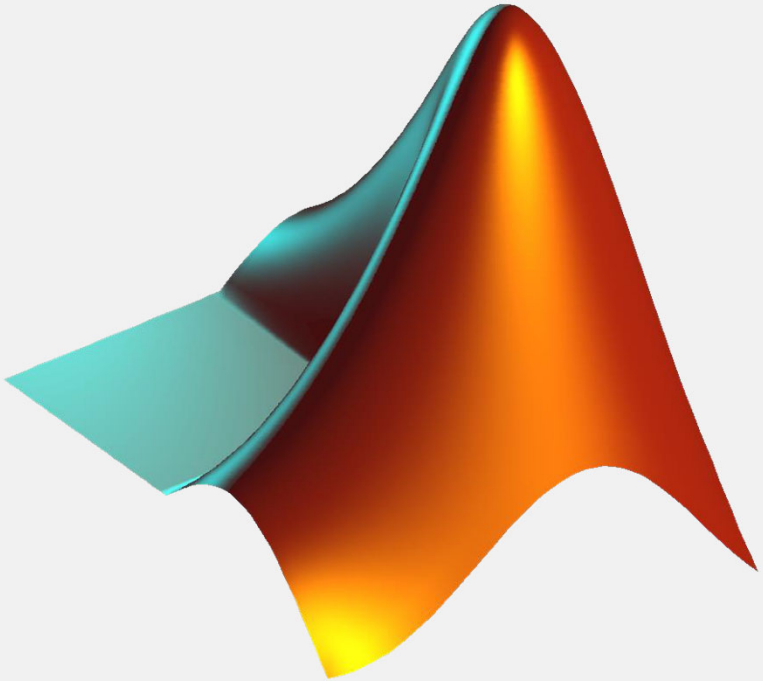
Fall 2024

# QCB W12: Intro to MATLAB

## Day 1/2

Lukasz Salwinski
lukasz@mbi.ucla.edu



UCLA
QCBio Collaboratory

Slides based on the original version by Amandine Gamble (amandinegamble@ucla.edu)
Errors/Typos/Inaccuracies by Lukasz Salwinski (lukasz@mbi.ucla.edu)

# Why MATLAB?

- High level language
  - Hides details of accessing computer hardware (eg memory management)
  - Provides access to operations on complex entities (eg matrices)
- Built-in Integrated development environment
- Simplifies numerical manipulation
- Includes graphics library (2D/3D plots of functions and data)
- Large library of efficiently implemented algorithms
- Interface with other languages
- …

# Outline of the workshop

## Day 1

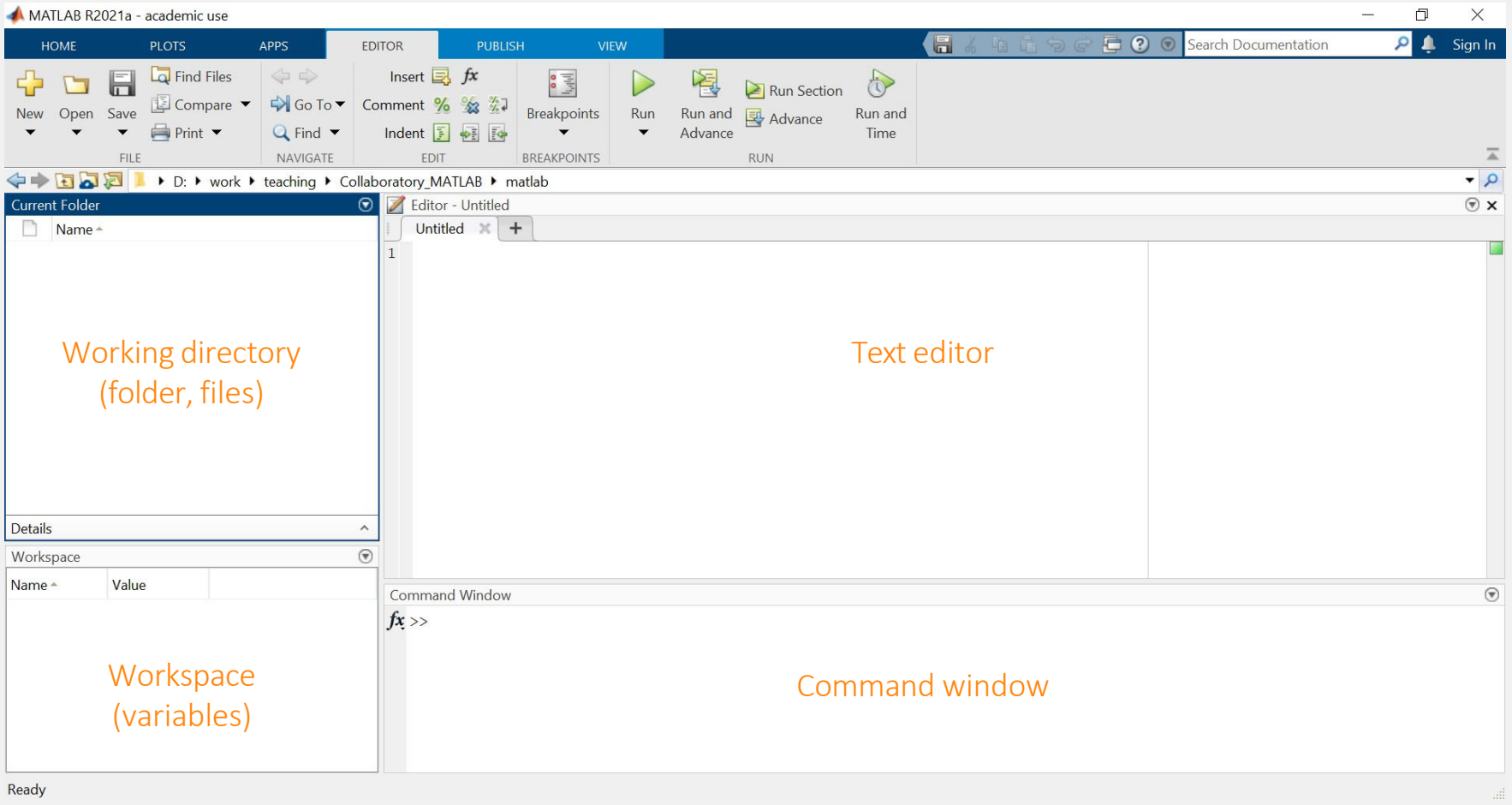
- User interface
- Console/Command line interface
- Scripts
- Variables and operations
- Matrix operations
- **`For`** and **`while`** loops
- Reading/writing files
- Data import
- Plotting GUI
  - Line plots
  - Scatter plots

## Day 2

- User-defined functions
- **`If/else`** statements
- Linear regression
- Plotting GUI
  - Box plots/violin plots
  - Surface plots
  - Heat maps

# Interface

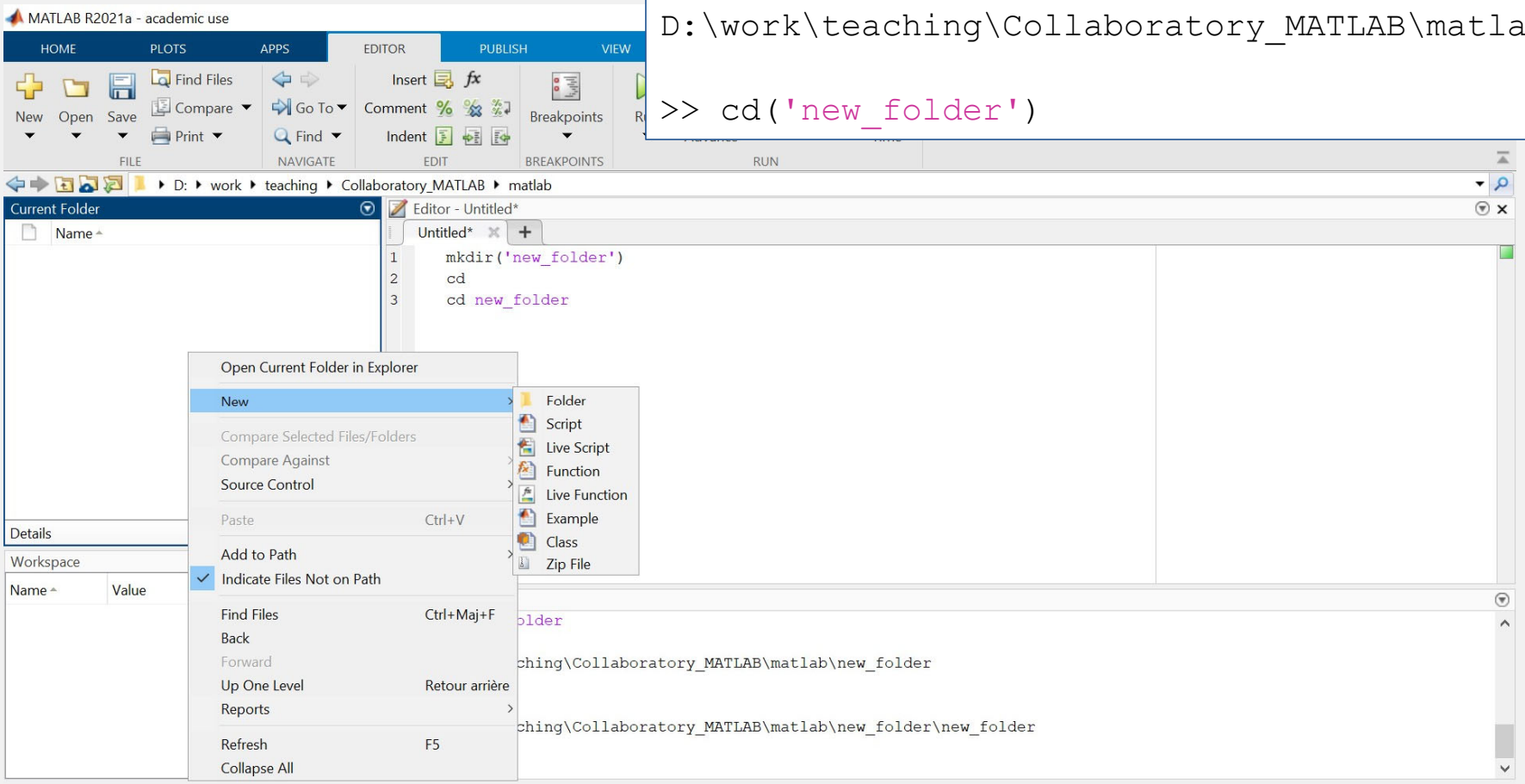
# MATLAB interface



# Getting started

- Create a new, clean folder

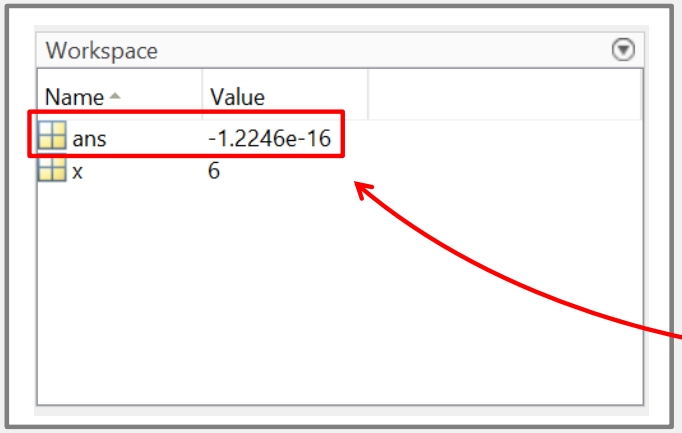
```
>> mkdir('new_folder')
>> cd

D:\work\teaching\Collaboratory_MATLAB\matlab

>> cd('new_folder')
```



# Command line interface
# and
# basic syntax

# Command line interface

- Use the Command Window as a calculator
- Note:
  - The variable **ans** → store the result
  - **pi** variable (and others – most notably i) defined by default – Watch out !!!
  - ; → hide the result (still stored)
  - Find current value of **ans** (and other variables) in Workspace.




```
>> 1+1

ans =

     2

>> ans

ans =

     2

>> 1*2

ans =

     2
```

```
>> ans

ans =

   1.2246e-16

>> sin(pi);
>> sin(pi)

ans =

   1.2246e-16

>> tan(pi);
>> ans

ans =

  -1.2246e-16
```

# Command line interface

A. Control format with **format**
   - default, long, short, rat, bank, …
B. Use **disp** for explicit displaying
   Equivalent to no ;
C. Find your variables with **who** and **whos**
D. Clean your workspace with **clear**

```
>> who

Your variables are:

ans   x
```
C

```
>> pi

ans =

    3.1416

>> format long
>> pi

ans =

   3.141592653589793

>> format short
>> pi

ans =

    3.1416
```
A

```
>> format bank
>> pi

ans =

            3.14

>> format rat
>> pi

ans =

      355/113
```
A

```
>> disp(pi)
    3.1416
```
B

```
>> clear x
>> clear
```
D

# Command line interface

Keeping track of progress

Note: Saves/restores workspace state but **not** operations

- Before closing MATLAB

```
>> save FileName
```

- When re-opening MATLAB

```
>> load FileName.mat
```

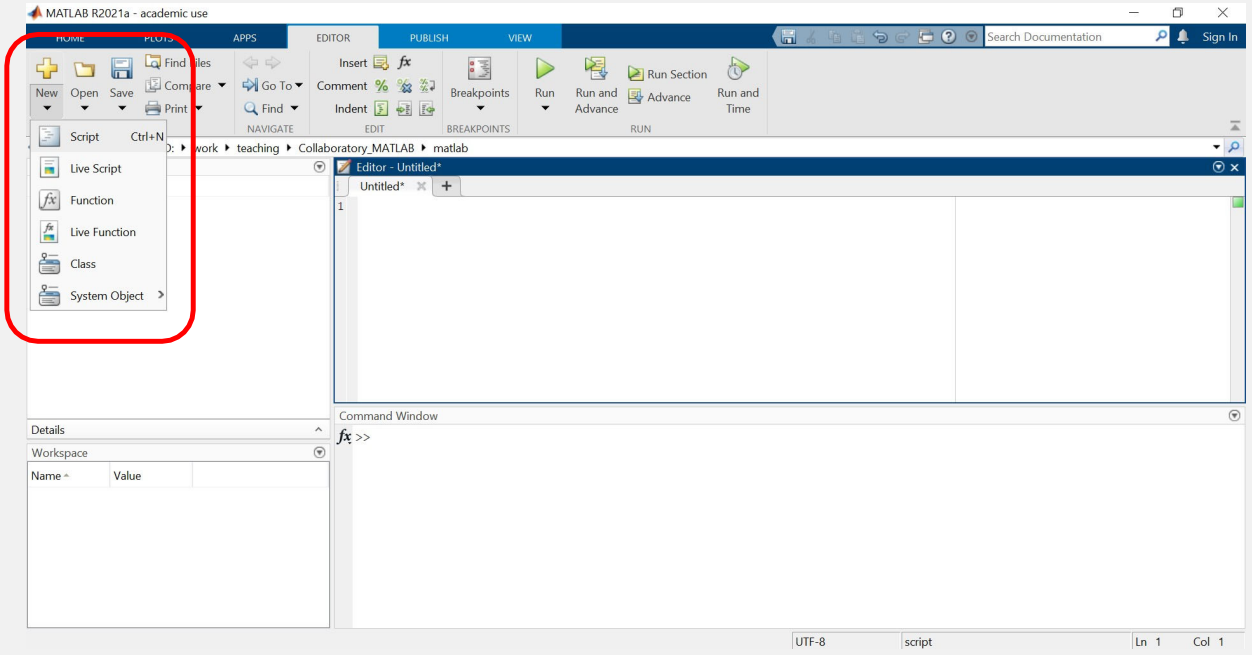
# Scripts

# Running code

- 3 ways of running code
  - Command window
  - Scripts
  - Functions

  } m-files (code)

```
>> edit W12_Day1_example
```

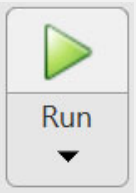


# Running code

- In the text editor

```
% QCB W12: Intro to MatLab
% Day 1: example
% ---------------------------------

a = 1;
b = 2;
c = 3;

A
```

- Look at the command window




```
>> run W12_Day1_example

a =

     1
```

# Keep your code readable

- Commenting code with %
  - For your future self
  - For colleagues using your code too
  - For the readers of your future paper (open-science)

```
>> tan(pi) %calculate the tangent of pi

ans =

  -1.2246e-16
```

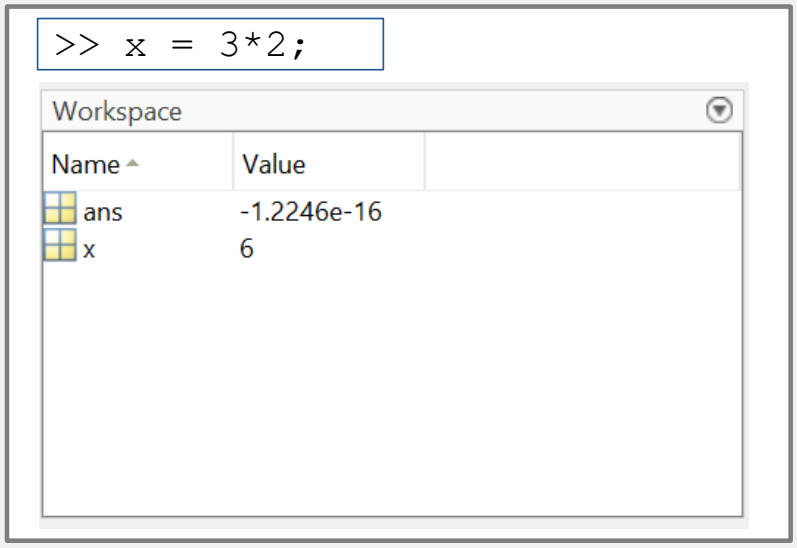
# Variables and operations

# Creating variables

- `variableName = variableValue`
- Created during the first assignment
- Naming variables
  - Watch out for default variables (pi, i,…) → Can be over-written → Check first…
  - Case sensitive names

```
>> i

ans =

   0.0000 + 1.0000i
```

```
>> x = 2

x =

     2

>> X
Unrecognized function or variable 'X'.

Did you mean:
>> x
```



# Variables: matrices

- Everything is a matrix for MATLAB

  *Number = 1 by 1 matrix*

- **`length`** returns the largest dimension
- **`size`** returns # rows & # of columns
- Vectors = 1 by *x* matrix
- Row vs column vector
  - `,` → On the same line
  - `;` → On the next line

```
>> x = pi;

>> size(x)

ans =

     1     1
```

# rows

# columns

```
>> columnVector = [1; 2; 3; 4];

>> size (columnVector)

ans =

     4     1
```

# rows

# columns

# Variables: metrices

- MATLAB performs operations on matrices
- Conventional matrix algebra operations
- Extensions for some operations (eg **+**/**−**)

```
>> firstVector = [1, 2, 3, 4];

>> size (firstVector)

ans =

     1     4
```


```
>> secondVector = [4, 3, 2, 1];

>> firstVector + secondVector

ans =

     5     5     5     5
```

```
>> sameVector = [1:4]

sameVector =

     1     2     3     4
```

# Variables: matrices

- MATLAB performs operations on matrices
- Conventional matrix algebra operations
- Extensions for some operations (eg **+**/**−**)
- Transpose (**'**) & complex transpose (**.'**)

```
>> rowVector

rowVector =

     1     2     3     4

>> rowVector'

ans =

     1
     2
     3
     4
```

# Variables: matrices

- MATLAB performs operations on matrices
- Conventional matrix algebra operations
- Extensions for some operations (eg **+**/**−**)
- Transpose (**'**) & complex transpose (**.'**)
- Element-wise operations preceded with **.**

```
>> myMatrix = [1,2,3;4,5,6];

>> myMatrix2 = [4,5,6;1,2,3];

>> myMatrix * myMatrix2
Error using  *

>> myMatrix .* myMatrix2

ans =

     4    10    18
     4    10    18
```

# Variables: matrices

- Concatenating matrices with `cat`

```
>> cat(1, myMatrix, myMatrix2)

ans =

     1     2     3
     4     5     6
     4     5     6
     1     2     3
```

Dimension

```
>> cat(2, myMatrix, myMatrix2)

ans =

     1     2     3     4     5     6
     4     5     6     1     2     3
```

```
>> [myMatrix; myMatrix2]

ans =

     1     2     3
     4     5     6
     4     5     6
     1     2     3
```

```
>> [myMatrix, myMatrix2]

ans =

     1     2     3     4     5     6
     4     5     6     1     2     3
```

# Variables: formatting output

- Format output with `fprintf`
  - **`%s`** Format as a string.
  - **`%d`** Format as an integer.
  - **`%f`** Format as a floating point value.
  - **`%e`** Format as a floating point value in scientific notation.
  - **`%g`** Format in the most compact form: **`%f`** or **`%e`**.
  - **`\n`** Insert a new line in the output string.
  - **`\t`** Insert a tab in the output string.

```
>> myGene = 'Y039W';
>> myExp = 0.001234;
>> disp(myGene, myExp)
Error using disp
Too many input arguments.
```

```
>> fprintf('%s:\t%8.3e\n', myGene,myExp)
Y039W:   1.234e-03
```

# `for` loops

# **for** loop

```
% for loop execution
for ii = [1,2,3]
   disp(ii)
end
```

← Executed once of every value of **ii**

```
>>
     1
     2
     3
```

# Working with files

# Working with data files: Writing

- Use **save**
- Use **fprintf( file, format, val1,…)**

```
mydata = [];     % say, #row times #column matrix


myOutFile = "my_data.tab";

fout = fopen( myOutFile, 'w');

for jj = 1:size(mydata,1)
    fprintf(fout, '%6.3f\t', mydata(jj,:));
    fprintf(fid,"\n");
end
fclose(fid);
```

Note: Implicit iteration over columns !!!

# Working with data files: Reading

- Use **load**
- Use **importdata( filename,...)**

```
>> D = importdata( myFile )
>> size(D)
ans =
         1000           10
```

Numerical data: matrix

- Use **importtable(myFile,options)**

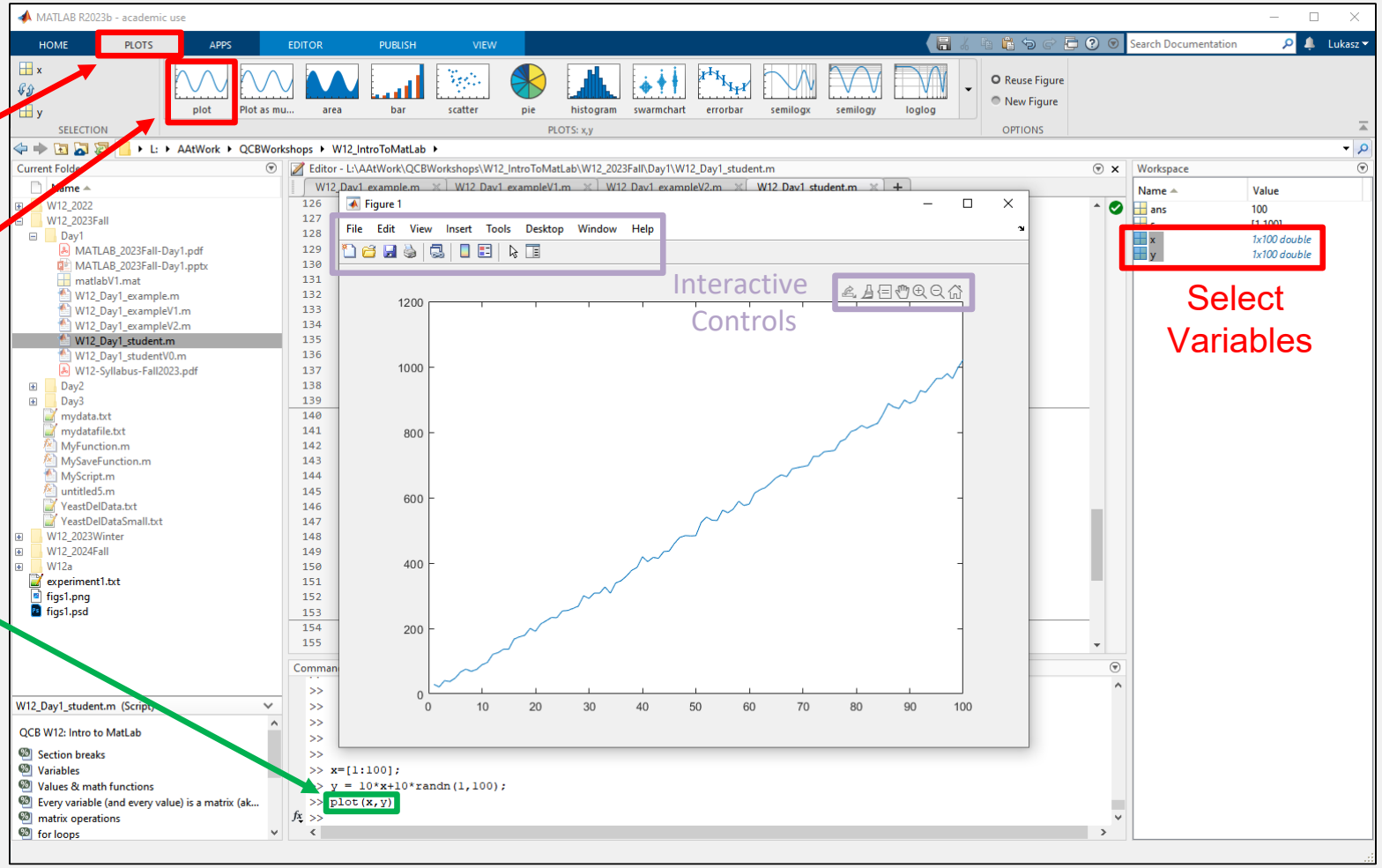
# Plotting 101

# Plotting

- Use GUI

- Script

```
plot(x,y)
scatter(x,y)
...
```



# Plotting

- Interactive controls
  - Pan/Zoom/Reset
  - Highlight data point(s)
  - Highlight data range(s)
  - Legend
  - Color scale (contours, heat maps)
- Fine-grain control
  - Property Inspector
    - Titles
    - Fonts
    - Ticks
    - Grids
    - …