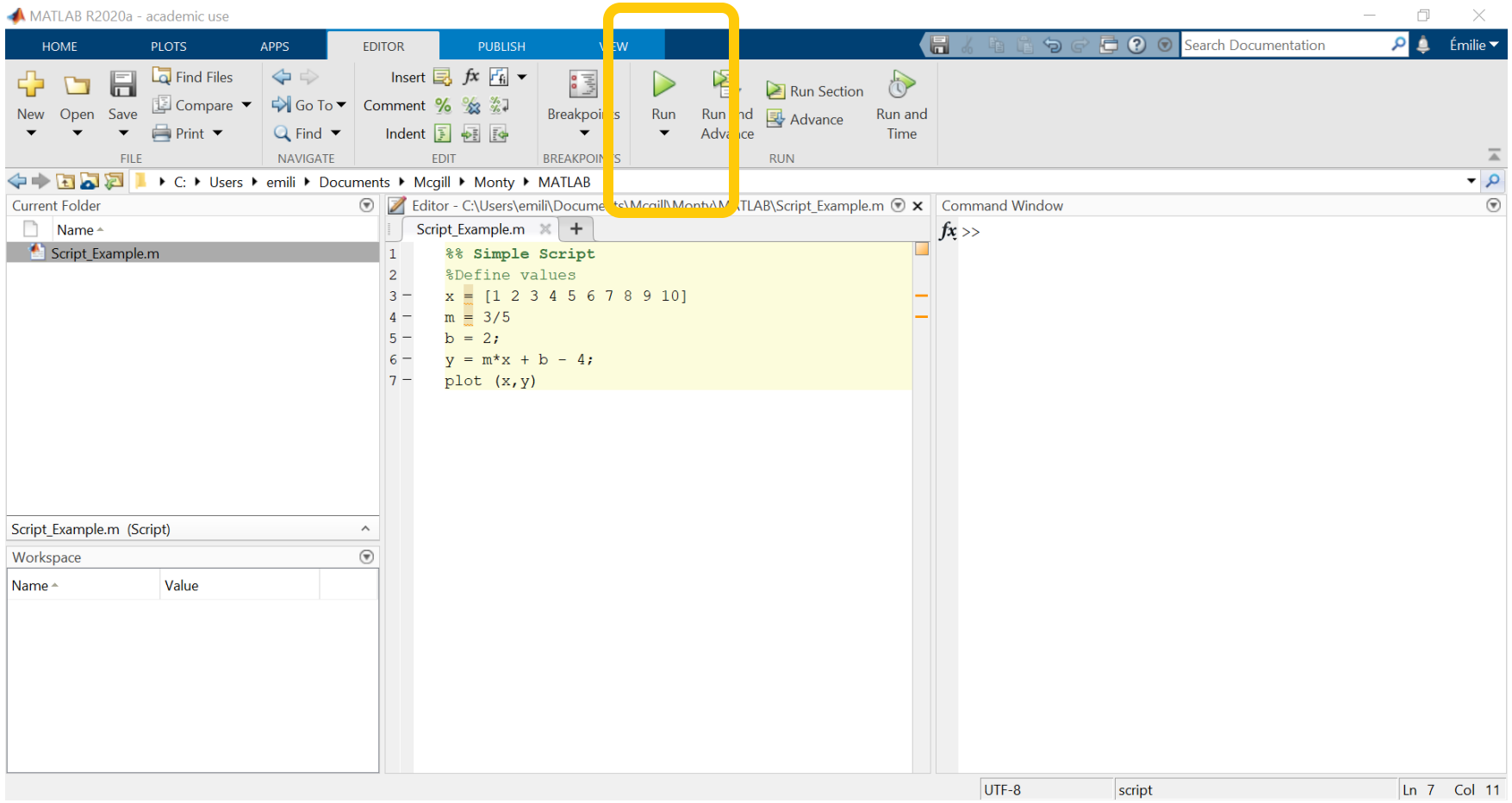
```
Command Window
  x =
    struct with no fields.
  x =
    struct with fields:

      num: 1
  x =
    struct with fields:

      num: 1
      arr: [1 2 3 3 4 4 5]
  x =
    struct with fields:

      num: 1
      arr: [1 2 3 3 4 4 5]
      str: "hello"
```

SCRIPT

Writing, Running, Debugging, ...

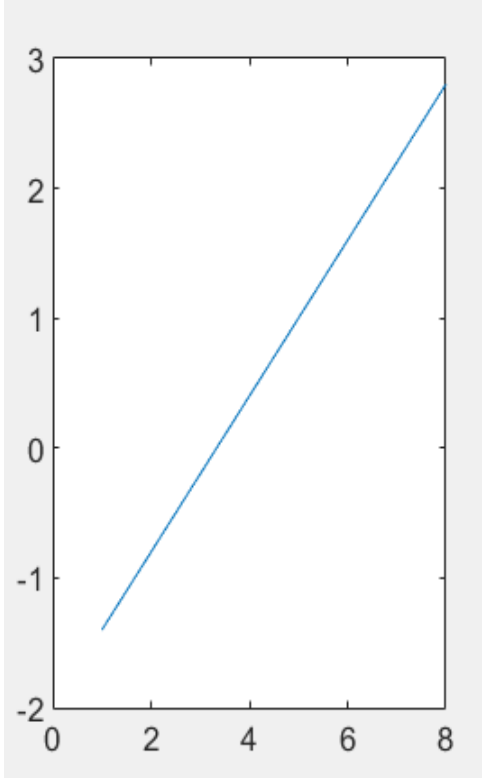
Running the Script

To run the Script, click on the green triangle titled "run"



Writing a Simple Script

%%	Section title
%	Comment
=	Assign Symbol
[ ]	Contains Arrays
/	Divide Operator
;	Output Suppression
*	Multiplication operator
+	Addition Operator
-	Subtraction Operator
plot()	Plot function



Editor - C:\Users\emili\Documents\Mc...

Script_Example.m

```
1      %% Simple Script
2      %Define values
3 -    x = [1 2 3 4 5 6 7 8]
4 -    m = 3/5
5 -    b = 2;
6 -    y = m*x + b - 4;
7 -    plot (x,y)
```

Command Window

```
>> Script_Example

x =

     1     2     3     4     5     6     7     8


m =

    0.6000
```

Debugging

Matlab runs your code line by line.

If it reaches a line of code that contains an error, it will stop and inform you of it. It will also tell you the reason for the error as well as the line on which it occurred.

You can see here that line 10 and 11 were run successfully before the error on line 12 occurred.

You can then go into your code, fix the error and run it again.

```
 9
10 -    %% Debugging
11 -    u = [2 3 4;
12 -         4 5 2]
13      v = [6 5;
             3 9 8]
```

```
u =

     2     3     4
     4     5     2

Error using vertcat
Dimensions of arrays being
concatenated are not consistent.

Error in Script_Example (line 12)
v = [6 5;
```

IMPORTING FILES

Demonstration!

TABLE

table()

table()

- Use the **table()** function to create a table
- Inputs of the function:
  - **Column** arrays
  - Array have to be the **same length**

```
%% Table
clear all
clc
txt = ["hello";"are";"you";"having";"fun";"with";"matlab";"yet"];
C = [1;2;3;5;6;7;8;5];
S = [5;6;7;1;3;9;2;9];
Table_Name = table(txt,C,S)
```

```
Table_Name =
  8×3 table
      txt          C     S
    ________       _     _
    "hello"        1     5
    "are"          2     6
    "you"          3     7
    "having"       5     1
    "fun"          6     3
    "with"         7     9
    "matlab"       8     2
    "yet"          5     9
```

EXPORTING FILES

writetable()

writetable()

- Use the **writetable()** function
- Input:
  - The table you want to export
  - The name of the file

```
%% Table
clear all
clc
txt = ["hello";"are";"you";"having";"fun";"with";"matlab";"yet"];
C = [1;2;3;5;6;7;8;5];
S = [5;6;7;1;3;9;2;9];
Table_Name = table(txt,C,S)
```

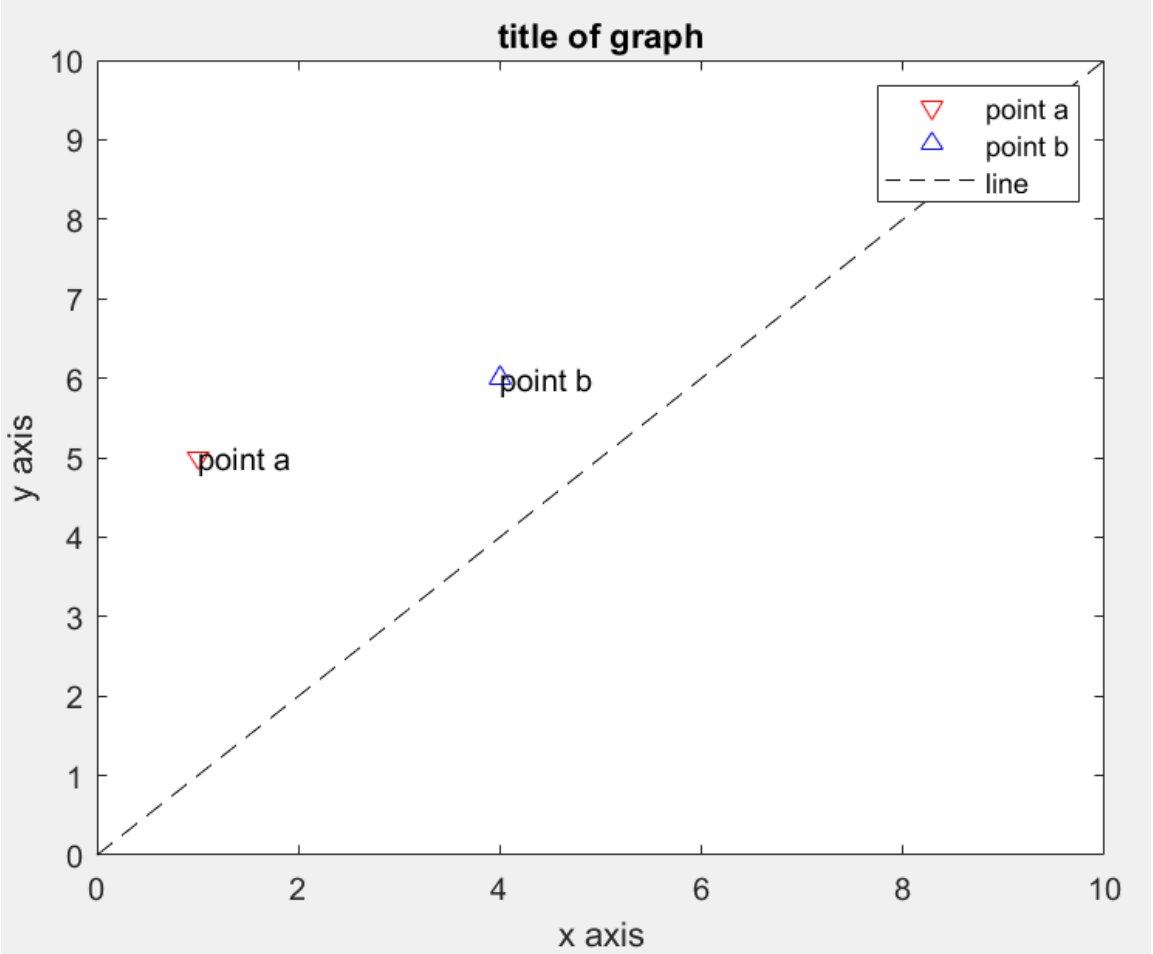
```
%% Exporting
writetable(Table_Name,'Excel_Table.xlsx')
```

PLOTS

plot()

plot()

```
%% Plot and Labelling
clear all
close all
clc
a = [1 5];
b = [4,6];
x = [0:10];
y = x;
close all
plot(a(1),a(2),'vr',b(1),b(2),'^b',x,y,'--k')
legend('point a','point b', 'line')
X = [1 4];
Y = [5 6];
txt = {'point a' 'point b'};
text(X,Y,txt);
xlabel('x axis')
ylabel('y axis')
title('title of graph')
```



USEFUL FUNCTIONS

Structures of function, pretty(), fprintf()

Using functions in Scripts

When writing scripts, you might want to use functions, which execute more complex tasks then regular operators.

Functions can be very simple, such as those already included in Matlab.

You can also create your own functions

It is also possible to download functions that have been made by other people form the internet.

Here we will first go over some easy functions that are already available for you to use.

Functions are mostly composed in this general way.

Output = FunctionName(Inputs)

pretty()

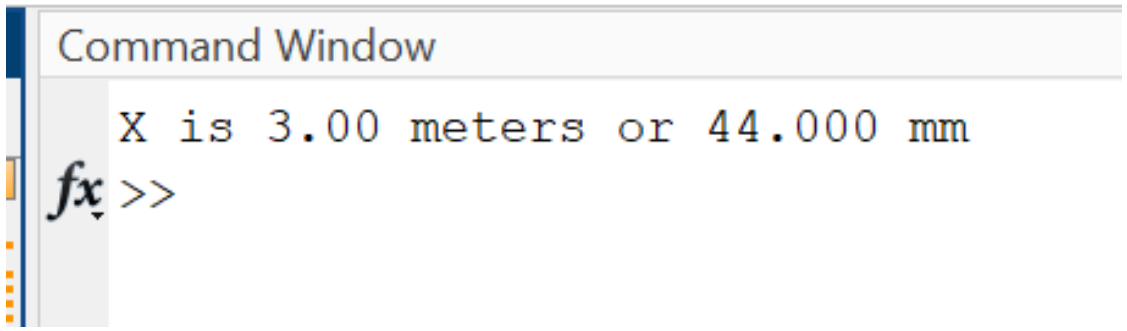
Pretty can be used to visualise the equations that you have typed into Matlab. This is a great way of verifying your parenthesis!

```
  phi5 - 2 phi50 + phi95    phi16 - 2 phi50 + phi84
- ---------------------- - ----------------------
     2 phi5 - 2 phi95          2 phi16 - 2 phi84
```

```
%% Calculate the graphic skewness of the grain-size distribution.
% Inclusive Graphic Skewness: SK_i = ((phi84+phi16-(2*phi50))/(2*(phi84-phi16))) +((phi95+phi5-(2*phi
syms phi5 phi16 phi50 phi84 phi95
Graphic_skew = ((phi84+phi16-(2*phi50))/(2*(phi84-phi16))) +((phi95+phi5-(2*phi50))/(2*(phi95-phi5)))
pretty (Graphic_skew)
```

fprintf()

```
%% Write in the command Window
clear all
clc
A1 = 3;
A2 = 44;
fprintf('X is %.2f meters or %.3f mm\n',A1,A2)
```



SEARCHING DOCUMENTATION

Demonstration

QUESTIONS?

What do you want to see in part2?

Matlab - Part 2

- Data Type part 2
- Plot part 2
- Converting Data Type
- Creating Functions
- Local Functions
- Useful Functions part 2
- Statistics
- Maps
- Apps

Map Containers

Description of The Slide

(Add a Screenshot from Matlab on the right)

hist()

Description of The Slide

(Add a Screenshot from Matlab on the right)

bar()

Description of The Slide

(Add a Screenshot from Matlab on the right)

box()

Description of The Slide

(Add a Screenshot from Matlab on the right)

CONVERTING DATA TYPE

CREATING FUNCTIONS

STATISTICS

MAPS

APPS

Title of The Slide

Description of The Slide

(Add a Screenshot from Matlab on the right)