- We can write "sin 2" in text, but Matlab requires parentheses surrounding the "2" (which is called the function argument or parameter).
- Thus, to evaluate "sin 2" in Matlab, we type sin(2). The Matlab function name must be followed by a pair of parentheses that surround the argument.
- Examples:
  To evaluate $\sin(x^2+5)$ you type: sin(x^2+5).
  To evaluate $\sin(\sqrt{x}+1)$, you type: sin(sqrt(x)+1).
- Using a function as an argument for another function is called function composition. Be sure to check the order of precedence, and the number and placement of parentheses when typing such expressions.

Common Mathematical Functions

`exp(x)`	Exponential e^x
`sqrt(x)`	Square Root
`log(x)`	Natural Logarithm
`log10(x)`	Common Base Logarithm (base 10)
`abs(x)`	Absolute value of a number or Magnitude of a complex number.
`angle(x)`	Angle of a Complex Number
`conj(x)`	Complex Conjugate
`imag(x)`	Imaginary Part of a Complex Number
`real(x)`	Real Part of a Complex Number
`ceil(x)`	Round to nearest integer towards ∞
`fix(x)`	Round to nearest integer towards zero
`floor(x)`	Round to nearest integer towards $-\infty$
`round(x)`	Round towards nearest integer
`sign(x)`	It gives 1 if x>0, 0 if x=0, -1 if x<0
`cos(x)`	Cosine
`cot(x)`	Cotangent
`csc(x)`	Cosecant
`sec(x)`	Secant
`sin(x)`	Sine
`tan(x)`	Tangent
`acos(x)`	Inverse Cosine
`acot(x)`	Inverse Cotangent
`acsc(x)`	Inverse Cosecant
`asec(x)`	Inverse Secant
`asin(x)`	Inverse Sine
`atan(x)`	Inverse Tangent
`atan2(x,y)`	Inverse Tangent specifying the Quadrant
`cosh(x)`	Hyperbolic Cosine
`coth(x)`	Hyperbolic Cotangent
`csch(x)`	Hyperbolic Cosecant

sech(x)	Hyperbolic Secant
sinh(x)	Hyperbolic Sine
tanh(x)	Hyperbolic Tangent
acosh(x)	Inverse Hyperbolic Cosine
acoth(x)	Inverse Hyperbolic Cotangent
acsch(x)	Inverse Hyperbolic Cosecant
asech(x)	Inverse Hyperbolic Secant
asinh(x)	Inverse Hyperbolic Sine
atanh(x)	Inverse Hyperbolic Tangent

Script Files – Introduction

- Up to this point, we have used Matlab in the interactive mode (Commands are entered in the Command Window)
- The interactive mode is desirable when we have few commands
- For larger problems, we use Script Files.
- The script file contains the same Matlab Commands that we type in the Command Window, but the script file can be constantly modified and saved without having to retype all the commands again.

Script Files – Creation

- Click on 'New Script' to create your script file.
- Enter your commands as desired.
- Use:
  ; to suppress the result of commands
  % to insert comments
- Script files are saved with a .m extension.
- Run your script file by typing fileName.m in the command window, or by clicking on Run in the file editor.

```
% Program fallingSpeed.m
% Plots speed of falling object
% Created on 23/9/2019 by S. Rishmawi
%
% Input Variable:
% tf: Final Time in seconds
%
% Output Variables:
% t: array of times at which the speed is computed in seconds
% v: array of speeds (m/s)

clear
clc

% Parameter Value:
g = 9.81;     % Gravitational Acceleration in SI Units
```

```
% Input Section:
tf = input('Enter final time in seconds:');

% Calculation Section:
dt = tf/500;
% Create an array of 501 time values:
t = [0:dt:tf];
% Compute Speed Values:
v = g*t;

% Output Section:
plot(t,v)
xlabel('Time (s)')
ylabel('Velocity (m/s)')
```

Notes on Script File Names

- The name of the script file must begin with a letter, and may include digits and the underscore character, up to 31 characters.
- Do not give a script file the same name as a variable.
- Do not give a script file the same name as a Matlab command or function. You can check to see if a command, function, or file name already exists by using the "exist" command.

Some input/output Commands

Command	Description
disp(A)	Displays the contents, but not the name, of the array A.
disp('text')	Displays the text string enclosed within quotes.
x = input('text')	Displays the text in quotes, waits for user input from the keyboard, and stores the value in x.
x = input('text','s')	Displays the text in quotes, waits for user input from the keyboard, and stores the input as a string in x.

User-Defined Function

- Another type of m-files.
- It is simply a script file that implements some operation that is not available in Matlab.
- It uses Matlab Commands and accepts user arguments.
- To define your function, begin the script file with
  `function [output variables] = name(input variables)`
- The output variables are enclosed in square brackets.
- The input variables are enclosed in parentheses.
- The function name should be the same as the file name in which it is saved with the .m extension.

```
function [z] = fun(x,y)
u = 3*x;
z = u+6*y.^2;
```

To call this function, type the following in the command window:

```
>> z = fun(3,7)
>> z = 303
```

The function uses x=3 and y=7 to compute z.

- The use of the semicolon prevents the values of u and z from being displayed.
- The use of (.^) enables the function to accept y as an array.
- The variables defined inside the function are local to the function and are cleared when the function returns.

Functions – Local Variables

- The names of the variables given in the function definition line are local to that function.
- Other variable names can be used when you call that function.
- All variables inside a function are erased after the function finishes executing, except when the same variable names appear in the output variable list used in the function call.

Functions – Global Variables

- The 'global' keyword declares certain variables global, and therefore their values are available to the basic workspace and to other functions that declare these variables global.
- The syntax to declare the variables a, x, and q global is
  `global a x q`
- Any assignment to those variables, in any function or in the base workspace, is available to all the other functions declaring them global.

Subfunctions

- Primary: The first function in an m-file which has the same name as the file name. It is used to call the function from the command line or from another m-file.
- Subfunctions: Functions defined in the same file as the primary function and they are available to it only. They are used to break large tasks into smaller ones.

```
% Function to compute Circle Parameters
% Input: Radius of the Circle
% Outputs:
% 1. Area of a Circle
% 2. Circumference of a Circle
% The function contains two subfunctions

% The Main Function
function [area,circum] = circleParameters(radius)
area = computeArea(radius);
circum = computeCircum(radius);

% Function to compute the area
function a = computeArea(r)
a = pi*r^2;

% Function to compute circumference
function c = computeCircum(r)
c = 2*pi*r;
```

Working with Data Files

- Matlab has the capability of importing data created by other applications into its workspace.
- Also, it is capable of exporting workspace variables so that it can be used by other applications.

Importing and Exporting Spreadsheet Files

- The command: `A = xlsread('filename')`imports the Microsoft Excel workbook file filename.xls into the array A.
- The command: `[A,B] = xlsread('filename')`imports all numeric data into the array A and all text data into the cell array B.
- The command: `[success,message] = xlswrite('filename',array)`writes the array into the Excel file whose name is given by 'filename'.

Working with CSV

- CSV stands for Comma-Separated-Values
- One format that is used to exchange data between applications

- Some Properties:
  - Each record is a line
  - Records are separated by Enter
  - Record fields are separated by commas
  - Leading and trailing spaces adjacent to commas are ignored
- Commands to:
  Read CSV data into matrix X
  `X = csvread(fileName)`
  Write matrix M into file fileName
  `csvwrite(fileName,M)`