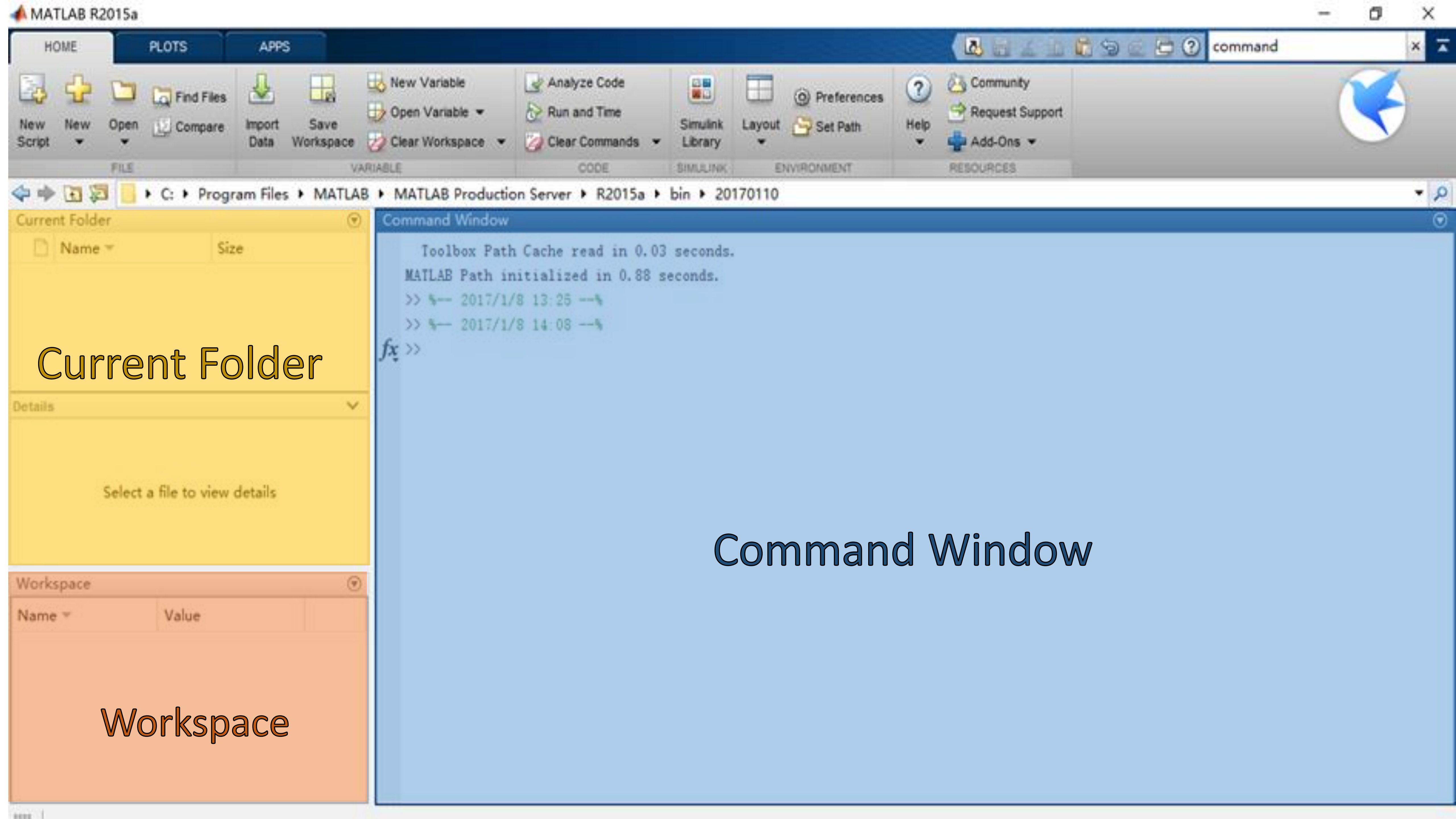
Introduction to MATLAB

Lecture 1: Quick Start

Outline

- Desktop Basics
- Matrices and Arrays
- Array Indexing
- Workspace Variables
- 2-D and 3-D Plots
- Programming and Scripts
- Help and Documentation



```
  Toolbox Path Cache read in 0.03 seconds.
MATLAB Path initialized in 0.88 seconds.
>> %-- 2017/1/8 13:25 --%
>> %-- 2017/1/8 14:08 --%
fx >>
```

Desktop Basics

- The desktop includes these panels:
- Current Folder — Access your files.
- Command Window — Enter commands at the command line, indicated by the prompt (>>).
- Workspace — Explore data that you create or import from files.
- Command History — Presents a log of the statements most recently run in the Command Window.

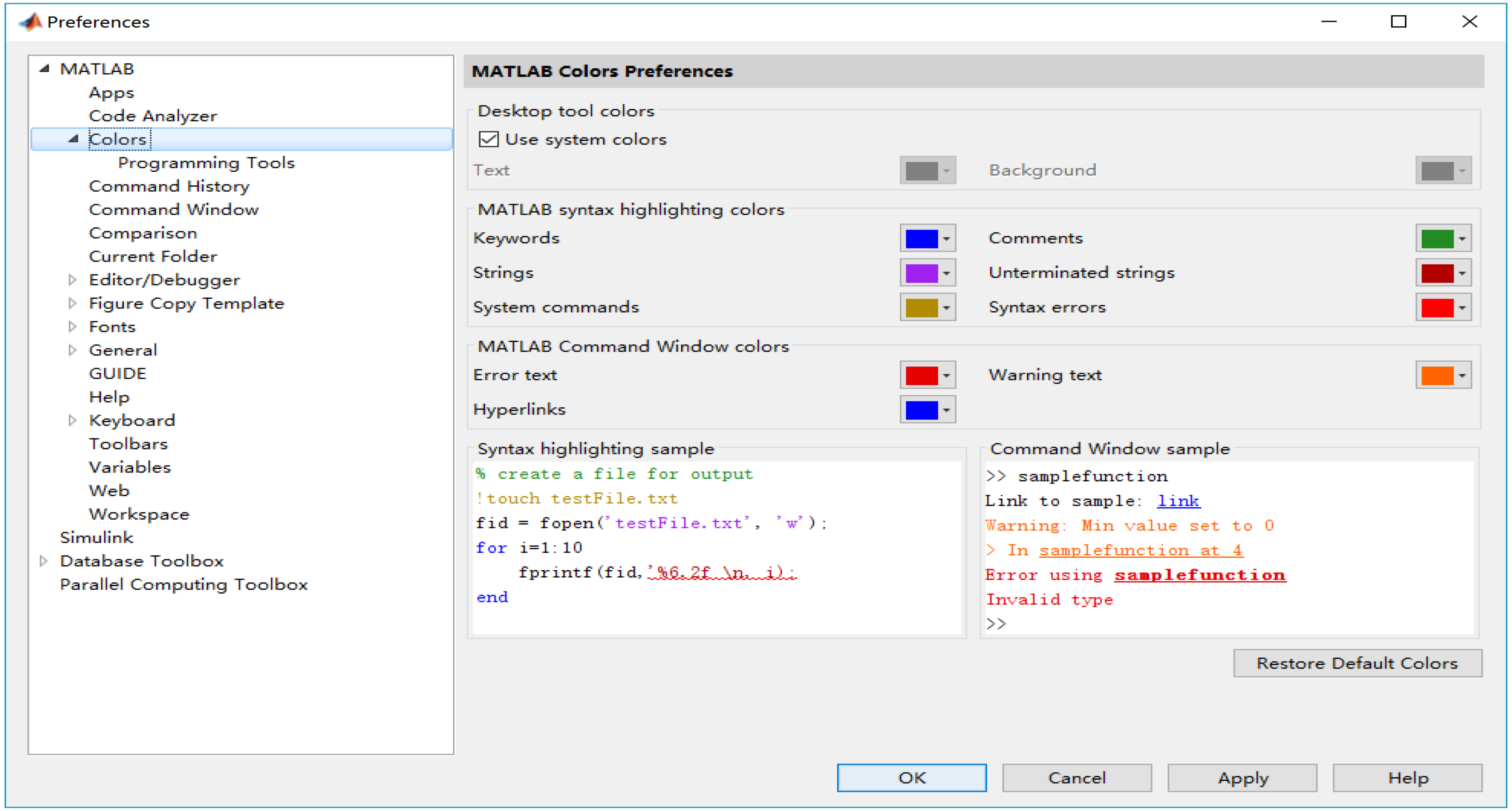
Making Folders

You can use folders to keep your programs organized.

- To make a new folder, click the ‘Browse’ button next to ‘Current Directory’
- Click the ‘New Folder’ button, and name the folder. **Do NOT use spaces** in folder names.
- The current directory is now the folder you just created

Customize

You can also customize your MATLAB by setting Preference.



Preferences

- MATLAB
  - Apps
  - Code Analyzer
  - Colors
    - Programming Tools
  - Command History
  - Command Window
  - Comparison
  - Current Folder
  - Editor/Debugger
  - Figure Copy Template
  - Fonts
  - General
  - GUIDE
  - Help
  - Keyboard
  - Toolbars
  - Variables
  - Web
  - Workspace
- Simulink
- Database Toolbox
- Parallel Computing Toolbox

MATLAB Colors Preferences

Desktop tool colors

☑ Use system colors

Text	⬛	Background	⬛

MATLAB syntax highlighting colors

Keywords	🟦	Comments	🟩
Strings	🟪	Unterminated strings	🟥
System commands	🟨	Syntax errors	🟥

MATLAB Command Window colors

Error text	🟥	Warning text	🟧
Hyperlinks	🟦		

Syntax highlighting sample

```
% create a file for output
!touch testFile.txt
fid = fopen('testFile.txt', 'w');
for i=1:10
    fprintf(fid,'%6.2f \n', i);
end
```

Command Window sample

```
>> samplefunction
Link to sample: link
Warning: Min value set to 0
> In samplefunction at 4
Error using samplefunction
Invalid type
>>
```

Creating Variables

- As you work in MATLAB, you issue commands that create variables and call functions. For example, create a variable named a by typing this statement at the command line:

```
>> a=1
```

- MATLAB adds variable a to the workspace and displays the result in the Command Window.

```
a =

     1
```

Creating Variables

- When you do not specify an output variable, MATLAB uses the variable ans, short for *answer*, to store the results of your calculation.

- If you **end a statement with a semicolon**, MATLAB performs the computation, but suppresses the display of output in the Command Window.

- You can recall previous commands by pressing the up- and down-arrow keys, ↑ and ↓. Press the arrow keys either at an empty command line or after you type the first few characters of a command. For example, to recall the command b = 2, type b, and then press the up-arrow key.

Naming Variables

Variable names

- first character must be a LETTER
- after that, any combination of letters, numbers and _
- CASE SENSITIVE! (var1 is different from Var1)

Built-in variables. Don’t use these names!

- i and j can be used to indicate complex numbers
- pi has the value 3.1415926…(Overwrite)
- ans stores the last unassigned value (like on a calculator)
- Inf and –Inf are positive and negative infinity
- NaN represents ‘Not a Number’

Outline

- Desktop Basics
- Matrices and Arrays
- Array Indexing
- Workspace Variables
- Text and Characters
- Programming and Scripts
- Help and Documentation

Array Creation

- To create an array with four elements in a single row, separate the elements with either a comma (,) or a space.

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

  This type of array is a row vector.(?Column vecs)

- To create a matrix that has multiple rows, separate the rows with semicolons.

  ```
  >a = [1 2 3; 4 5 6; 7 8 9]
  ```

- Another way to create a matrix is to use a function, such as ones, zeros, or rand. For example, create a 4-by-4 column vector of zeros.

  ```
  >z = zeros(4,4) ,?ones ?magic
  ```

Matrix and Array Operations

- MATLAB allows you to process all of the values in a matrix using a single arithmetic operator or function.

```
>>a + 10
```

- To transpose a matrix, use a single quote ('):

```
>>a‘
```

- You can perform standard matrix multiplication, which computes the inner products between rows and columns, using the * operator. For example, confirm that a matrix times its inverse returns the identity matrix:

```
>>p = a*inv(a)
```

- To perform element-wise multiplication rather than matrix multiplication, use the .* operator:

```
>>p = a.*a
```

- The matrix operators for multiplication, division, and power each have a corresponding array operator that operates element-wise.

```
>>a.^3
```

Concatenation

- *Concatenation* is the process of joining arrays to make larger ones. In fact, you made your first array by concatenating its individual elements. The pair of square brackets [] is the concatenation operator.

```
>>A = [a,a]
```

- Concatenating arrays next to one another using commas is called horizontal concatenation. Each array must have the same number of rows. Similarly, when the arrays have the same number of columns, you can concatenate vertically using semicolons.

```
>>A = [a; a]
```

Complex Numbers

- Complex numbers have both real and imaginary parts, where the imaginary unit is the square root of -1.

```
>>sqrt(-1)
```

- To represent the imaginary part of complex numbers, use either i or j .

```
>>c= [3+4i, 4+3j; -i, 10j]
```

Outline

Array Indexing

- Every variable in MATLAB® is an array that can hold many numbers. When you want to access selected elements of an array, use indexing.

  For example, consider the 4-by-4 random square A:

  ```
  >> A = rand(4,4)
  ```

- There are two ways to refer to a particular element in an array. The most common way is to specify row and column subscripts, such as

  ```
  >>A(4,2)
  ```

- Less common, but sometimes useful, is to use a single subscript that traverses down each column in order:

  ```
  >>A(8)
  ```

- Using a single subscript to refer to a particular element in an array is called *linear indexing*.

Array Indexing

- If you try to refer to elements outside an array, MATLAB throws an error.

```
>>test = A(4,5)
Index exceeds matrix dimensions.
```

- However, you can specify elements outside the current dimensions. The size of the array increases to accommodate the newcomers.

```
>>A(4,5)=44
```

Array Indexing

- To refer to multiple elements of an array, use the colon operator, which allows you to specify a range of the form start:end. For example, list the elements in the first three rows and the second column of A:

  ```
  >>A(1:3,2)
  ```

- The colon alone, without start or end values, specifies all of the elements in that dimension. For example, select all the columns in the third row of A:

  ```
  >>A(3,:)
  ```

- The colon operator also allows you to create an equally spaced vector of values using the more general form start:step:end.

  ```
  >>B = 0:0.01pi:2pi
  ```

Outline

- Desktop Basics
- Matrices and Arrays
- Array Indexing
- Workspace Variables
- 2-D and 3-D Plots
- Programming and Scripts
- Help and Documentation

Workspace Variables

- The workspace contains variables that you create within or import into MATLAB from data files or other programs. For example, these statements create variables A and B in the workspace.

```
>>A = ones(4,4);
>>B = ones(6,6);
```

- You can view the contents of the workspace using whos.

```
>>whos
```

- The variables also appear in the Workspace pane on the desktop.

Workspace Variables

- Workspace variables do not persist after you exit MATLAB. Save your data for later use with the save command,

```
>>save myfile.mat
```

- Saving preserves the workspace in your current working folder in a compressed file with a .mat extension, called a MAT-file.
- To clear variables from the workspace, use the clear command.

```
>>clear a b
>>clear all
```

- Restore data from a MAT-file into the workspace using load.

```
>>load myfile.mat
```

Outline

- Desktop Basics
- Matrices and Arrays
- Array Indexing
- Workspace Variables
- 2-D and 3-D Plots
- Programming and Scripts
- Help and Documentation

Line Plots

- To create two-dimensional line plots, use the plot function. For example, plot the value of the sine function from 0 to 2:

```
>>x = 0:pi/100:2*pi; y = sin(x);
>>plot(x,y)
```

- You can label the axes and add a title.

```
>>xlabel('x');
>>ylabel('sin(x)');
>>title('Plot of the Sine Function')
```

- The 'r--' string is a line specification. Each specification can include characters for the line color, style, and marker. A marker is a symbol that appears at each plotted data point, such as a +, o, or *. For example, 'g:*' requests a dotted green line with * markers.
- Notice that the titles and labels that you defined for the first plot are no longer in the current figure window. By default, MATLAB® clears the figure each time you call a plotting function, resetting the axes and other elements to prepare the new plot.

Line Plots

- To add plots to an existing figure, use hold.

```
>>x = 0:pi/100:2*pi;
>>y = sin(x);
>>plot(x,y)
>>hold on
>>y2 = cos(x); plot(x,y2,':')
>>legend('sin','cos')
```

- Until you use hold off or close the window, all plots appear in the current figure window.

3D Plots

- Three-dimensional plots typically display a surface defined by a function in two variables, $z = f(x,y)$.
- To evaluate z, first create a set of (x,y) points over the domain of the function using meshgrid.

```
>>[X,Y] = meshgrid(-2:.2:2);
>>Z = X .* exp(-X.^2 - Y.^2);
```

- Then, create a surface plot.

```
>>surf(X,Y,Z)
```

Both the surf function and its companion mesh display surfaces in three dimensions. surf displays both the connecting lines and the faces of the surface in color. mesh produces wireframe surfaces that color only the lines connecting the defining points.

Outline

- Desktop Basics
- Matrices and Arrays
- Array Indexing
- Workspace Variables
- 2-D and 3-D Plots
- Programming and Scripts
- Help and Documentation

Scripts Overview

- Scripts are :

  collection of commands executed in sequence

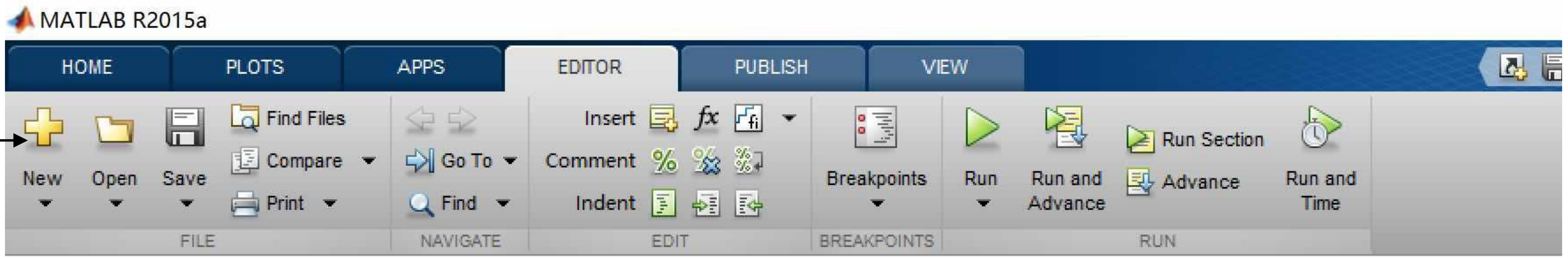
  written in the MATLAB editor

  saved as MATLAB files (.m extension)

- To create an MATLAB file from command-line

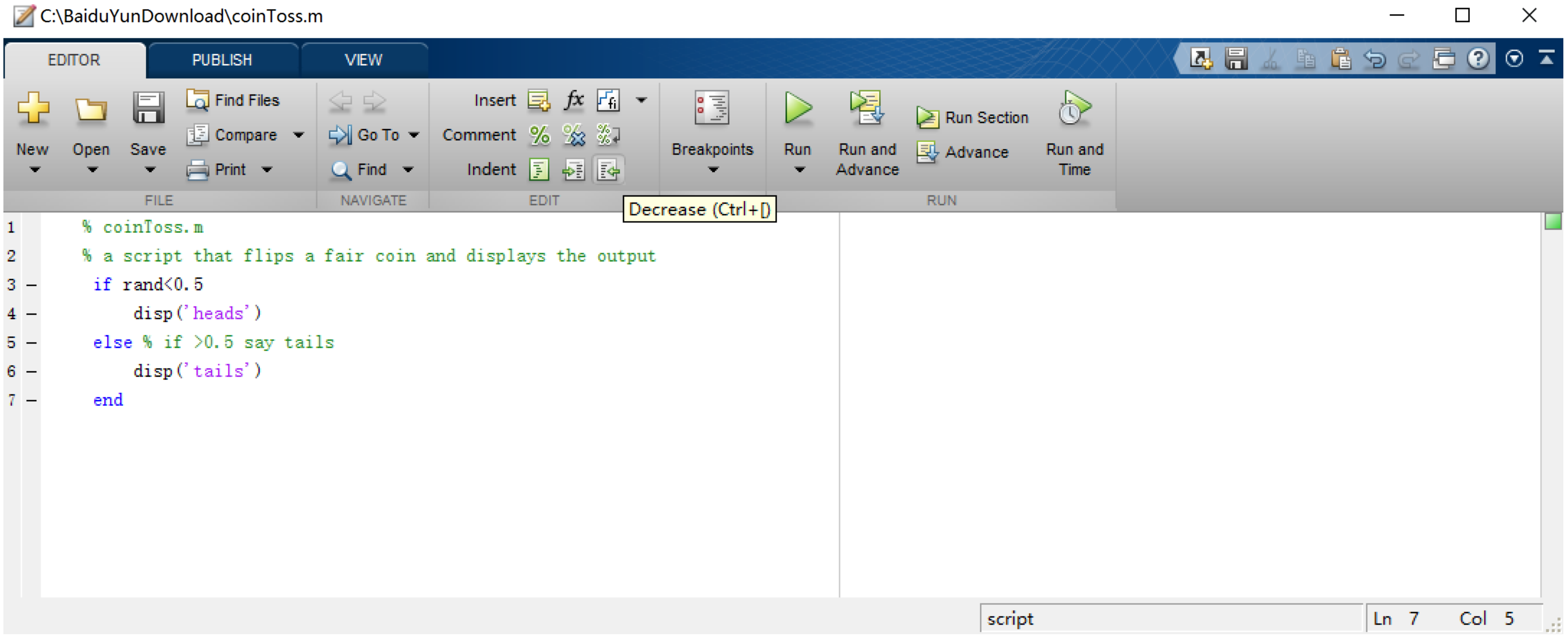
  >>edit crossToss.m

  or click



  or press ctrl + N

Scripts: the Editor



```matlab
    % coinToss.m
    % a script that flips a fair coin and displays the output
     if rand<0.5
         disp('heads')
     else % if >0.5 say tails
         disp('tails')
     end
```

Scripts: Comments

- COMMENT!
  - Anything following a % is seen as a comment
  - The first contiguous comment becomes the script's help file
  - Comment thoroughly to avoid wasting time later
- Note that scripts are somewhat static, since there is no input and no explicit output
- All variables created and modified in a script exist in the workspace even after it has stopped running

Outline

- Desktop Basics
- Matrices and Arrays
- Array Indexing
- Workspace Variables
- Text and Characters
- Calling Functions
- 2-D and 3-D Plots
- Programming and Scripts
- Help and Documentation

Help and Documentation

- All MATLAB functions have supporting documentation that includes examples and describes the function inputs, outputs, and calling syntax. There are several ways to access this information from the command line:
- To get info on how to use a function

```
>>help log
```

Help lists related functions at the bottom and links to the doc

- To get a nicer version of help with examples and easy-toread descriptions:

```
>>doc log
```

Exercise 1

- Create a variable start using the function clock
- What does start contain? See help clock
- Convert the vector start to a string. Use the function datestr and name the new variable startString
- Save start and startString into a mat file named startTime

Exercise 2

- Plot y=-x^2 and y1=x^2 in the same figure
- Label the axes and add a title

- Hint: Use function power.

Useful references

- 1. Introduction to Programming with MATLAB （Coursera）
- 2. MATLAB程式設計 張智星
- 3. https://mirlab.org/jang/books/matlabProgramming4beginner/