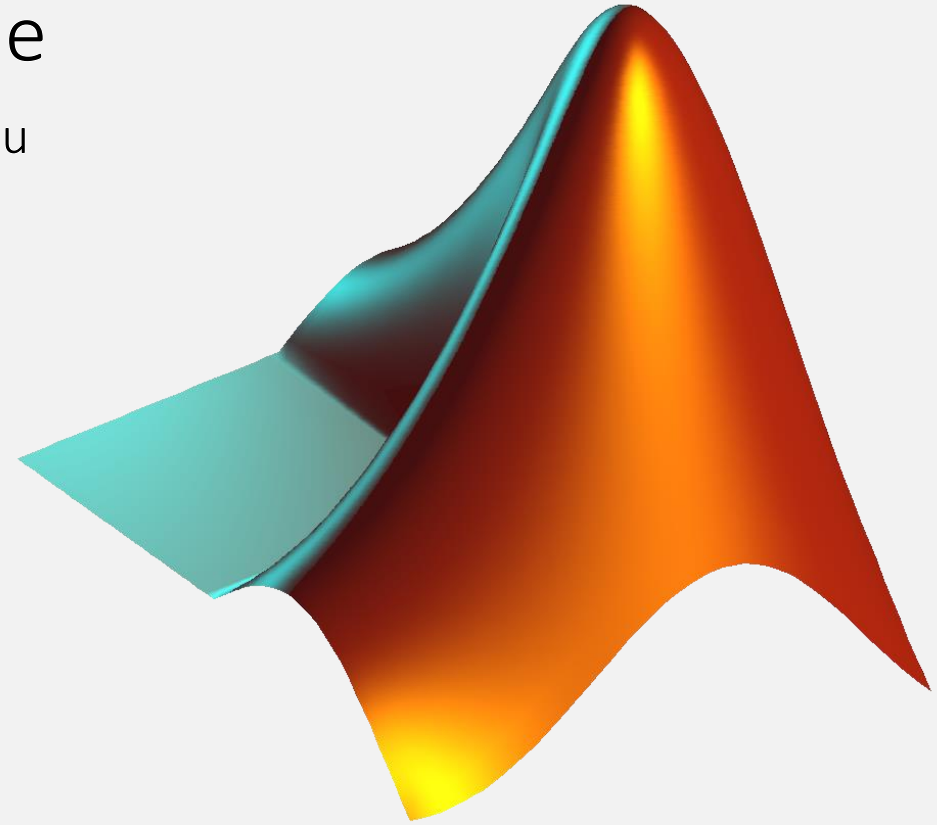
Winter 2022

Intro to MATLAB

Part 1/3

Amandine Gamble

amandinegamble@ucla.edu



UCLA

QCBio Collaboratory

The instructor

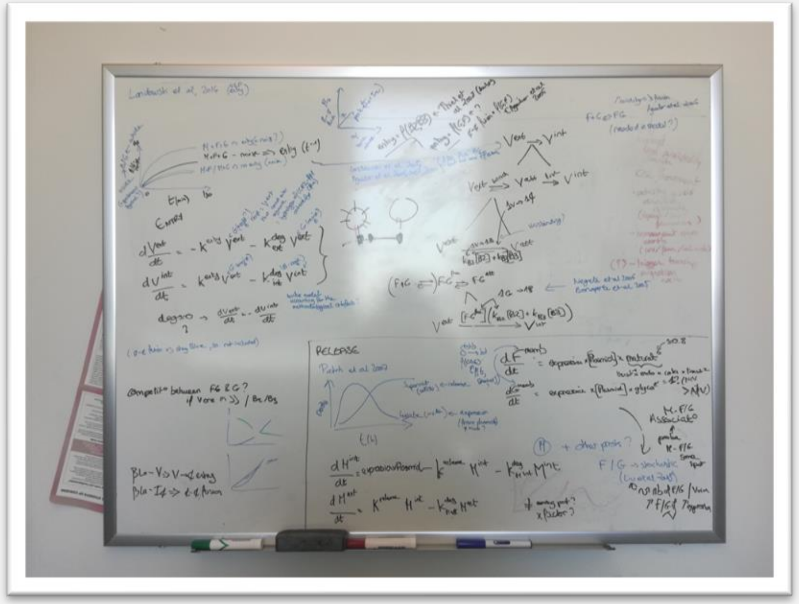
Amandine Gamble

Amanda + Marilyn

Infectious disease dynamics in wildlife

Postdoc @Lloyd-Smith Lab, Ecology and Evolutionary Biology

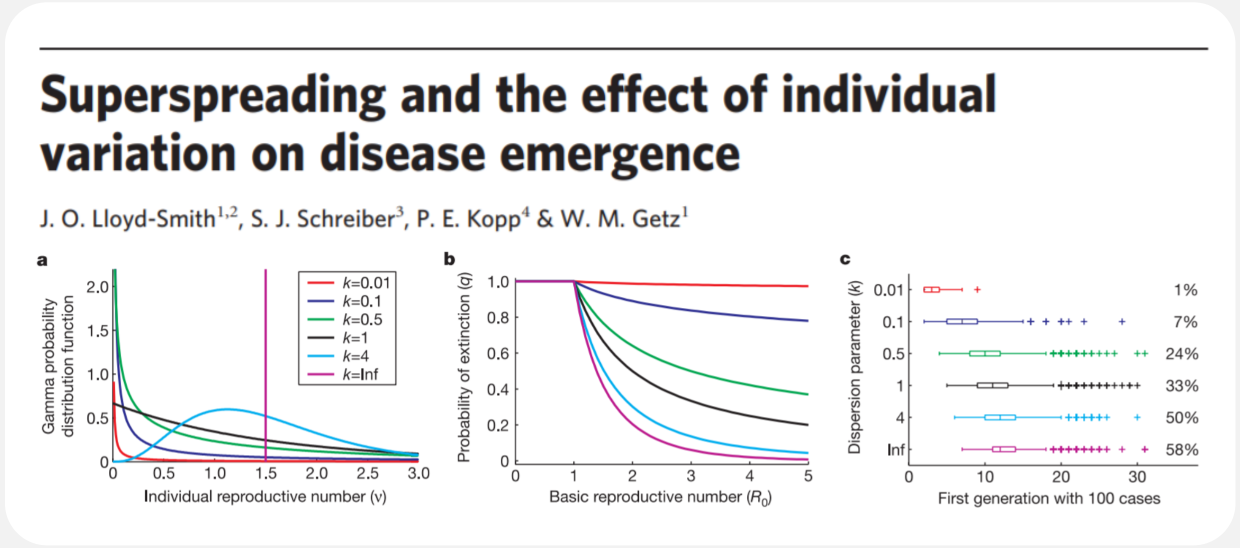
From veterinary medicine → To mathematical modelling



Why MATLAB?

- High level language
- Programing language and development environment
- Built-in development tools
- Numerical manipulation
- Plotting of functions and data
- Implement algorithms
- Create models and applications
- Many built in functions
- Interface with other languages
- …

What about you?



Superspreading and the effect of individual variation on disease emergence

J. O. Lloyd-Smith[1,2], S. J. Schreiber[3], P. E. Kopp[4] & W. M. Getz[1]

Lloyd-Smith et al. 2005. Superspreading and the effect of individual variation on disease emergence. *Nature*

W10: Mathematical Modeling of Cell Signaling

Outline of the workshop

Day 1

- **Interface**
- **Command lines and basic syntax**
- **Variables and operations**
- **Scripts**
- **`if` statements**

Day 2

- `for` and `while` loops
- More matrices
- Functions
- Files

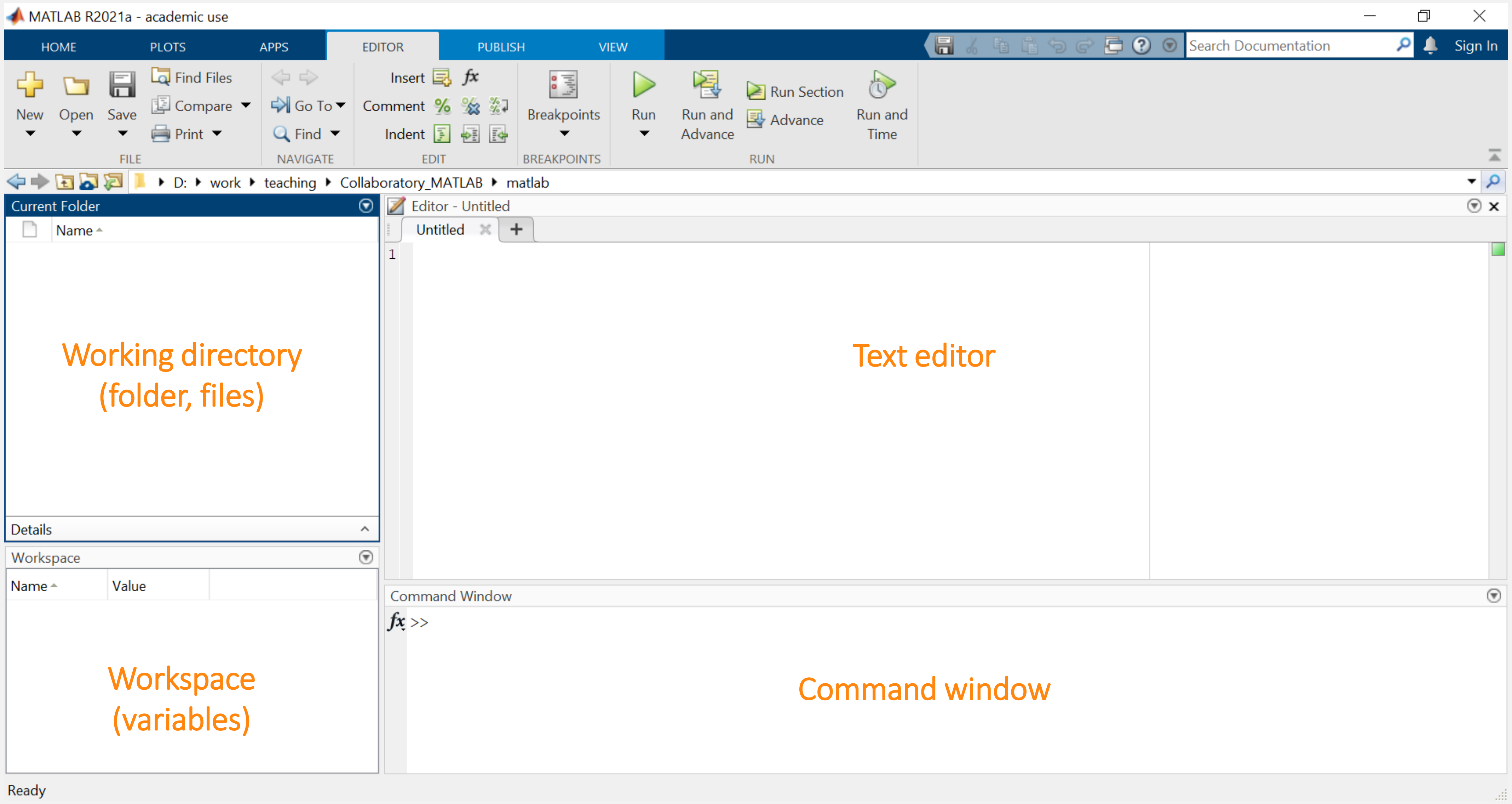
Day 3

- Plotting
- Introduction to dynamical systems: ODEs

Disclaimer: it is the first time I teach this workshop

Interface

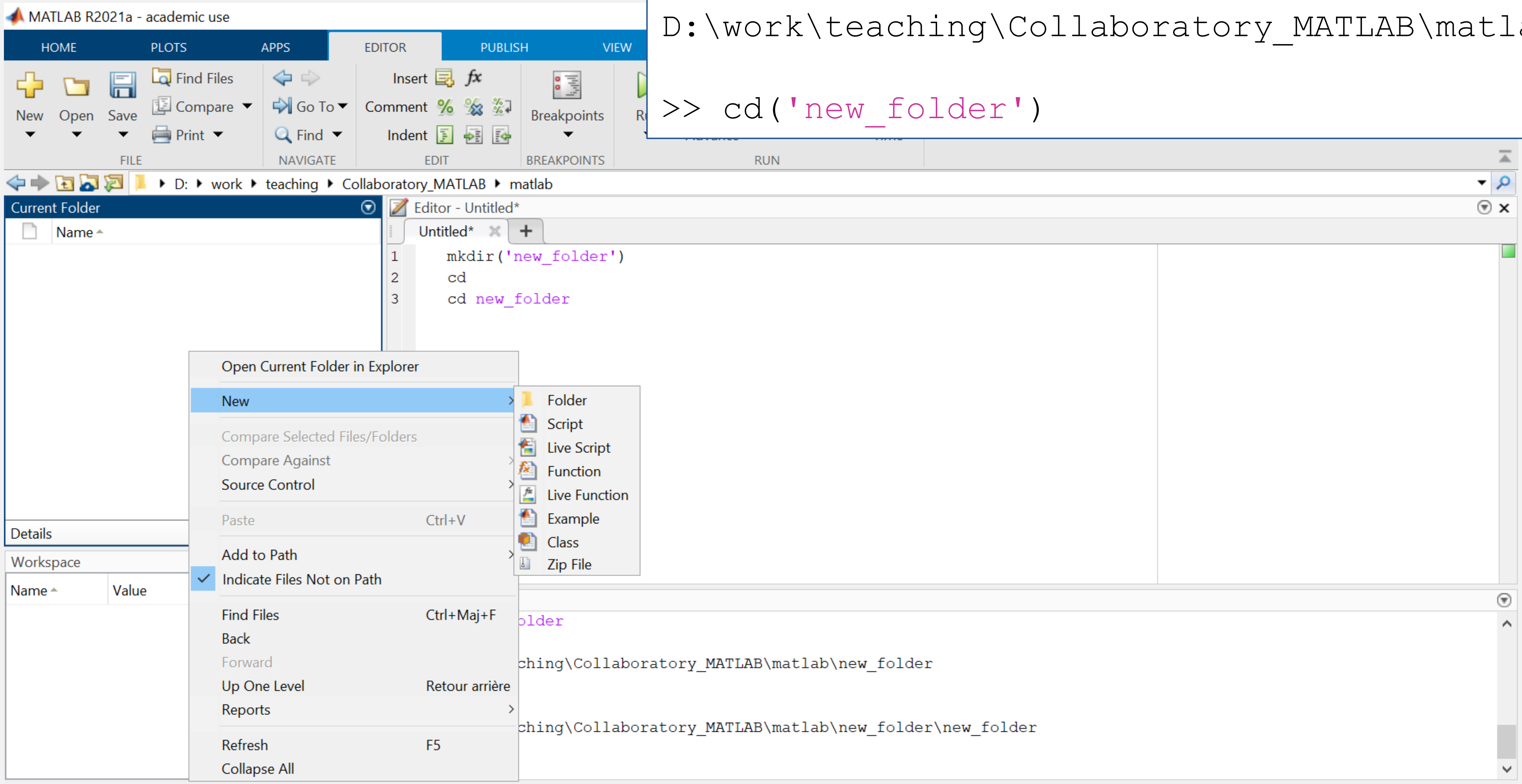
MATLAB interface



Geeting started

- Create a new, clean folder

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

D:\work\teaching\Collaboratory_MATLAB\matlab

>> cd('new_folder')
```



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

```
older
ching\Collaboratory_MATLAB\matlab\new_folder
ching\Collaboratory_MATLAB\matlab\new_folder\new_folder
```

Command lines and basic syntax

Our first command lines

- Use the Command Window as a calculator
- Notice
  - The variable `ans` → store the result
  - `pi` variable defined by default

```
>> 1+1

ans =

     2

>> ans

ans =

     2

>> 1*2

ans =

     2
```

```
>> 1/0

ans =

   Inf

>> 2*(3*3)

ans =

    18

>> sin(pi)

ans =

   1.2246e-16
```

Our first command lines

- Use the Command Window as a calculator
- Notice
  - The variable `ans` → store the result
  - `pi` variable defined by default
  - `;` → hide the result (still stored)

```
>> ans

ans =

   1.2246e-16

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

ans =

   1.2246e-16

>> tan(pi);
>> ans

ans =

  -1.2246e-16
```

Keep your code literate

- 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
```

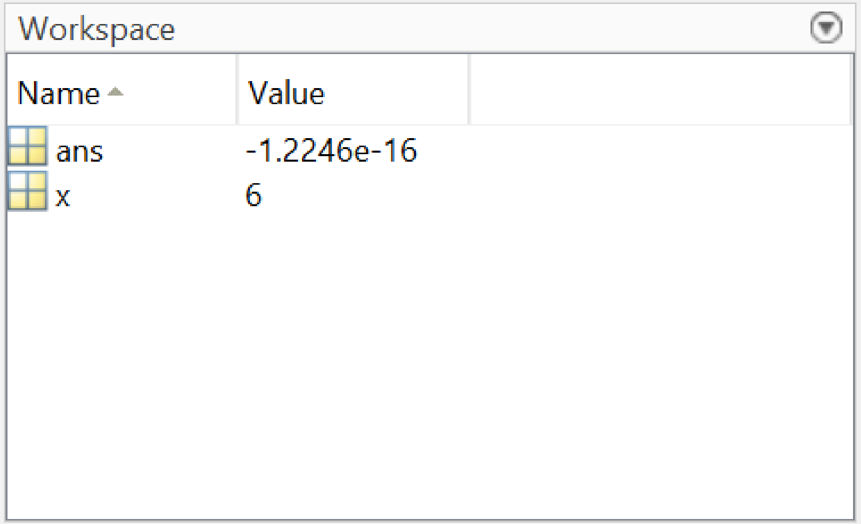
Variables and operations

Creating variables

- `variable_name = variable_value`

Watch out for default variables (pi, i,...)
Can be over-written
Check first...

```
>> x = 3*2; % define x
```


Workspace

Name	Value
ans	-1.2246e-16
x	6

```
>> 3*x

ans =

    18
```

Creating variables

- `variableName = variableValue`
- Naming variables
  - Watch out for default variables (pi, i,…) → Can be over-written → Check first…
  - Case sensitivity

```
>> i

ans =

   0.0000 + 1.0000i
```

```
>> x = 2

x =

     2

>> X
Unrecognized
function or variable
'X'.

Did you mean:
>> x
```

Saving your progress

- Before closing MATLAB

```
>> save FileName
```

- When re-opening MATLAB

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

Keeping your workspace tidy

- Find your variables with `who` and `whos`
- Clean your workspace with `clear`

```
>> who

Your variables are:

ans   x
```

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

Displaying

- Control format with `format`
- `disp` for explicit displaying
  Equivalent to no `;`

```
>> pi

ans =

    3.1416

>> format long
>> pi

ans =

   3.141592653589793

>> format short
>> pi

ans =

    3.1416
```

```
>> format bank
>> pi

ans =

         3.14

>> format rat
>> pi

ans =

     355/113
```

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

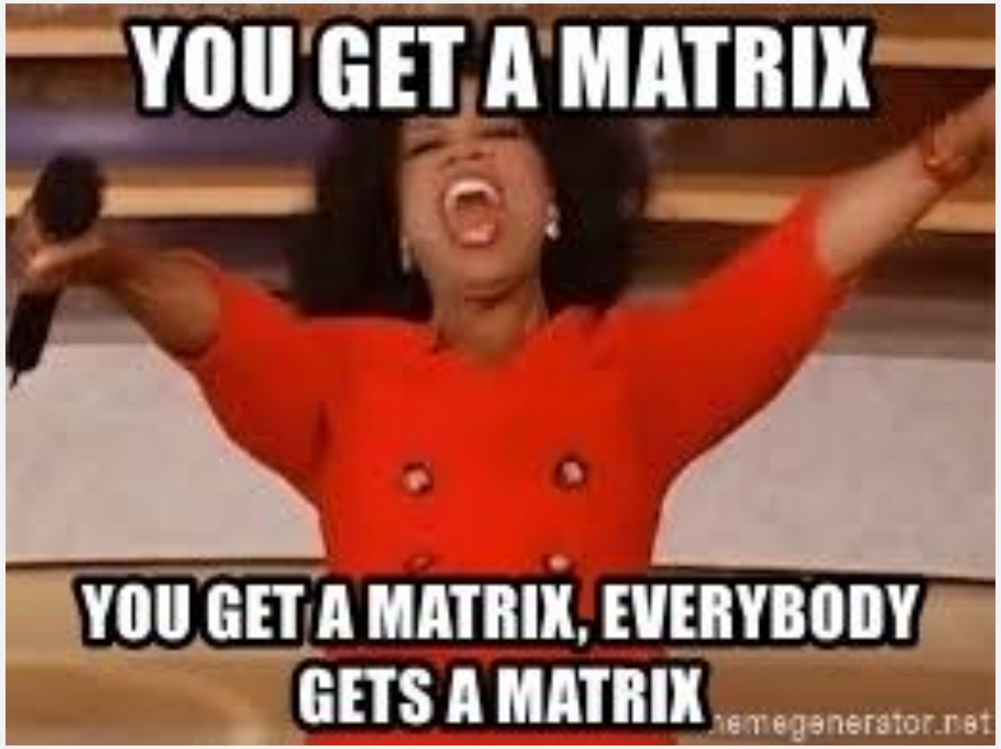
Numbers, vectors, and matrices

- Everything is a matrix for MATLAB
  Number = 1 by 1 matrix
- Nb: `length` returns the largest dimension

```
>> x = pi;

>> size(x)

ans =

     1     1
```

# rows

# columns



Numbers, vectors, and matrices

- Vectors = 1 by *x* matrix
- MATLAB can perform operations on vectors

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

>> size (firstVector)

ans =

     1     4
```

# rows

columns

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

>> firstVector + secondVector

ans =

     5     5     5     5
```

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

sameVector =

     1     2     3     4
```

Numbers, vectors, and matrices

- Vectors = 1 by *x* matrix
- Row vs column vector
  - `,` → On the same line
  - `;` → On the next line

```
>> columnVector + rowVector

ans =

     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8
```

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

>> size (columnVector)

ans =

     4     1
```

# rows

columns

```
>> rowVector =  firstVector
```

Numbers, vectors, and matrices

- Vectors = 1 by *x* matrix
- Row vs column vector
  - `,` → On the same line
  - `;` → On the next line
- Transposition with `

```
>> rowVector

rowVector =

     1     2     3     4

>> rowVector'

ans =

     1
     2
     3
     4
```

Numbers, vectors, and matrices

- *x* by *y* matrix

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

myMatrix =

     1     2     3
     4     5     6

>> size(myMatrix)

ans =

     2     3
```

Numbers, vectors, and matrices

- Matrices operations
    - Matrix-wise (default)
    - Element-wise → add `.` before operator

```
>> 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
```

Numbers, vectors, and matrices

- Matrices operations
  - Matrix-wise (default)
  - Element-wise → add `.` before operator
  - Some functions work column-wise

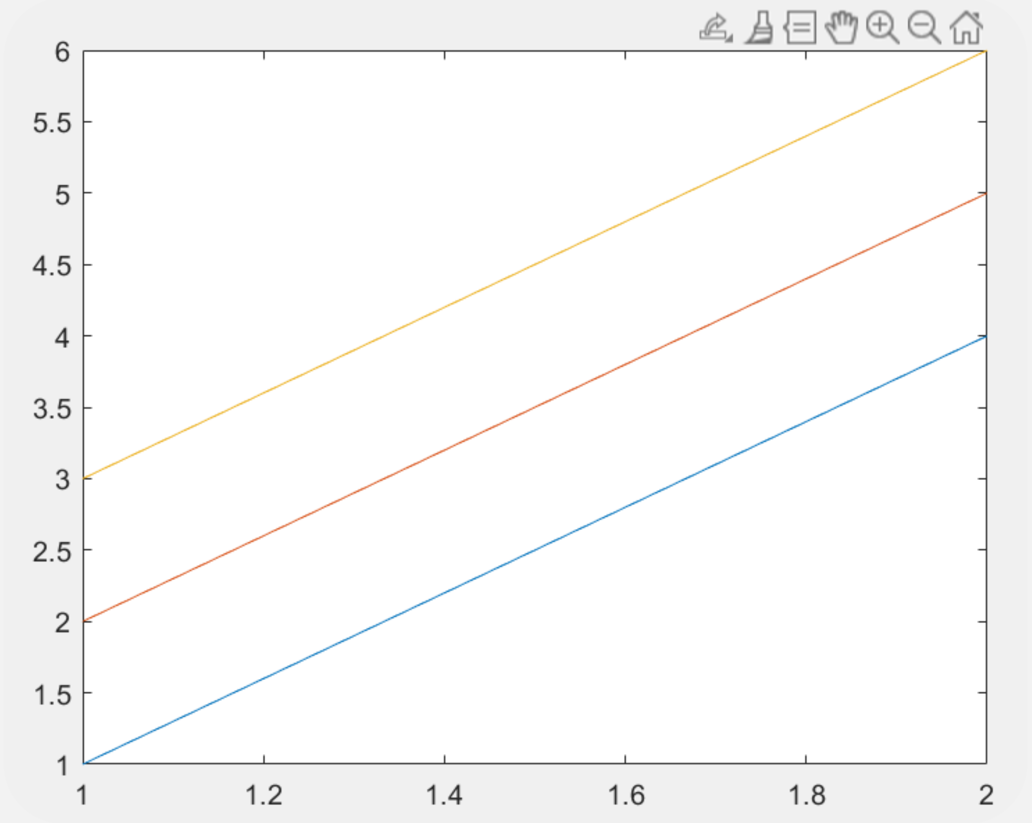
```
>> max(myMatrix)

ans =

     4     5     6

>> mean(myMatrix)

ans =

    2.5000    3.5000    4.5000
```

```
>> 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
```

Numbers, vectors, and matrices

- Matrices operations
  - Matrix-wise (default)
  - Element-wise → add `.` before operator
  - Some functions work column-wise

```
>> plot(myMatrix)
>> myMatrix

myMatrix =

     1     2     3
     4     5     6
```



Numbers, vectors, and 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
```

Numbers, vectors, and matrices

- Finding non-zero elements in a matrix with `find`
  Counts down then across

```
>> find([0,0,0,1])

ans =

     4

>> find([0,0,0,1;0,0,0,1])

ans =

     7
     8

>> find([0,0,0,1;0,1,1,1])

ans =

     4
     6
     7
     8
```

Strings

- Print text 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.

```
>> myName = 'Amandine';

>> fprintf(myName);
Amandine>>
```

```
>> fprintf('%s \n', myName)
Amandine
```

```
>> myNumberOfCats = 0;
>> fprintf('My name is %s and I have %d cat(s) \n', myName, myNumberOfCats)
My name is Amandine and I have 0 cat(s)
>> fprintf('My name is %s and I have %s cat(s) \n', myName, myNumberOfCats)
My name is Amandine and I have   cat(s)
```

Check data types

- No need to declare variables
- Check with `class`
  - single - single precision numerical data
  - double - double precision numerical data
  - logical - logical values of 1 or 0, represent true and false respectively
  - char - character data (strings are stored as vector of characters)
  - cell array - array of indexed cells, each capable of storing an array of a different dimension and data type
  - structure - named fields capable of storing an array of a different dimension and data type
  - function handle - pointer to a function
  - user classes - objects constructed from a user-defined class int8 uint8 int16 uint16 int32 uint32 int64 uint64…

```
>> class(myName)

ans =

    'string'

>> class(myMatrix)

ans =

    'double'
```

Scripts

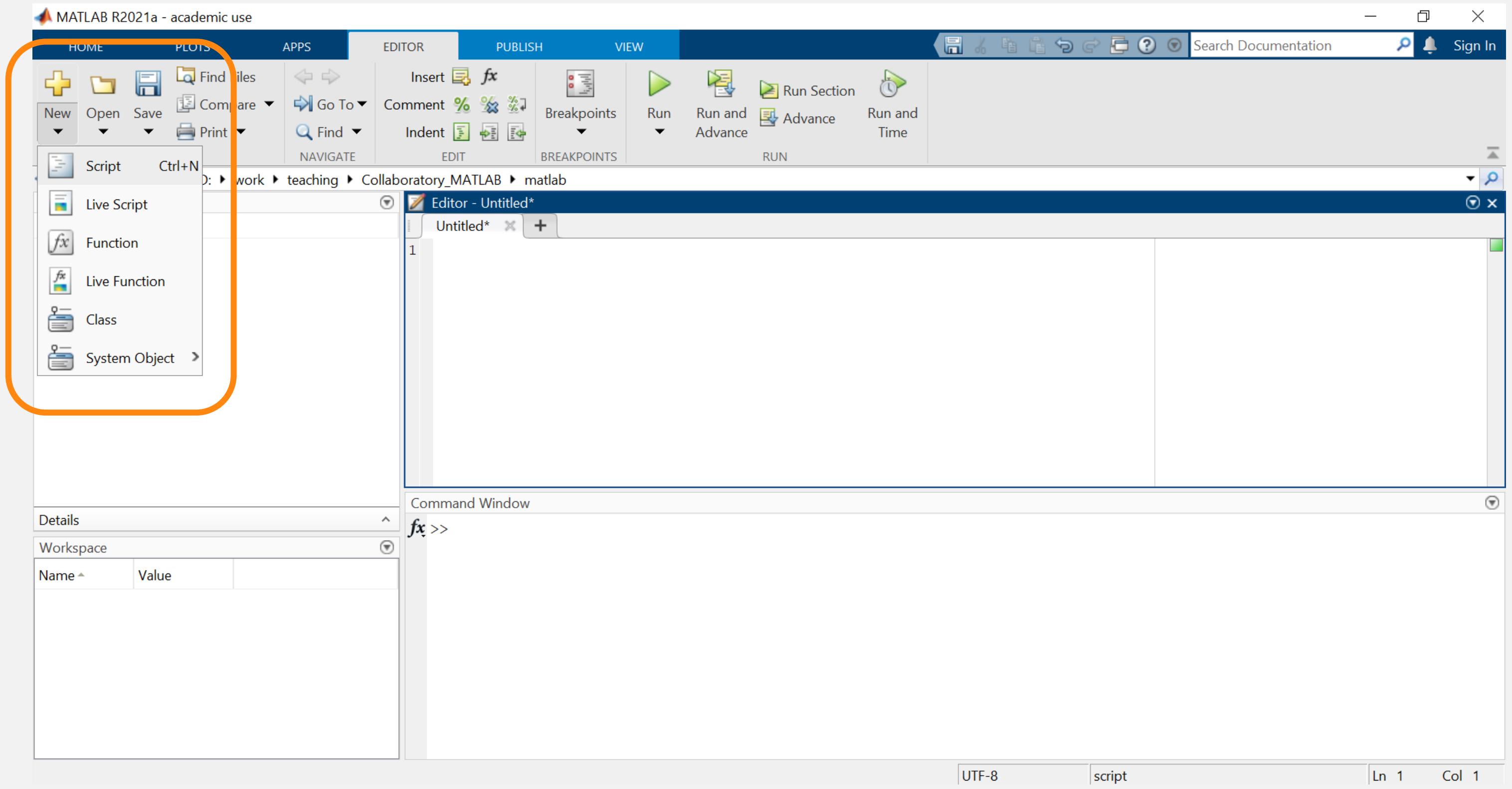
Running code

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

m-files (code)

```
>> edit geneScript
```

mat-files (variables)

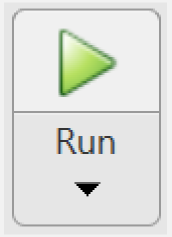


Our first scripts

- In the text editor

```
genesExp1 = 260;
genesExp2 = 58;
genesExp3 = 79;
totalGenes = genesExp1 + genesExp2 + genesExp3;
avgGenes = totalGenes/3;
disp(avgGenes);
```

- Look at the command window



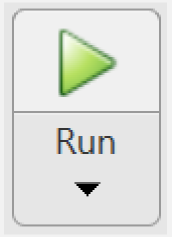
```
>> geneScript
  132.3333
```

Our first scripts

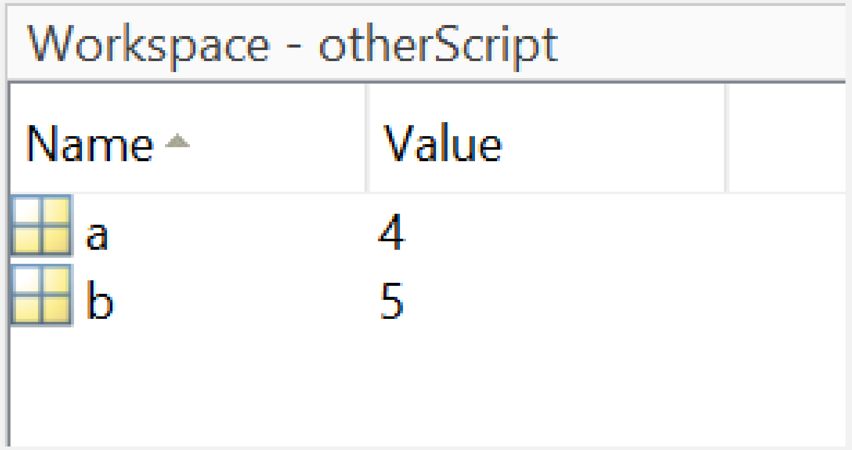
- In the text editor

```
fprintf('hi');
a = 4;
b = 5;
disp (a);
```

- Look at the command window



```
>> otherScript
hi      4
```



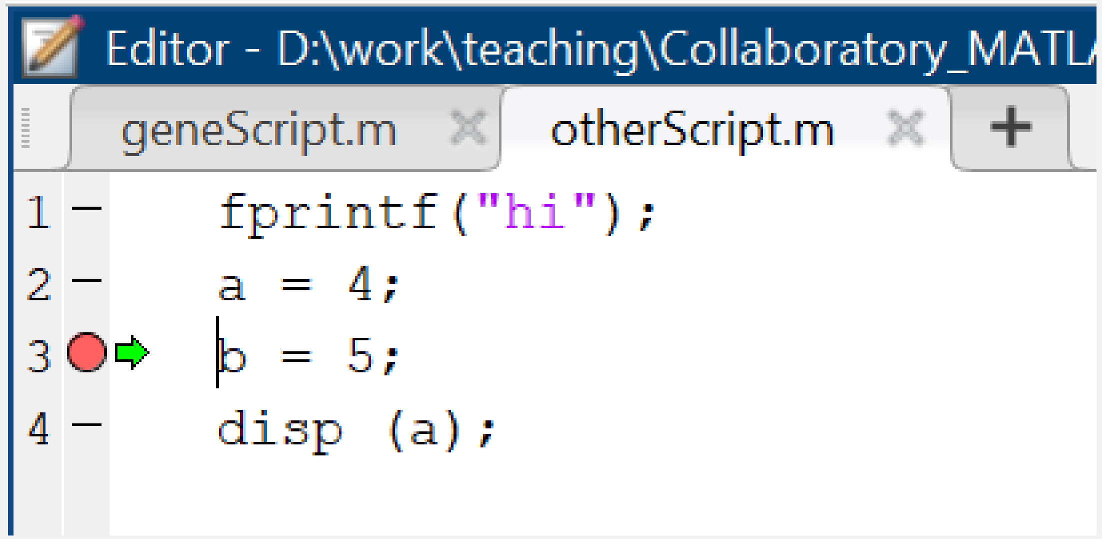
Workspace - otherScript

Name	Value
a	4
b	5

Our first scripts

- Running a script is equivalent to typing all the commands in the command window, but easier to save, edit, debugging, etc…

- Debugging with breakpoints
  Stops at the red dot



Editor - D:\work\teaching\Collaboratory_MATL

```
fprintf("hi");
a = 4;
b = 5;
disp (a);
```

Nb: on my French keyboard I use the " instead of '

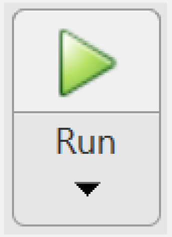
A script to test data type

- In the text editor

```
myName = 'Amandine';
disp(myName);
myNumberOfCats = 0;
doubleVal = double(myNumberOfCats);
charVal = num2str(myNumberOfCats);
```

```
>> class(charVal)

ans =

    'char'

>> class(myNumberOfCats)

ans =

    'double'
```

- Look at the command window



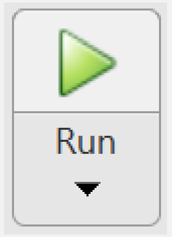
```
>> dataTypes
Amandine
```

A script to test data type

- In the text editor

```
myName = 'Amandine';
disp(myName);
myNumberOfCats = 0;
doubleVal = double(myNumberOfCats);
charVal = num2str(myNumberOfCats);
```

- Look at the command window



```
>> dataTypes
Amandine
```

```
>> ischar(myNumberOfCats)

ans =

  logical

   0

>> ischar(charVal)

ans =

  logical

   1
```

A script to test data type

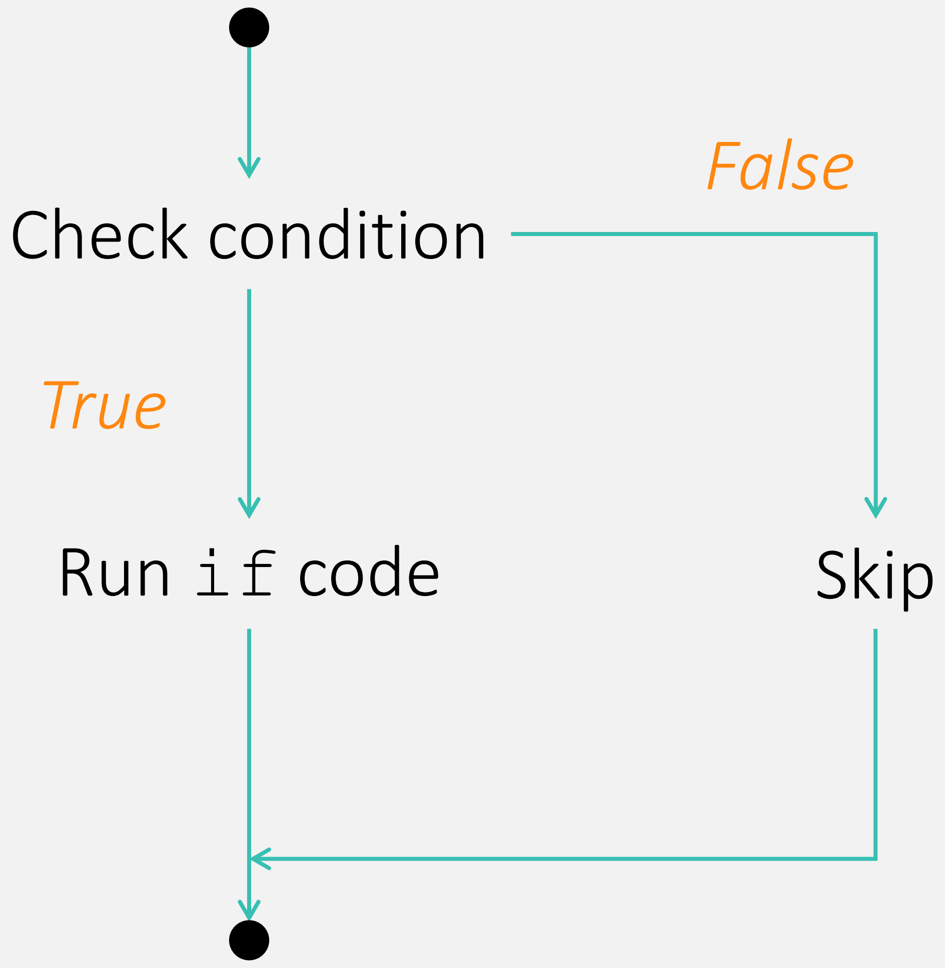
- Test data types
  - isfloat
  - isvector
  - isscalar
  - Ischar

- Other relational operators
  - < Less than
  - <= Less than or equal to
  - > Greater than
  - >= Greater than or equal to
  - == Equal to
  - ~= Not equal to

```
>> 1<1

ans =

  logical

   0

>> 1<2

ans =

  logical

   1
```

`if` statements

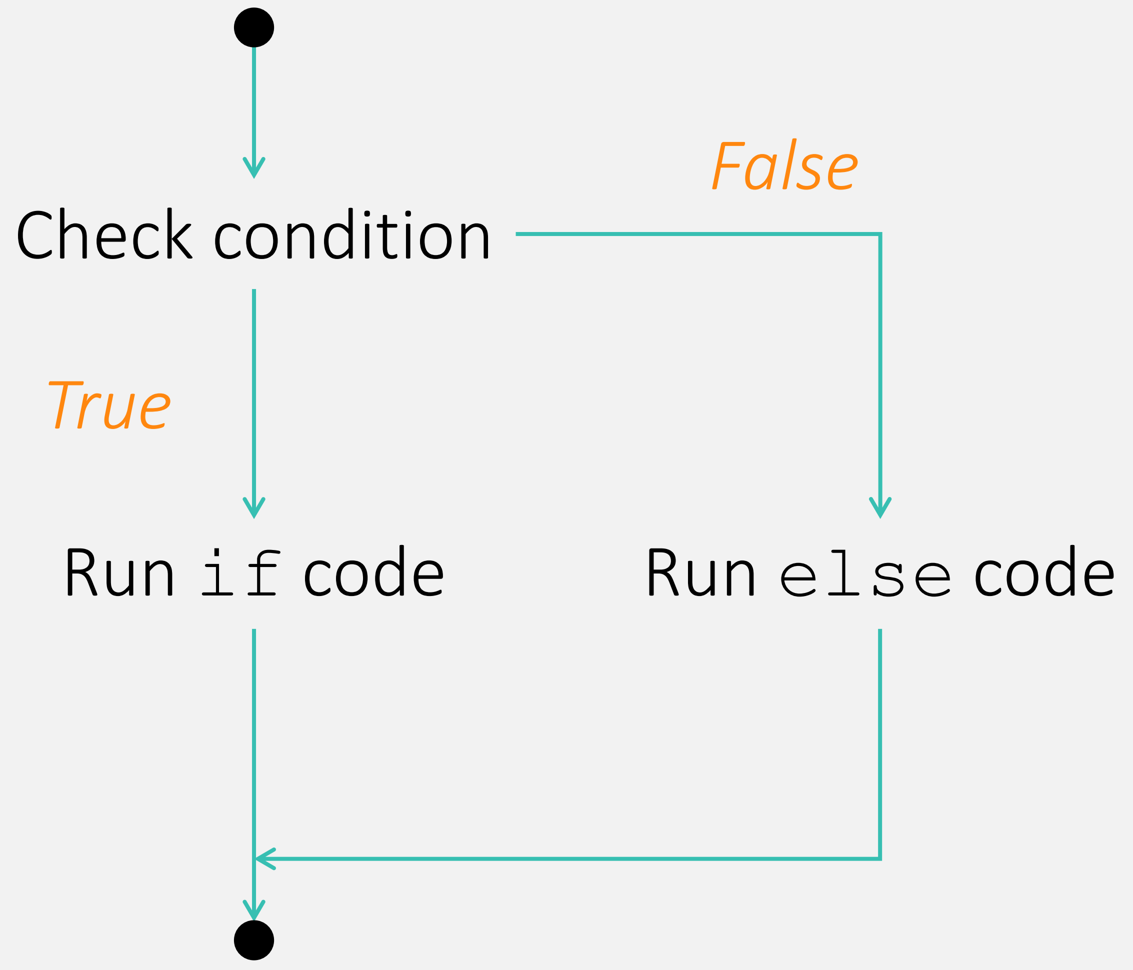
`if` statements



Modify the code to make it print something

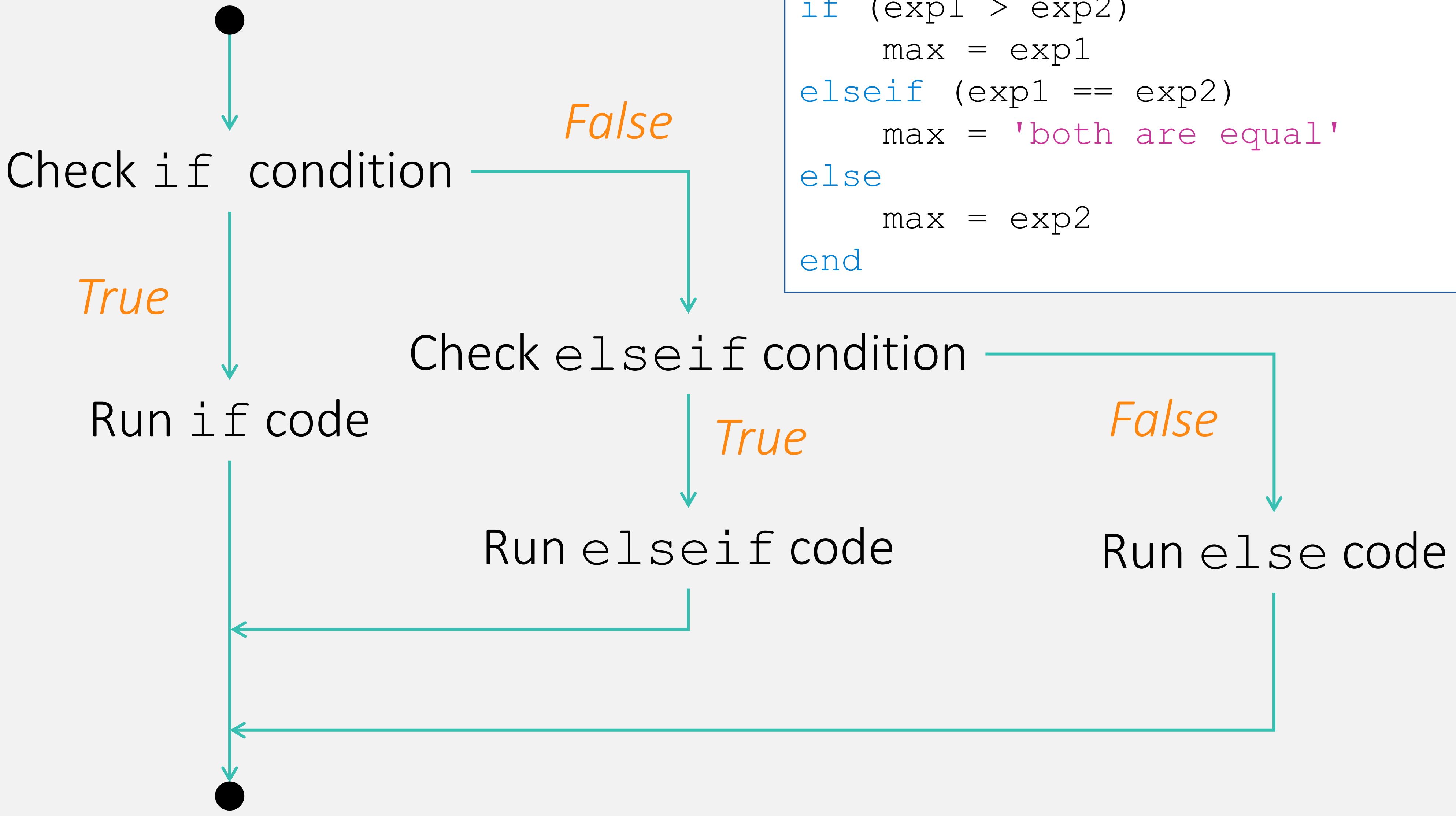
```
exp1 = 400;
exp2 = 500;
if (exp1 < exp2)
    min = exp1
end
```

`if` statements



```
exp1 = 400;
exp2 = 500;
if (exp1 >= exp2)
    max = exp1
else
    max = exp2
end
```

`if` statements



```
exp1 = 400;
exp2 = 500;
if (exp1 > exp2)
    max = exp1
elseif (exp1 == exp2)
    max = 'both are equal'
else
    max = exp2
end
```

Take home, questions?

Outline of the workshop

Day 1

- Interface
- Command lines and basic syntax
- Variables and operations
- Scripts
- `if` statements

Day 2

- **`for` and `while` loops**
- **More matrices**
- **Functions**
- **Files**

Day 3

- Plotting
- Introduction to dynamical systems: ODEs