First steps in MATLAB / Octave

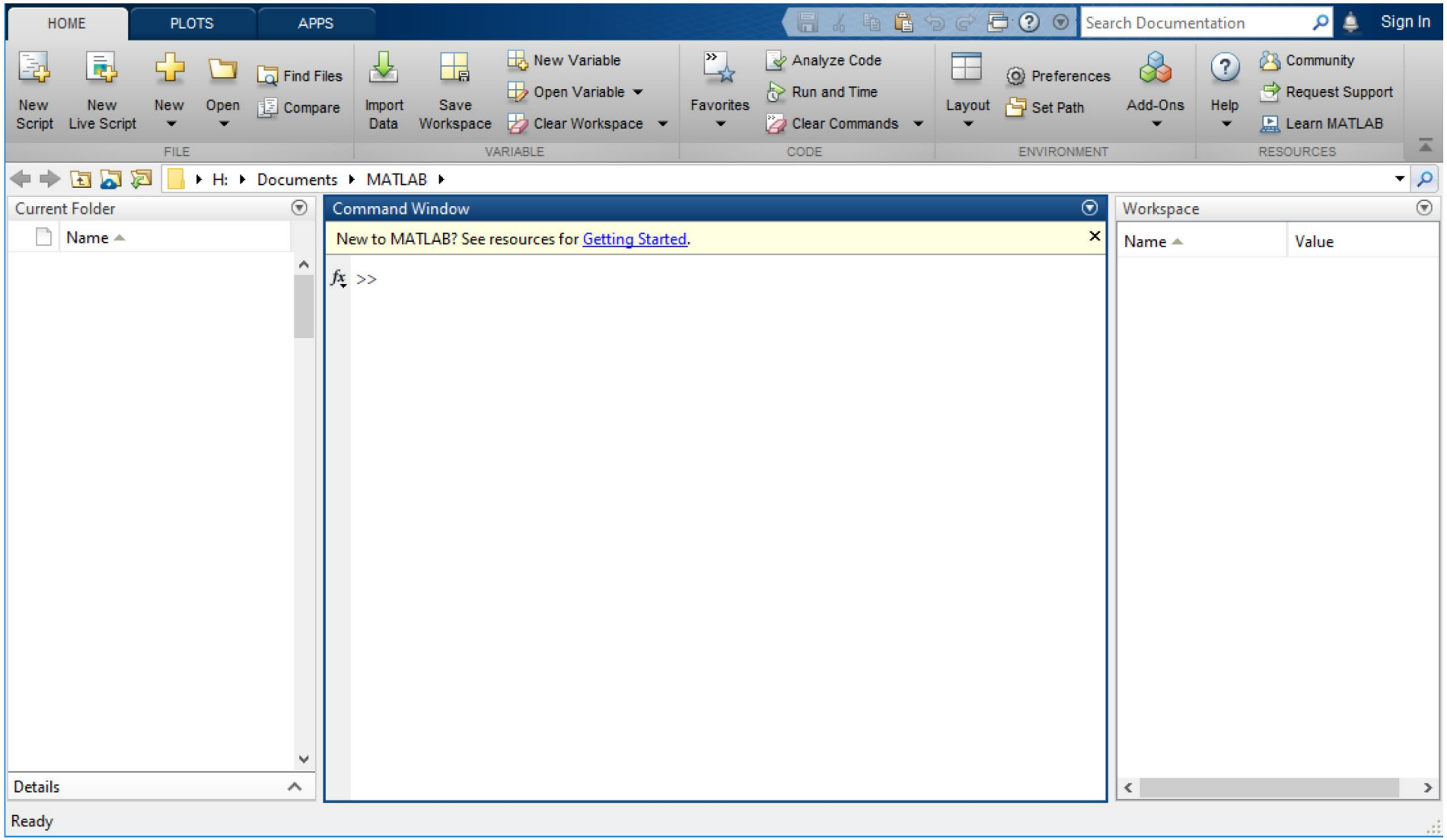
Exercise Sessions

(Course materials for internal use only!)

Computer Vision in Engineering – Prof. Dr. Rodehorst
M.Sc. Mariya Kaisheva
mariya.kaisheva@uni-weimar.de

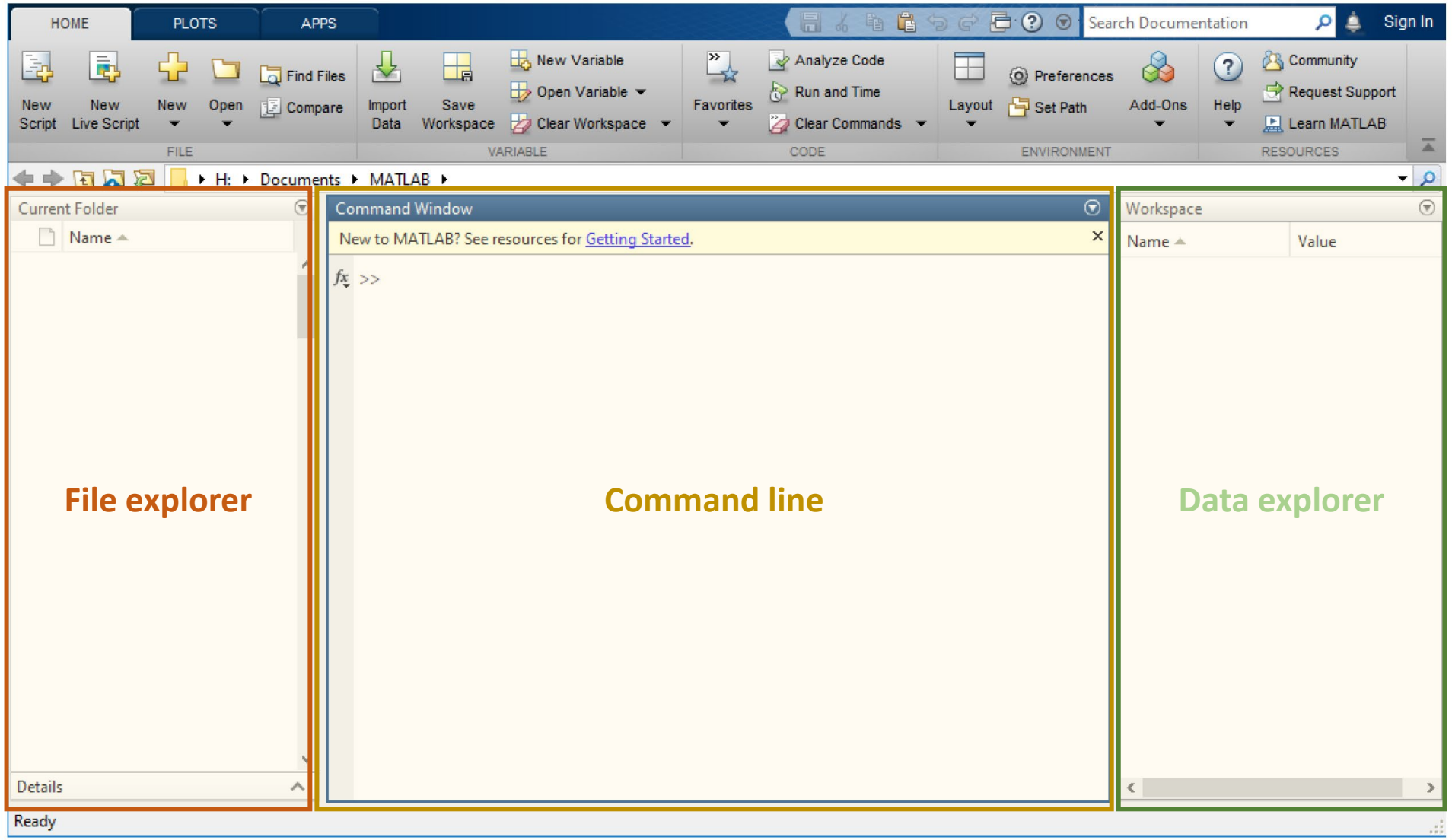
Bauhaus-Universität Weimar

Desktop Basics: MATLAB



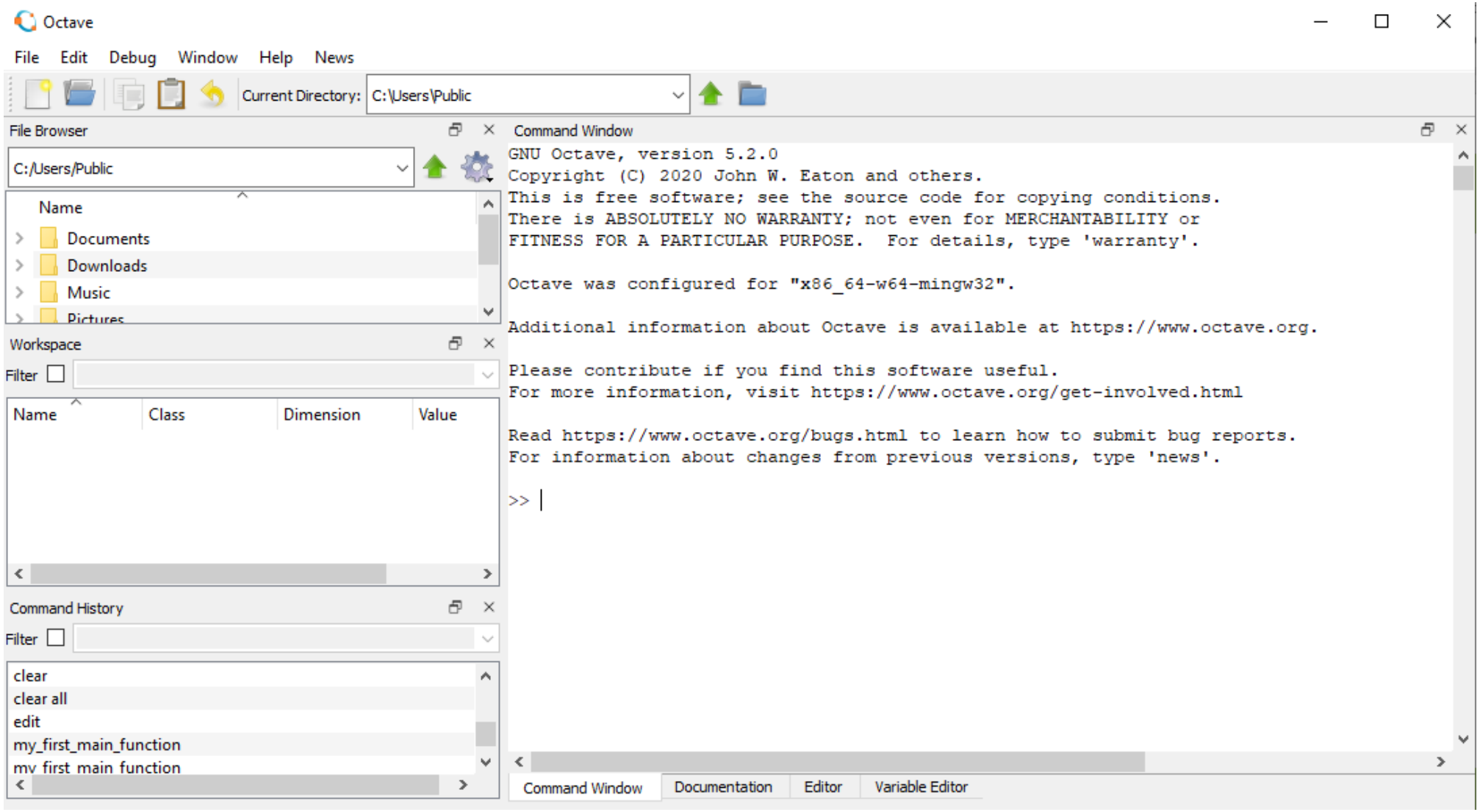
Bauhaus-Universität Weimar

Desktop Basics: MATLAB

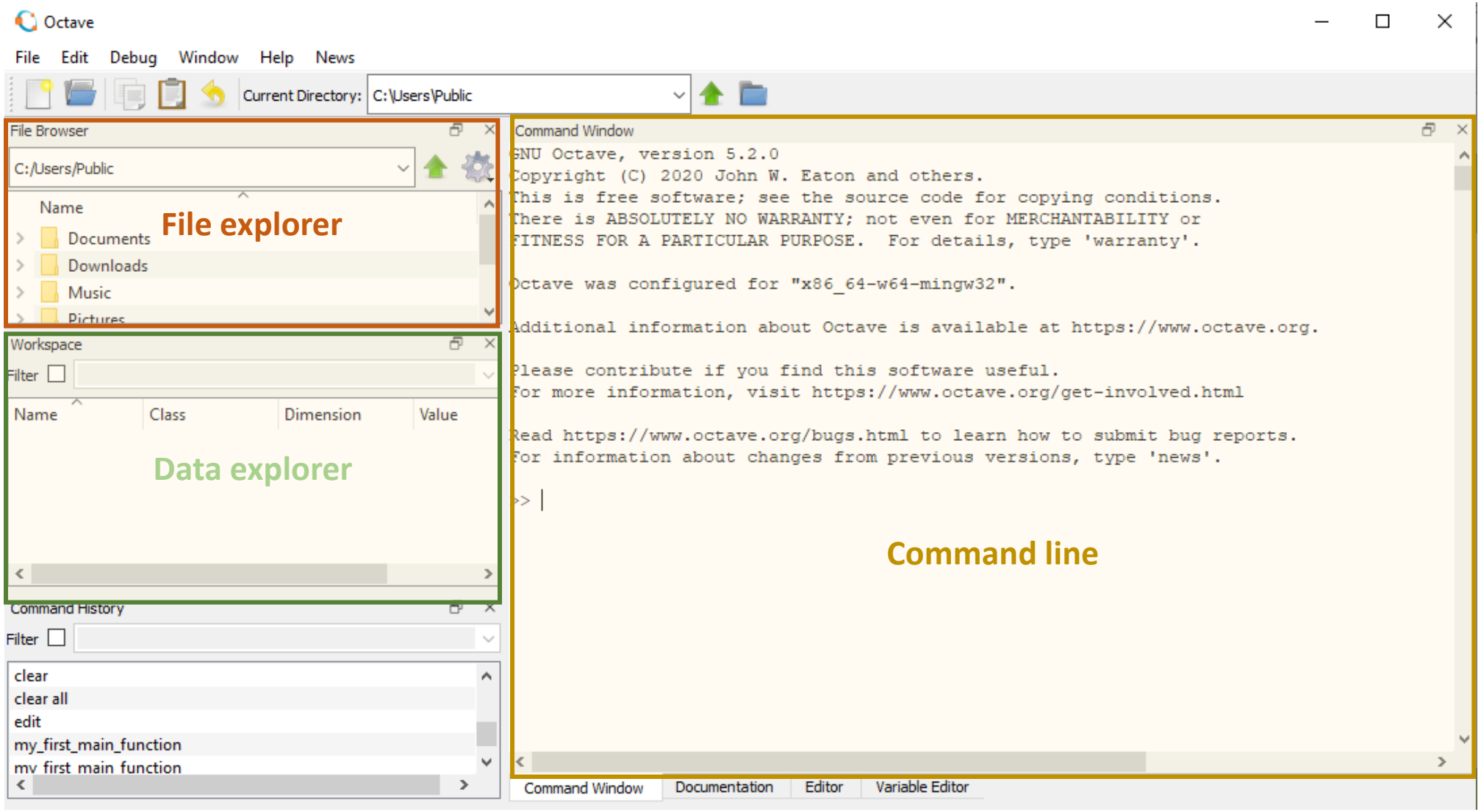


Bauhaus-Universität Weimar

Desktop Basics: Octave



Desktop Basics: Octave



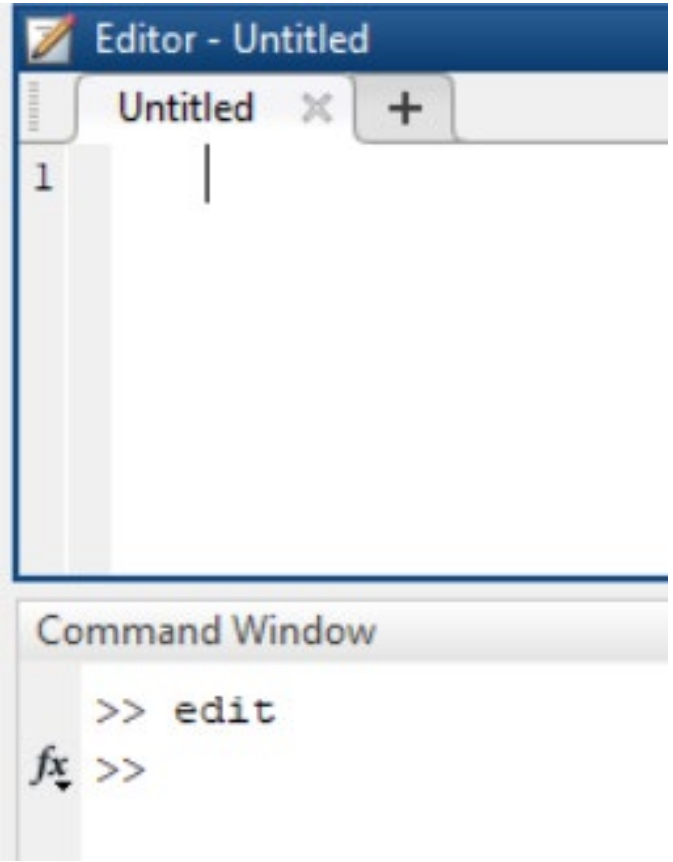
Bauhaus-Universität Weimar

Function Definitions

- New functions are defined in ***.m** files
- The **filename** has to correspond to the **function name**

  → e.g. the function `myFunc` is defined in `myFunc.m`
- New *.m files can be created using command `edit` in the Command Window or the menu item **Home → New Script** (old: File → New → M-File)
- General syntax:

```
function functionName
    . . .
end
```

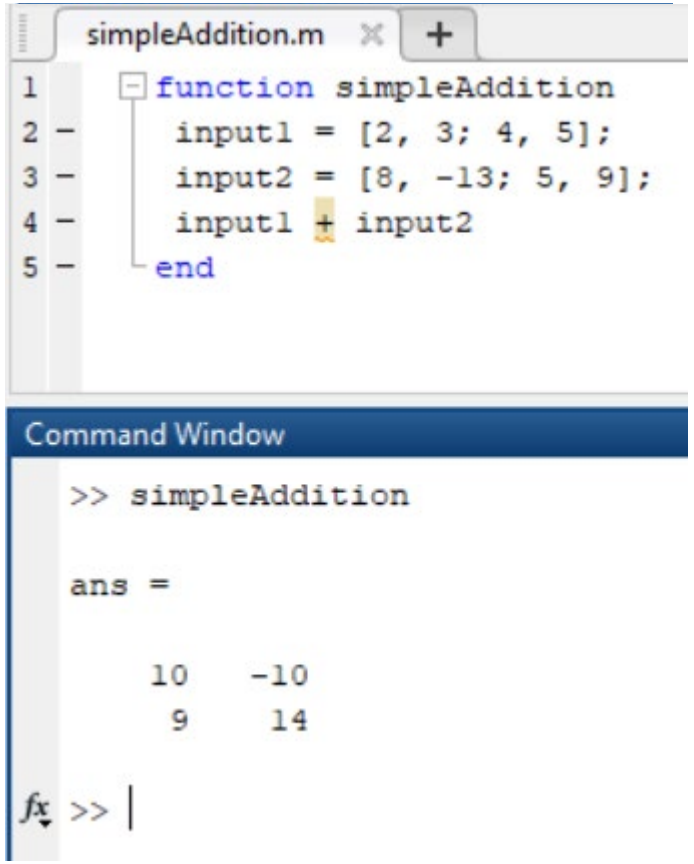


Function Definitions

- New functions are defined in ***.m** files
- The **filename** has to correspond to the **function name**

  → e.g. the function `myFunc` is defined in `myFunc.m`
- New *.m files can be created using command `edit` in the Command Window or the menu item **Home → New Script** (old: File → New → M-File)
- General syntax:

```
function functionName
    . . .
end
```



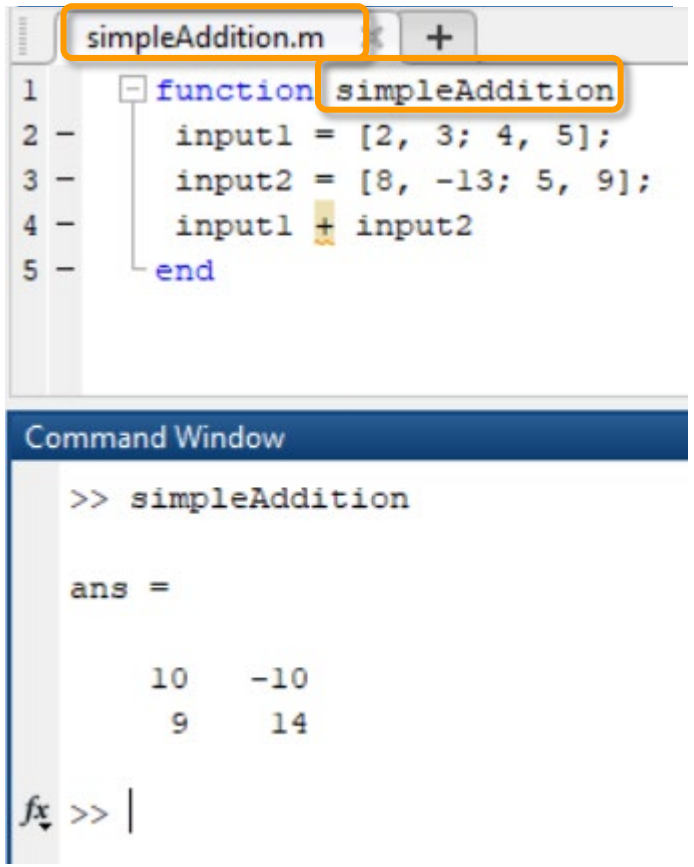
Function Definitions

- New functions are defined in ***.m** files
- The **filename** has to correspond to the **function name**

  → e.g. the function `myFunc` is defined in `myFunc.m`

- New *.m files can be created using command `edit` in the Command Window or the menu item **Home → New Script** (old: File → New → M-File)
- General syntax:

```
function functionName
    . . .
end
```



Function Definitions

- Several **sub-functions** can be defined in a single *.m file, however only the **main function** is visible to the outside
- **Important:** MATLAB will only find your *.m file if the directory is
  - set as the **"Current Directory"** in the main window, or
  - permanently added to the search path using

    Home → Set Path (old: File → Set Path...)

my_first_main_function.m

```
% my first main function
function my_first_main_function

    input1 = 23
    input2 = 14

    [s,m] = multANDsum(input1, input2);

end

% this awesome function can do multiplication and summation in one step
function [summation, multiplication] = multANDsum(input1, input2)


    summation = input1 + input2

    multiplication = input1 * input2
    multiplication = my_multiplication(input1, input2)

end

%-----------------------------------------------------------------------
function [output] = my_multiplication(input1, input2)
    output = input1 * input2;
end
```

Matrix Notations

- **Definition of vectors/matrices**
  - element are contained in **[ ]**
  - row elements are separated by **commas** (or blanks)
  - columns are separated by **semicolons**

```
>> a = [23, 12]

a =

    23    12
```

```
>> a = [23 12]

a =

    23    12
```

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

a =

     1     2     3
     4     5     6
```

Matrix Notations

- **Definition of vectors/matrices**
  - element are contained in **[ ]**
  - row elements are separated by **commas** (or blanks)
  - columns are separated by **semicolons**
  - **ones(**numRows, numColumns**)** initializes a matrix of ones
  - **zeros(**numRows, numColumns) initializes a matrix of zeros
  - **eye**(numRows, numColumns) returns an identity matrix
  - `a:b` initializes a vector [*a*, *a+1*, *a+2*, ... , *b*]
  - `a:c:b` initializes a vector [*a*, *a+c*, *a+2*c*, ... , *b*]

```
>> zeros(2,3)

ans =

   0   0   0
   0   0   0
```

```
>> ones(2,3)

ans =

   1   1   1
   1   1   1
```

```
>> 1:5

ans =

   1   2   3   4   5
```

```
>> 20:-2:14

ans =

   20   18   16   14
```

Basic Operations

- Matrix multiplication *****
- Multiplication of corresponding matrix elements **.***
- Right-matrix division (solution for Ax = B) **/**
- Element-wise division **./**
- Transposition **'**

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

a =

    1    2
    3    4
```

```
>> a'

ans =

    1    3
    2    4
```

```
>> a = [0,1;1,0]; b = [1;2];
>> a*b

ans =

    2
    1

>> b*a
Error using  *
Inner matrix dimensions must agree.
```

Index Expressions

- Indices start at **1** (not at 0 like in other languages)
- Index expressions can be used to **reference** or **extract selected elements** of a matrix or vector
- Index expressions can contain
  - **scalars**,
  - **vectors**,
  - **ranges**,
  - or the **special operator :**

<u>Example index expressions for matrix **a**</u>

a(1, 2)	# row 1, column 2
a(1, [1, 3])	# row 1, columns 1 and 3
a(1, 1:5)	# row 1, columns in range 1-5
a(1, :)	# row 1, all columns

Index Expressions: Examples

```
>> a = zeros(5, 3)

a =

     0     0     0
     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

row column

```
>> a(2,4)= 45.7

a =

         0         0         0         0
         0         0         0   45.7000
         0         0         0         0
         0         0         0         0
         0         0         0         0
```

all rows list of columns

```
>> a(:, [1, 3]) = 2.5

a =

    2.5000         0    2.5000         0
    2.5000         0    2.5000   45.7000
    2.5000         0    2.5000         0
    2.5000         0    2.5000         0
    2.5000         0    2.5000         0
```

Relational and Logical Operators

Relational operators

<	**less than**
>	**greater than**
<=	**less than or equal**
>=	**greater than or equal**
==	**equal**
~=	**not equal**

Logical operators

&	**and**
|	**or**
~	**not**

```
>> a = zeros(2)

a =

     0     0
     0     0

>> b = ones(2)

b =

     1     1
     1     1

>> if a~=b
t = 'different elements';
else
t = 'same elements';
end
>> t

t =

different elements
```

Relational and Logical Operators

Relational operators

<	less than
>	greater than
<=	less than or equal
>=	greater than or equal
==	equal
~=	not equal

Logical operators

&	and
|	or
~	not

```
>> a = zeros(2)

a =

     0     0
     0     0

>> b = ones(2)

b =

     1     1
     1     1

>> if a~=b
t = 'different elements';
else
t = 'same elements';
end
>> t

t =

different elements
```

For-loop

- General syntax

```
for index = values
    … % the operations you want to repeat at every iteration
end
```

- Simple examples

```
for index = 1:100
    a = index * 2;
    disp(a)
end
```

```
for index = 1:2:100
    a = index * 2;
    disp(a)
end
```

```
for index = [5 27 16 43]
    a = index * 2;
    disp(a)
end
```

Vectorization

- Uses whole-array operations
- Often leads to significant speed-up compared to loop-based implementations
- Allows more compact implementations

```
for index = 1:length(a)
  if (a(index) > 5)
    a(index) -= 20
  end
end
```

loop-based version

```
a(a>5) -= 20;
```

vectorized version

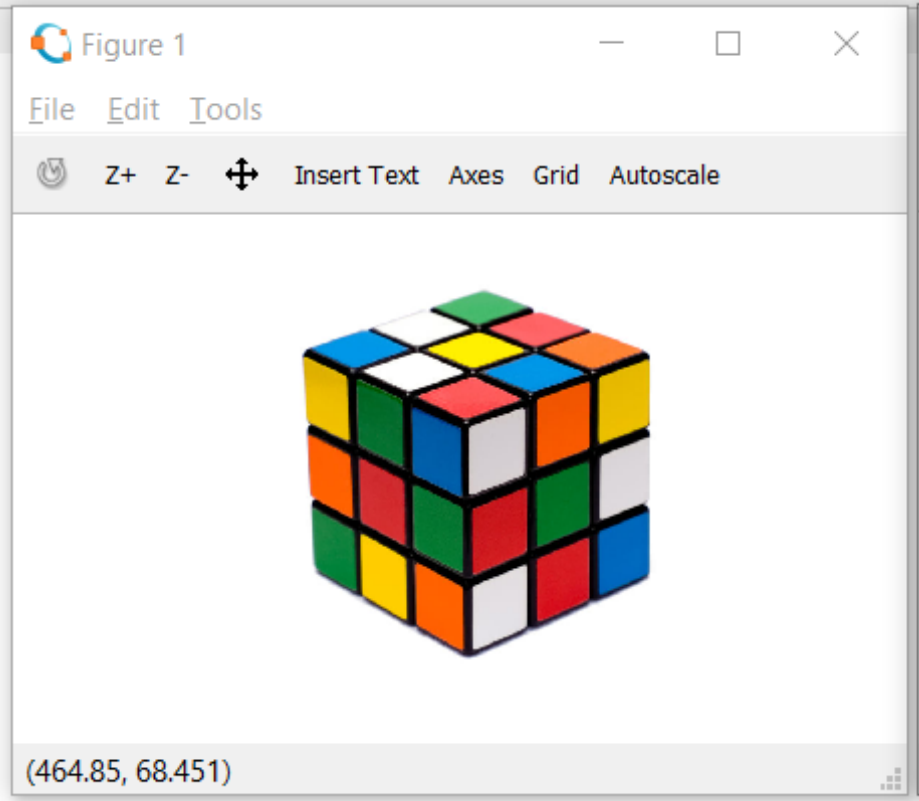
Reference documentation:

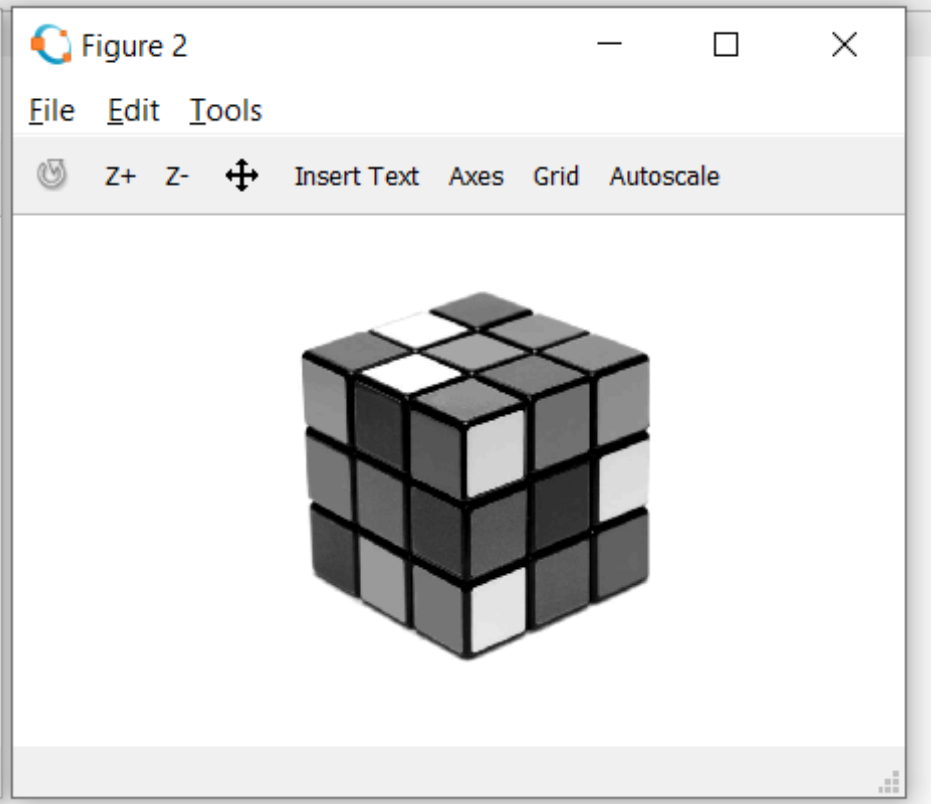
- MATLAB: https://nl.mathworks.com/help/matlab/matlab_prog/vectorization.html
- Octave: https://octave.org/doc/v4.0.1/Basic-Vectorization.html

Operations on Images Matrices

Command Window

```
>> I = imread('input.jpg');
>> size(I)
ans =

   600   600     3

>> figure(1), imshow(I);
>> Imean = uint8(mean(I, 3));
>> figure(2), imshow(Imean);
>> size(Imean)
ans =

   600   600

>>
```



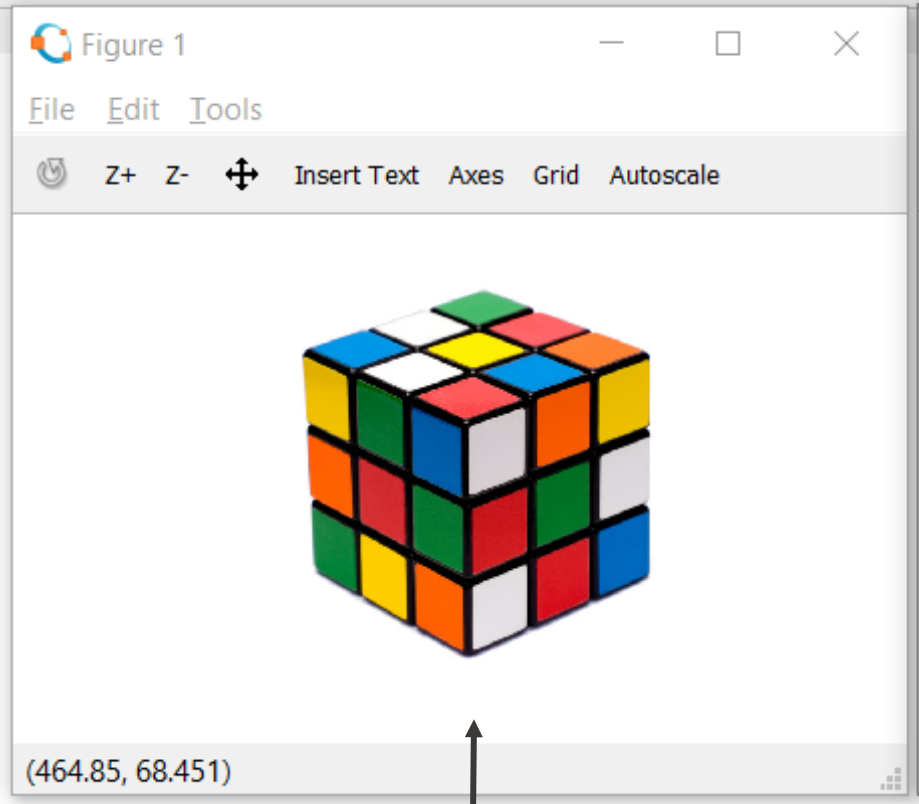


Operations on Images Matrices

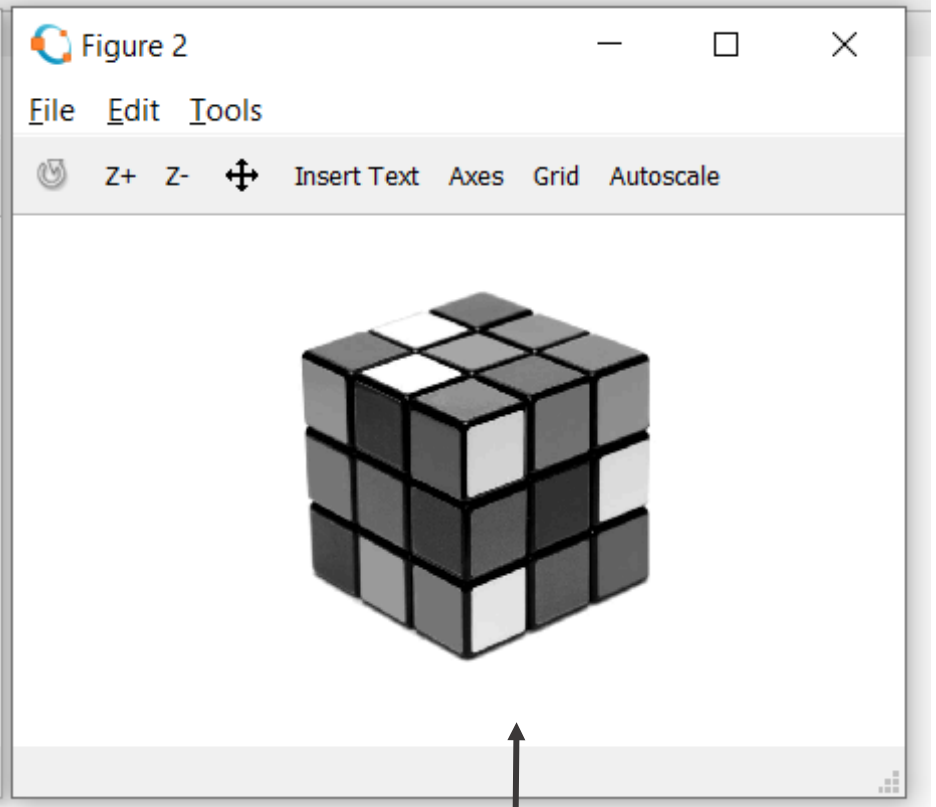
Command Window

```
>> I = imread('input.jpg');
>> size(I)
ans =

   600   600     3

>> figure(1), imshow(I);
>> Imean = uint8(mean(I, 3));
>> figure(2), imshow(Imean);
>> size(Imean)
ans =

   600   600

>> |
```

When and why is it important to add **;** at the end of a line?



3 channels



1 channel

Help and Documentation

- `doc` - open the function documentation in a separate window
- `help` - view a text version of the function documentation in the Command Window
- `lookfor` - search for keyword in all help entries

- *function_name*(- display function hints by pausing after you type in the Command Window the open parentheses for the function input arguments

MATLAB-specific functionality

Online Documentation

- **MATLAB**
  - https://de.mathworks.com/help/matlab/index.html
  - https://www2.cs.duke.edu/courses/fall02/cps150/matlab/primer.pdf
  - https://www.tutorialspoint.com/matlab/matlab_overview.htm

- **GNU Octave**
  - https://octave.org/doc/interpreter/
  - https://octave.org/doc/interpreter/Simple-Examples.html#Simple-Examples
  - http://ais.informatik.uni-freiburg.de/teaching/ws11/robotics2/pdfs/rob2-03-octave.pdf
  - http://www-h.eng.cam.ac.uk/help/programs/octave/tutorial/