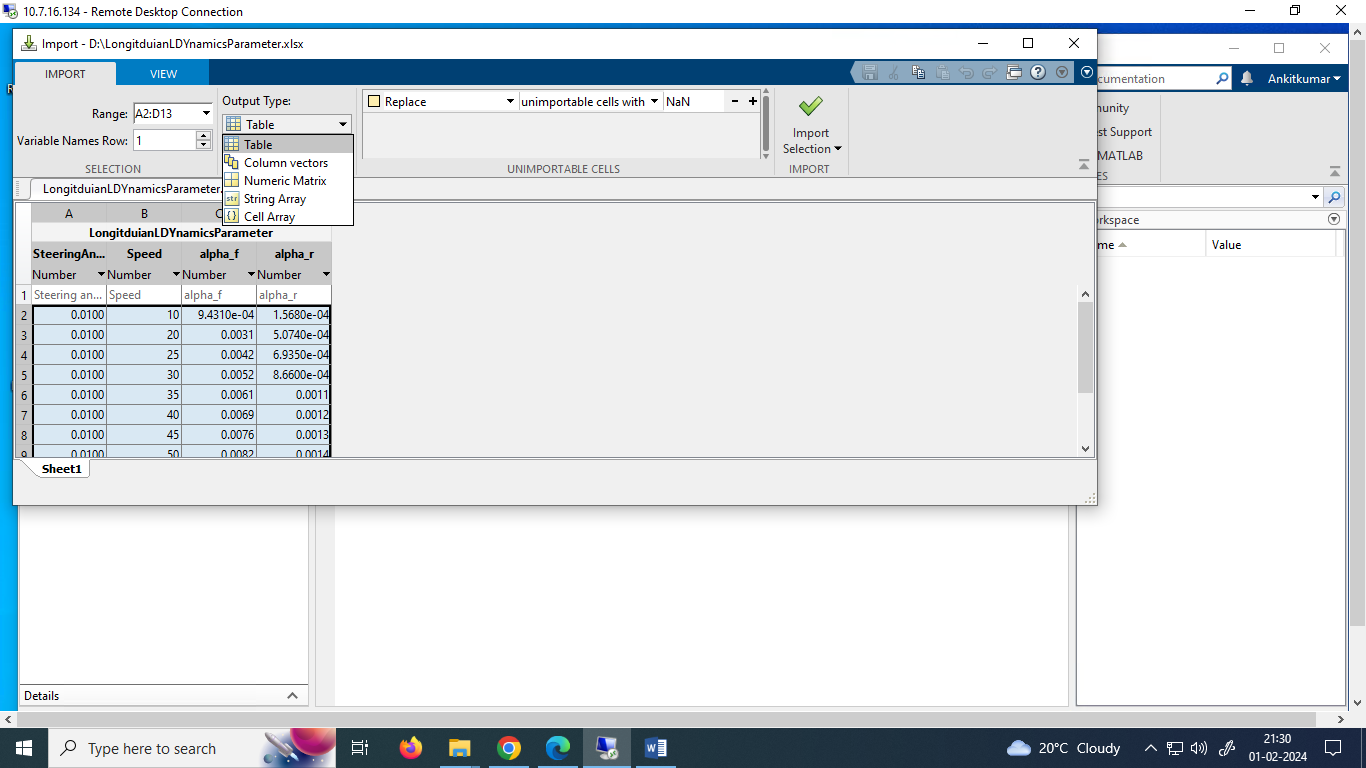
## Import EXCEL sheet as .. access, slice data



1. **Imported the excel data as Table**

>> table

StAngle VehSpeed alpha\_f alpha\_r

0.0100000000000000 10 0.000943100000000000 0.000156800000000000

0.0100000000000000 20 0.00305300000000000 0.000507400000000000

0.0100000000000000 25 0.00417300000000000 0.000693500000000000

0.0100000000000000 30 0.00521100000000000 0.000866000000000000

0.0100000000000000 35 0.00613200000000000 0.00107900000000000

0.0100000000000000 40 0.00692500000000000 0.00115000000000000

0.0100000000000000 45 0.00760000000000000 0.00126200000000000

0.0100000000000000 50 0.00816800000000000 0.00135600000000000

0.0100000000000000 55 0.00864700000000000 0.00143600000000000

0.0100000000000000 60 0.00905000000000000 0.00150200000000000

0.0100000000000000 70 0.00968000000000000 0.00160700000000000

0.0100000000000000 80 0.0101400000000000 0.00168200000000000

**Accessing column of the table by its name:**

>> table.VehSpeed

ans =

10

20

25

30

35

40

45

50

55

60

70

80

**Accessing whole column:**

>> table(:,2)

ans =

12×1 table

VehSpeed

\_\_\_\_\_\_\_\_

10

20

25

30

35

40

45

50

55

60

70

80

**Accessing the whole row:**

>> table(2,:)

ans =

1×4 table

StAngle VehSpeed alpha\_f alpha\_r

\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

0.01 20 0.003053 0.0005074

**Slicing of table:**

>> table(1:2,2:3)

ans =

2×2 table

Speed alpha\_f

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

10 0.0009431

20 0.003053

1. **Imported that ecel data as Numeric Matrix**

>> NumericMatrix

NumericMatrix =

0.0100 10.0000 0.0009 0.0002

0.0100 20.0000 0.0031 0.0005

0.0100 25.0000 0.0042 0.0007

0.0100 30.0000 0.0052 0.0009

0.0100 35.0000 0.0061 0.0011

0.0100 40.0000 0.0069 0.0011

0.0100 45.0000 0.0076 0.0013

0.0100 50.0000 0.0082 0.0014

0.0100 55.0000 0.0086 0.0014

0.0100 60.0000 0.0091 0.0015

0.0100 70.0000 0.0097 0.0016

0.0100 80.0000 0.0101 0.0017

**Slicing of Numeric matrix:**

>> NumericMatrix(6)

ans =

0.0100

>> NumericMatrix(2,3)

ans =

0.0031

>> NumericMatrix(:,3)

ans =

0.0009

0.0031

0.0042

0.0052

0.0061

0.0069

0.0076

0.0082

0.0086

0.0091

0.0097

0.0101

>> NumericMatrix(2,:)

ans =

0.0100 20.0000 0.0031 0.0005

1. **Imported as Cell Array**

>> CellArray

CellArray =

12×4 cell array

{[0.0100]} {[10]} {[9.4310e-04]} {[1.5680e-04]}

{[0.0100]} {[20]} {[ 0.0031]} {[5.0740e-04]}

{[0.0100]} {[25]} {[ 0.0042]} {[6.9350e-04]}

{[0.0100]} {[30]} {[ 0.0052]} {[8.6600e-04]}

{[0.0100]} {[35]} {[ 0.0061]} {[ 0.0011]}

{[0.0100]} {[40]} {[ 0.0069]} {[ 0.0011]}

{[0.0100]} {[45]} {[ 0.0076]} {[ 0.0013]}

{[0.0100]} {[50]} {[ 0.0082]} {[ 0.0014]}

{[0.0100]} {[55]} {[ 0.0086]} {[ 0.0014]}

{[0.0100]} {[60]} {[ 0.0091]} {[ 0.0015]}

{[0.0100]} {[70]} {[ 0.0097]} {[ 0.0016]}

{[0.0100]} {[80]} {[ 0.0101]} {[ 0.0017]}

>> CellArray(2)

ans =

1×1 cell array

{[0.0100]}

>> CellArray(:,2)

ans =

12×1 cell array

{[10]}

{[20]}

{[25]}

{[30]}

{[35]}

{[40]}

{[45]}

{[50]}

{[55]}

{[60]}

{[70]}

{[80]}

>> CellArray(2,:)

ans =

1×4 cell array

{[0.0100]} {[20]} {[0.0031]} {[5.0740e-04]}

>> CellArray(1:2,2:3)

ans =

2×2 cell array

{[10]} {[9.4310e-04]}

{[20]} {[ 0.0031]}

1. **Imported as String Array**

>> stringArray

stringArray =

12×4 string array

"0.01" "10" "0.0009431" "0.0001568"

"0.01" "20" "0.003053" "0.0005074"

"0.01" "25" "0.004173" "0.0006935"

"0.01" "30" "0.005211" "0.000866"

"0.01" "35" "0.006132" "0.001079"

"0.01" "40" "0.006925" "0.00115"

"0.01" "45" "0.0076" "0.001262"

"0.01" "50" "0.008168" "0.001356"

"0.01" "55" "0.008647" "0.001436"

"0.01" "60" "0.00905" "0.001502"

"0.01" "70" "0.00968" "0.001607"

"0.01" "80" "0.01014" "0.001682"

>> stringArray(1:2,2:3)

ans =

2×2 string array

"10" "0.0009431"

"20" "0.003053"

>> stringArray(:,2)

ans =

12×1 string array

"10"

"20"

"25"

"30"

"35"

"40"

"45"

"50"

"55"

"60"

"70"

"80"

>> stringArray(2,:)

ans =

1×4 string array

"0.01" "20" "0.003053" "0.0005074"

1. **Imported as vector**

Columns will be separated and saved as vectors with heading of the column as a name of the vector. For example: All 12 x 1 duble in the screenshot of workspace below are the vectors

>> alpha\_r(2)

ans =

5.0740e-04

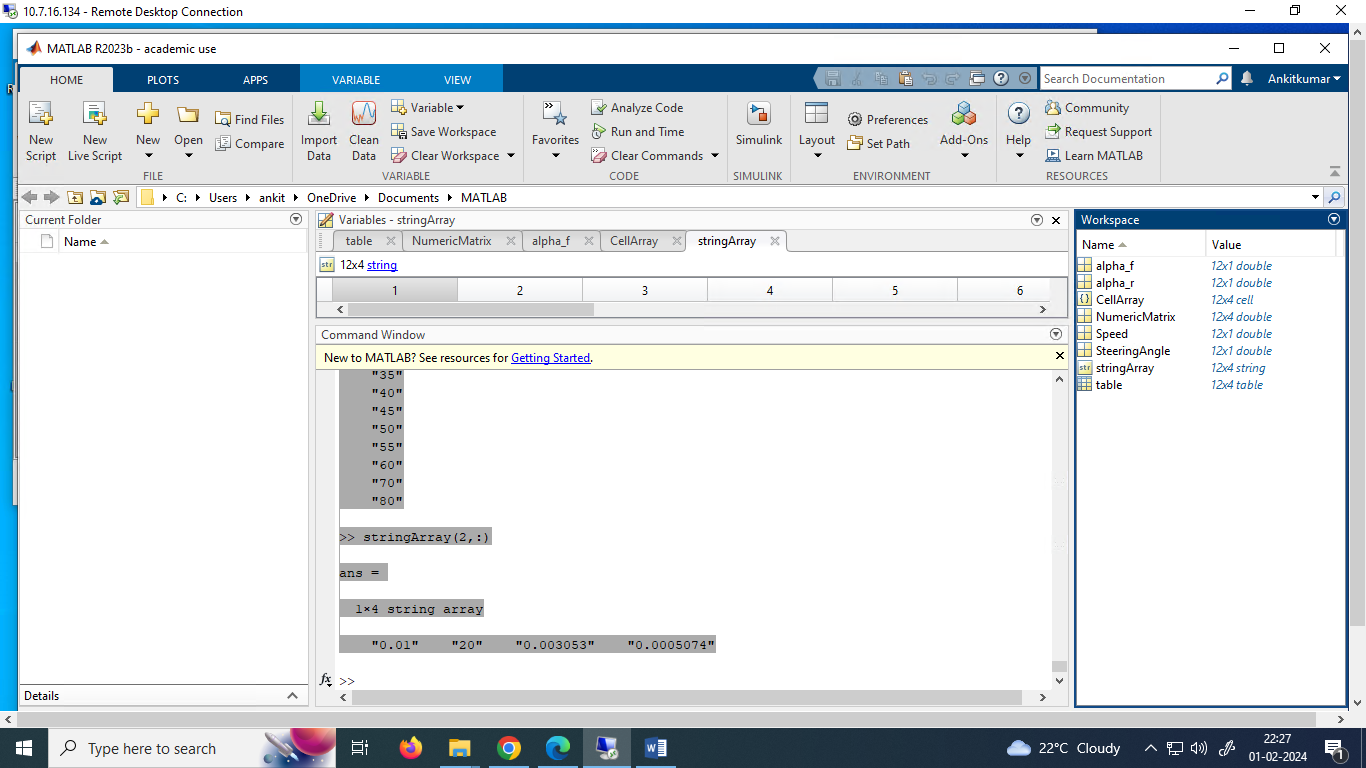
>> alpha\_r(2:3)

ans =

1.0e-03 \*

0.5074

0.6935



## Access an element

% Create a 5x5 matrix

A = rand(5, 5);

% Access the element in the 3rd row and 2nd column

element = A(3, 2);

disp(element);

## Find an element

% Find the indices of a specific value (e.g., 0.5)

value = 0.5;

[row, col] = find(A == value);

disp([row, col]);

or

A = [2, 0, 5; 0, 10, 0; 3, 0, 2];

indices = find(A > 0);

indices = [1;3;5;7;9]

## Slicing of string

1. Extract a single character: **singleChar = str(1);** for "H"
2. Extract a substring: **substring = str(1:5);** for "Hello"
3. Extract from start to a specific position: **startToPosition = str(1:7);** for "Hello, "
4. Extract the last character: **lastChar = str(end);** for "!"
5. Extract a range from the end: **rangeFromEnd = str(end-5:end);** for "World!"

## Indexing and Slicing

% Create a 3x3 matrix

A = [1, 2, 3; 4, 5, 6; 7, 8, 9];

**% Extract a single element**

element = A(2, 3); % Retrieves the element in the 2nd row and 3rd column

**% Extract an entire row or column**

row\_vector = A(2, :); % Retrieves the entire 2nd row

col\_vector = A(:, 3); % Retrieves the entire 3rd column

**% Create a logical matrix**

B = A > 4;

% Use logical indexing to extract elements that satisfy a condition

selected\_elements = A(B);

B

0 0 0

0 1 1

1 1 1

Selected\_elements

7 5 8 6 9

% Create a 4x4 matrix

D = **magic**(4);

**% Extract a submatrix**

submatrix = D(2:3, 1:2); % Rows 2 to 3, Columns 1 to 2

% Create a 5x5 matrix

E = rand(5, 5);

% Extract portions of a matrix using 'end'

first\_column = E(:, 1:**end**-1); % All rows, columns 1 to end-1

## Slicing of Numeric Data

**% Create a row vector**

vector = [1, 2, 3, 4, 5];

% Slice: Access elements 2 to 4

sliced\_vector = vector(2:4);

**% Create a column vector**

column\_vector = [1; 2; 3; 4; 5];

% Slice: Access elements 2 to 4

sliced\_column\_vector = column\_vector(2:4);

**% Create a 2x3 matrix**

matrix = [1, 2, 3; 4, 5, 6];

% Slice: Access element in the first row, second column

sliced\_element = matrix(1, 2);

**% Create a 2x3 array**

array = [1, 2, 3; 4, 5, 6];

% Slice: Access element in the first row, second column

sliced\_element = array(1, 2);

**% Create a 3D array (2x3x4)**

threeD\_array = rand(2, 3, 4);

% Slice: Access element in the first row, second column, third layer

sliced\_element = threeD\_array(1, 2, 3);

**% Create a table**

t = table('Size', [3, 2], 'VariableTypes', {'double', 'char'}, 'VariableNames', {'Age', 'Name'});

t.Age = [25; 30; 22];

t.Name = {'John'; 'Mary'; 'Bob'};

% Slice: Access ages

sliced\_ages = t.Age;

## Precedence of operator (PEMDAS)

1. **Parentheses ()**
2. Transpose (.'), **power (.^)**, complex conjugate transpose ('), matrix power (^)
3. Power with unary minus (.^-), unary plus (.^+), or logical negation (.^~) as well as matrix power with unary minus (^-), unary plus (^+), or logical negation (^~).
4. Unary plus (+), unary minus (-), logical negation (~)
5. **Multiplication (.\*)**, **right division (./)**, left division (.\), matrix multiplication (\*), matrix right division (/), matrix left division (\)
6. **Addition (+), subtraction (-)**
7. Colon operator (:)
8. **Less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)**
9. Element-wise AND (&)
10. Element-wise OR (|)
11. Short-circuit AND (&&)
12. Short-circuit OR (||)

## Operators Code

% Vectors

v = [1, 2, 3, 4, 5];

% or using colon operator

v = 1:5;

% Matrices

A = [1, 2, 3; 4, 5, 6; 7, 8, 9];

% Matrix operations

B = [9, 8, 7; 6, 5, 4; 3, 2, 1];

sum\_matrices = A + B;

matrix\_product = A \* B; % martices multiplication

elementwise\_product = A .\* B; % element-wise multiplication

### Washing machine Problem

#### %Store result as table

*%Cloths given*

cloths = ["baby"; "woolen"; "synthetic"];

*%Initialize vectors for washtime and hotwater*

washTime = zeros(length(cloths),1);

hotWater = zeros(length(cloths),1);

*%Implement logic*

for i = 1:length(cloths)

if cloths(i) == "baby"

washTime(i) = 20;

hotWater(i) = 1;

elseif cloths(i) == "woolen"

washTime(i) = 25;

else

washTime(i) = 15;

end

end

*% results in table format*

result = table(cloths, washTime, hotWater,'VariableNames', {'Cloths', 'Wash time', 'Hot water'});

disp(result)

*%write table to the exceland csv*

writetable(result,"clothsInfo.xlsx");

writetable(result, "clothsInfo.csv");

**>> WashingMachine**

**Cloths Wash time Hot water**

**\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_**

**"baby" 20 1**

**"woolen" 25 0**

**"synthetic" 15 0**

#### %Store result as matrix

%Cloths given

cloths = ["baby"; "woolen"; "synthetic"];

%Initialize matrix

result = zeros(length(cloths),3); % 3 columns respectively for cloths, washTime and hotWater

for i = 1:length(cloths)

if cloths(i) == "baby"

washTime(i) = 20;

hotWater(i) = 1;

elseif cloths(i) == "woolen"

washTime(i) = 25;

else

washTime(i) = 15;

end

result(i,1) = cloths(i);

result(i,2) = washTime(i);

result(i,3) = hotWater(i);

end

disp(result)

**>> WashingMachine**

**NaN 20 1**

**NaN 25 0**

**NaN 15 0**

### % while loop

j = 1;

while j <= 5

disp(j);

j = j + 1;

end

## Plot

### % Plot-1

x = linspace(0, 2\*pi, 100);

y = sin(x);

plot(x, y);

title('Sine Wave');

xlabel('Angle (radians)');

ylabel('sin(x)');

### %Plot-2

hold off;

plot(linspace(0,3.14,100), sin(linspace(0,3.14,100)),'LineWidth',1,'Color','k','LineStyle','-')

h=legend('Ideal trajectory');

XLabel = xlabel('x (rad)');

YLabel = ylabel('sin(x)');

title('Sine Wave');

set(gca,'Box','off','TickDir','out','TickLength',[0.020 0.020], ...

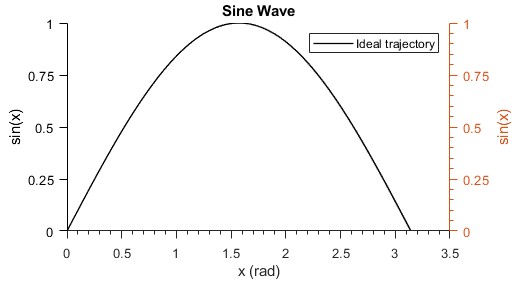
'XMinorTick' , 'on' , ...

'YMinorTick' , 'on' , ...

'YGrid' , 'off' ,...

'XGrid' , 'off' );

yticks([0 0.25 0.5 0.75 1.0 ])



### %Plot-3 (Dual Y axis)

hold off;

yyaxis left;

plot(linspace(0,pi,100),sin(linspace(0,3.14,100)),'LineWidth',1,'Color','k','LineStyle','-');

ylabel('sin(x)');

yticks([0 0.25 0.5 0.75 1.0 ])

yyaxis right;

plot(linspace(0,pi,100),cos(linspace(0,3.14,100)),'LineWidth',1,'Color','k','LineStyle','-')

ylabel('cosine(x)');

yticks([-1.0, -0.75, -0.5, -0.25, 0 0.25 0.5 0.75 1.0 ])

xlabel('x (rad)');

h=legend('Sine', 'cosine');

title('Sine and Cosine Wave');

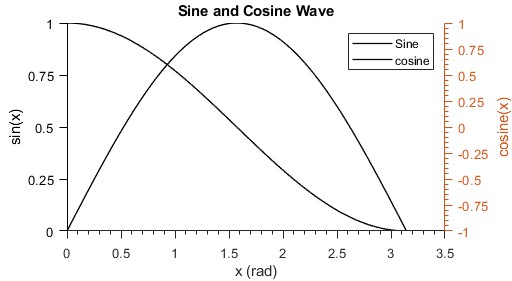
set(gca,'Box','off','TickDir','out','TickLength',[0.020 0.020], ...

'XMinorTick' , 'on' , ...

'YMinorTick' , 'on' , ...

'YGrid' , 'off' ,...

'XGrid' , 'off' );



## Define functions and method

### %Example-1 (Adding two numbers)

*% This is the main script part*

my\_result = add\_numbers(2,3);

disp(my\_result);

*% Below is the local function section*

function result = add\_numbers(x, y)

result = x + y;

end

## Object oriented programming

### Example-1 (Digit Adder)

**///////////////////////////////////////Save this as “class name.m”. here, “TwoDigitAdder.m”//////////////////////////////////////////**

*%Class Definition*

classdef TwoDigitAdder

properties

Digit1 = 1;

Digit2 = 2;

end

methods

function obj = TwoDigitAdder(digit1,digit2)

*%Constructor*

if nargin == 2

obj.Digit1 = digit1;

obj.Digit2 = digit2;

elseif nargin == 1

obj.Digit1 = digit1;

end

end

function result= addDigits(obj)

result = obj.Digit1+obj.Digit2;

end

end

end

**////////////////////////////////////////////////////////////////////////////////////////You will see class “TwoDigitAdder” created in workspace. It contains its properties///////////////////////////////////////////////////////////////////////////////////////////////////////////////**

**////////////////////////////////////////Calling class, accessing its propoertis and using its methods///////////////////////////////////**

*%Object Creation*

MyAdder = TwoDigitAdder(10,20); *%alternatively check with 0,1, and 2 arguments*

*%Access properties*

disp(["Digit1 is",MyAdder.Digit1]);

disp(["Digit2 is",MyAdder.Digit2]);

*%Use methods*

my\_result = MyAdder.addDigits();

disp(["my\_result is: ", my\_result])

**%Results returned:**

>> OOP\_TwoDigitAdder

"Digit1 is" "10"

"Digit2 is" "20"

"my\_result is: " "30"

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**/////////////////////////////////////Save this as “calss’ name.m” here “ThreeDigitAdder.m”//////////////////////////////////////////**

*% Class definition for ThreeDigitAdder that inherits from TwoDigitAdder*

classdef ThreeDigitAdder < TwoDigitAdder

properties

Digit3; % New property for the third digit

end

methods

*% Constructor*

function obj = ThreeDigitAdder(digit1, digit2, digit3)

*% Provide default values if no arguments are given*

if nargin < 1

digit1 = 1;

digit2 = 2;

digit3 = 3;

end

if nargin < 2

digit2 = 2;

digit3 = 3;

end

if nargin < 3

digit3 = 3;

end

*% Call superclass constructor*

obj@TwoDigitAdder(digit1, digit2);

*% Initialize the third digit*

obj.Digit3 = digit3;

end

*% Overridden addDigits method to add three digits*

function result = addDigits(obj)

*% Call the addDigits method from the superclass*

result = addDigits@TwoDigitAdder(obj);

*% Add the third digit to the result*

result = result + obj.Digit3;

end

end

end

**////////////////////////////////////////////////////////////////////////////////////////You will see class “TwoDigitAdder” created in workspace. It contains its properties///////////////////////////////////////////////////////////////////////////////////////////////////////////////**

**////////////////////////////////////////Calling class, accessing its properties and using its methods///////////////////////////////////**

*%Object Creation*

MyAdder = ThreeDigitAdder(10,20,30); *%alternatively check with 0,1,2,and 3 arguments*

*%Access properties*

disp(["Digit1 is",MyAdder.Digit1]);

disp(["Digit2 is",MyAdder.Digit2]);

disp(["Digit2 is",MyAdder.Digit3]);

*%Use method*

my\_result = MyAdder.addDigits();

disp(["my\_result is: ", my\_result])

**%Results returned:**

>> OOP\_ThreeDigitAdder

"Digit1 is" "10"

"Digit2 is" "20"

"Digit2 is" "30"

"my\_result is: " "60"

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////