# Index:

Contents

[Index: 1](#_Toc530644070)

[Tutorial Week 1: 2](#_Toc530644071)

[Create variables in the command window: 2](#_Toc530644072)

[Create variables based on other variables: 2](#_Toc530644073)

[Tutorial Week 2: 3](#_Toc530644074)

[Examples: 3](#_Toc530644075)

[Create Vectors: 3](#_Toc530644076)

[Operations with vectors 3](#_Toc530644077)

[Create column vectors: 3](#_Toc530644078)

[Other ways to create vectors: 4](#_Toc530644079)

[Creating matrices: 4](#_Toc530644080)

[Operations with matrices: 5](#_Toc530644081)

[Accessing elements in matrix: 6](#_Toc530644082)

[Tutorial Week 4: 7](#_Toc530644083)

# Tutorial Week 1:

## Create variables in the command window:

* 1. a = 2
     1. 2
  2. b = 3
     1. 3
  3. a + b
     1. 5
  4. first\_string = 'My name is '
     1. ‘My name is ‘
  5. second\_string = ' John'
     1. ‘ John’

## Create variables based on other variables:

* 1. c = a \* 2
     1. 4
  2. d = cos(b)
     1. -0.9900
  3. e = c + d
     1. 3.0100
  4. r = 5
     1. 5
  5. A = 2 \* pi \* r
     1. 31.4159
  6. C = 2 \* pi \* r
     1. 31.4159
  7. x = 0
     1. 0
  8. curve\_f = sin(x) + cos(x/3+1)
     1. 0.5403

# Tutorial Week 2:

## Examples:

* 1. A = ones(100,100);
     1. 100x100
  2. A = ones(1000,1000);
     1. 1000x1000
  3. A = ones(10000,10000);
     1. 10000x10000
  4. A = ones(100000,100000);
     1. Error = array exceeds max size

## Create Vectors:

* 1. vector\_1 = [1 2 3 4 5 6 7 8 9 10]
     1. 1 2 3 4 5 6 7 8 9 10
  2. vector\_2 = [12 13 14 15 16 17 8767826264]
     1. 1.0e+09 \*
     2. Columns 1 through 6
     3. 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
     4. Column 7
     5. 8.7678

Operations with vectors:

* 1. vec\_1 = [1 2 3]
     1. 1 2 3
  2. vec\_2 = [ 7 8 9]
     1. 7 8 9
  3. vec\_1 + 10
     1. 11 12 13
  4. vec\_1 + vec\_2
     1. 8 10 12
  5. vec\_1 - vec\_2
     1. -6 -6 -6
  6. times(vec\_1, vec\_2)
     1. 7 16 27

## Create column vectors:

* 1. colu\_1 = [1; 2; 3; 4; 5]
     1. 1

2

3

4

5

* 1. colu\_2 = [23; 24; 25; 26]
     1. 23

24

25

26

* 1. colu\_3 = ['aa'; 'bb'; 'cc'; 'dd']
     1. 4×2 char array
     2. 'aa'

'bb'

'cc'

dd'

## Other ways to create vectors:

* 1. z = zeros(5,1)
     1. 0

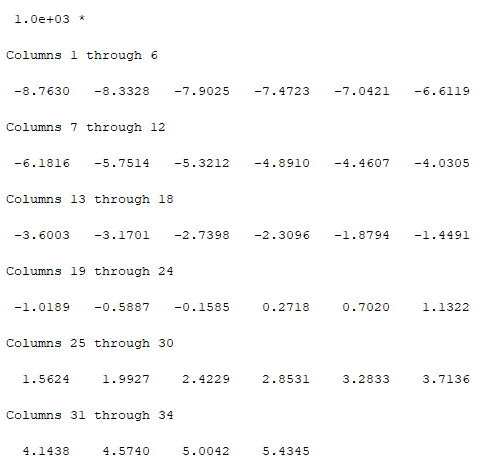
0

0

0

0

* 1. zz = zeros ( 1, 5)
     1. 0 0 0 0 0
  2. zzz = [0 : 1 : 10]
     1. 0 1 2 3 4 5 6 7 8 9 10
  3. zzzz = [-8763 : 430.2265 : 5634.23]



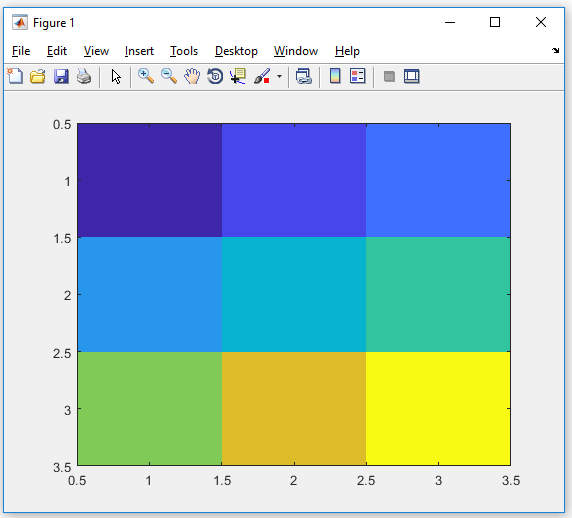
## Creating matrices:

* 1. matr\_1 = [ 1 2 3 ; 4 5 6; 7 8 10]
     1. 1 2 3
     2. 4 5 6
     3. 7 8 10
  2. matr\_2 = ['lala' 'lele'; 'lili' 'lolo'; 'lulu' 'lala']
     1. 'lalalele'
     2. 'lililolo'
     3. 'lululala'

## Operations with matrices:

* 1. matr\_1 + 10
     1. 11 12 13
     2. 14 15 16
     3. 17 18 20
  2. sin(matr\_1)
     1. 0.8415 0.9093 0.1411
     2. -0.7568 -0.9589 -0.2794
     3. 0.6570 0.9894 -0.5440
  3. matr\_1'
     1. 1 4 7
     2. 2 5 8
     3. 3 6 10
  4. inv(matr\_1)
     1. -0.6667 -1.3333 1.0000
     2. -0.6667 3.6667 -2.0000
     3. 1.0000 -2.0000 1.0000
  5. indetity\_matrix = matr\_1 \* inv(matr\_1)
     1. 1.0000 -0.0000 -0.0000
     2. 0.0000 1.0000 -0.0000
     3. 0.0000 -0.0000 1.0000
  6. element\_mulitiplication = matr\_1.\*matr\_1
     1. 1 4 9
     2. 16 25 36
     3. 49 64 100

## Accessing elements in matrix:

* 1. matr\_1(1,2)
     1. 2
  2. matr\_1(8)
     1. 6
  3. matr\_1(1:3,2)
     1. 2
     2. 5
     3. 8
  4. matr\_1(3, :)
     1. 7 8 10
  5. imagesc(matr\_1)
     1. 

# Tutorial Week 4:

## Exercise 1:

1. a=0:1:4
2. b=5:-1:1

a + b = 5 5 5 5 5

a \* b = error

a' \* b =

0 0 0 0 0

5 4 3 2 1

10 8 6 4 2

15 12 9 6 3

20 16 12 8 4

a . / b = 0 0.2500 0.6667 1.5000 4.0000

## Exercise 2:

A = (1:3)'

1

2

3

B = repmat(A,1,4)

1 1 1 1

2 2 2 2

3 3 3 3

## Exercise 3:

### Create a multi-dimensional matrix based on the figure below

A(:,:, 1) = [1 0 3; 4 -1 2; 8 2 1]] =

1 0 3

4 -1 2

8 2 1

A(:,:, 2) = [6 8 3; 4 3 6; 5 9 2] :

A(:,:,1) =

1 0 3

4 -1 2

8 2 1

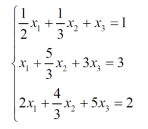
A(:,:,2) =

6 8 3

4 3 6

5 9 2

## Exercise 4:



Step 1: Create a matrix containing coefficients

X = [1/2 1/3 1; 1 5/3 3; 2 4/3 5] =

0.5000 0.3333 1.0000

1.0000 1.6667 3.0000

2.0000 1.3333 5.0000

Step 2: create matrix containing results of each equation

Z = [ 1; 3; 2] =

1

3

2

Step 3:

X \ Z

4

3

-2

Therefore:

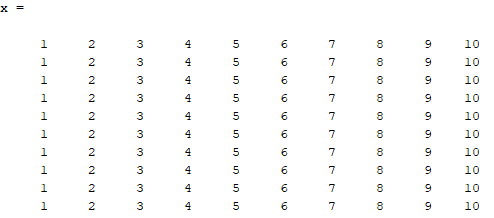
X1 = 4

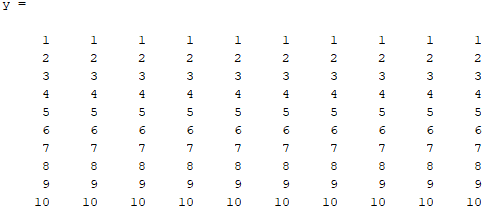
X2 = 3

X3 = -2

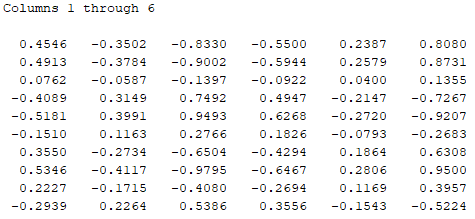
## Exercise 5:

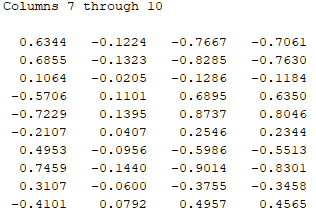
[x y] = meshgrid(1:10, 1:10)





z = cos(x) .\*sin(y





location = find(z == max(max(z)))

=58

fprintf('The maximum occurs when x is %. Of and y is %.Of\n', x(location), y(location))

The maximum occurs when x is >> \*input\*

ans = \*input\*

## Exercise 6:

[a b] = meshgrid(1:997,1:997)

c = sqrt(a.^2 + b.^2)

location = find((a+b+c)==1000 & a<b)

198778

# Tutorial Week 5:

## Exercise 1:

Rewrite following program in a more elegant way;

# Tutorial Week 6:

## Exercise 1:

x = linspace(0, 8, 4) //plots 4 points between 0 & 8

= 0 2.6667 5.3333 8.0000

# Tutorial Week 7:

# Tutorial Week 8:

# Tutorial Week 9:

# Useful links:

<https://uk.mathworks.com/help/matlab/math/multidimensional-arrays.html>

|  |  |
| --- | --- |
| Function: | Purpose: |
| Length | Length of vector or largest matrix dimension |
| Ndims | Number of matrix dimensions |
| Numel | Number of matrix elements |
| Size | Array dimensions |
| Iscolumn | Determines whether input is column vector |
| Isempty | Determines whether array is empty |
| Isrow | Determines whether input is row vector |
| Isscalar | Determines whether input is scalar |
| Blkdiag | Constructs block diagonal matrix from input arguments |
| Ctranspose | Complex conjugate transpose |
| Diag | Diagonal matrices and diagonals of matrix |
| Flipdim | Flips matrix along specified dimension |
| Fliplr | Flips matrix from left to right |
| Ipermute | Inverses permute dimensions of N-D matrix |
| Permute | Rearranges dimensions of N-D matrix |
| Repmat | Replicates and tile matrix |
| Reshape | Reshapes matrix |
| rot90 | Rotates matrix 90 degrees |
| Shiftdim | Shifts dimensions |
| Issorted | Determines whether set elements are in sorted order |
| Sort | Sorts matrix elements in ascending or descending order |
| Sortrows | Sorts rows in ascending order |
| Squeeze | Removes singleton dimensions |
| Transpose | Transpose |
| Vectorise | Vectorizes expression |