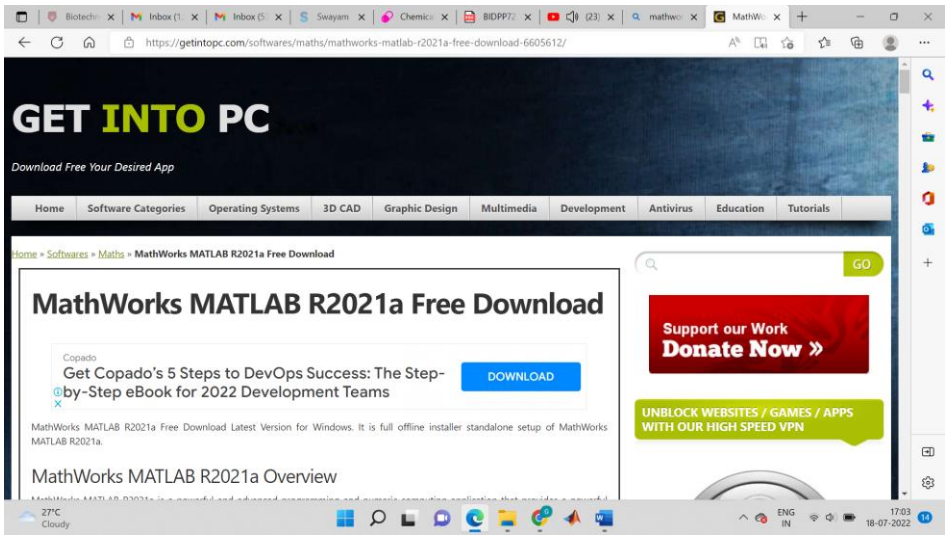
**BID-701 MATLAB in Bioinformatics**

**Assignment 1: Installation of MATLAB**

**Step 1: Search for 2021 link of MATLAB application on google and download the zip file from the link:**

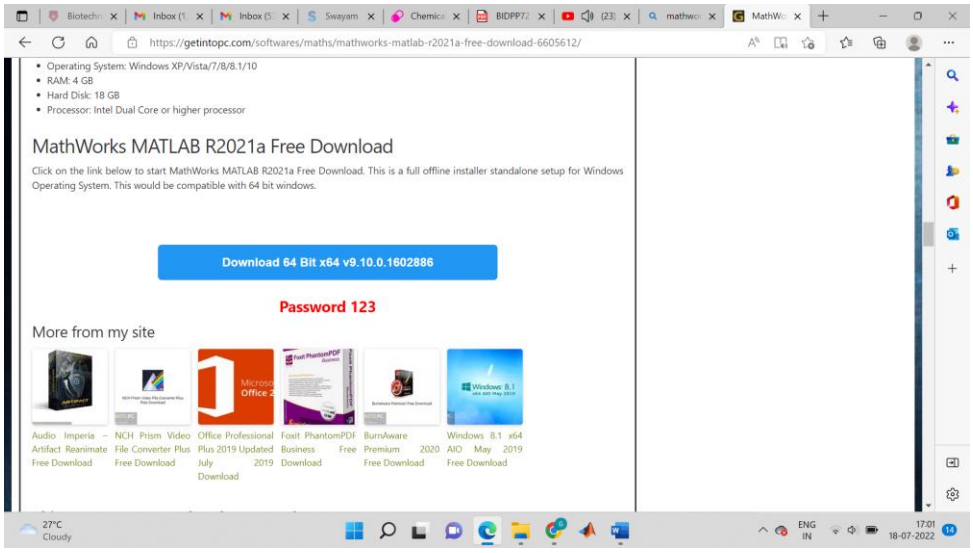
**https://51-68-135-75.xyz/Getintopc.com/MathWorks_MATLAB_R2021a_v9.10.0.1602886.rar?md5=GSTMn7u2otVTo4dgtfNEAw&expires=1660113751**



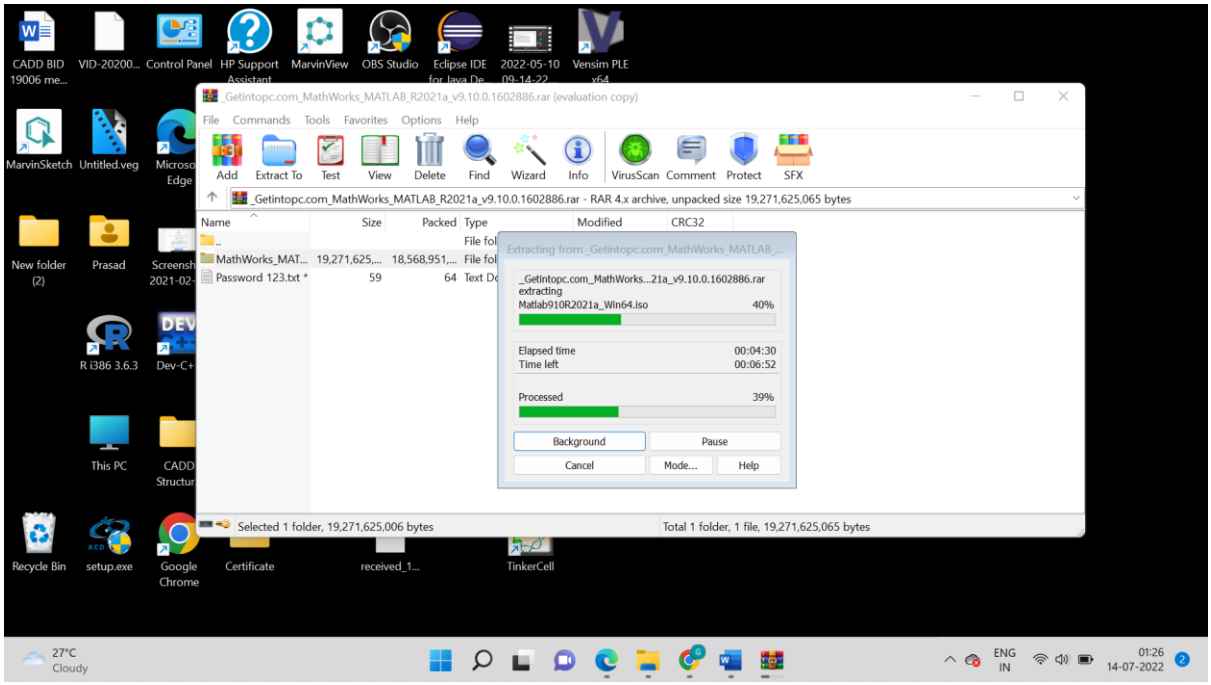

**Check your system requirement:**

**System Requirements for MathWorks MATLAB R2021a**

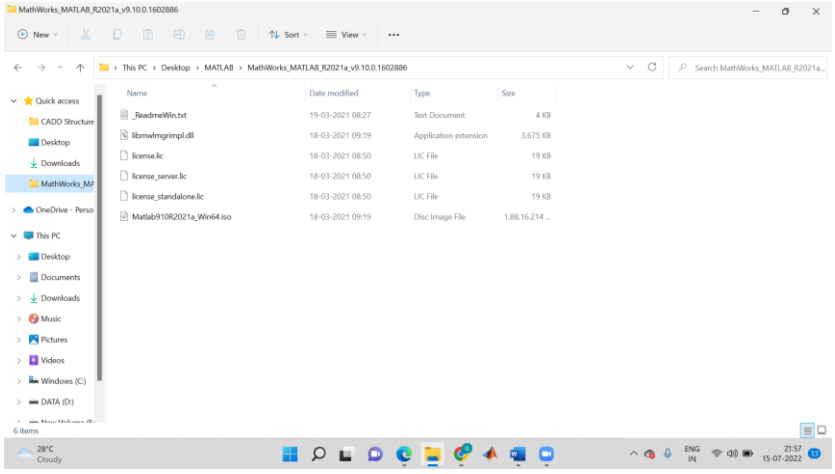
- **Operating System: Windows XP/Vista/7/8/8.1/10**
- **RAM: 4 GB**
- **Hard Disk: 18 GB**
- **Processor: Intel Dual Core or higher processor**



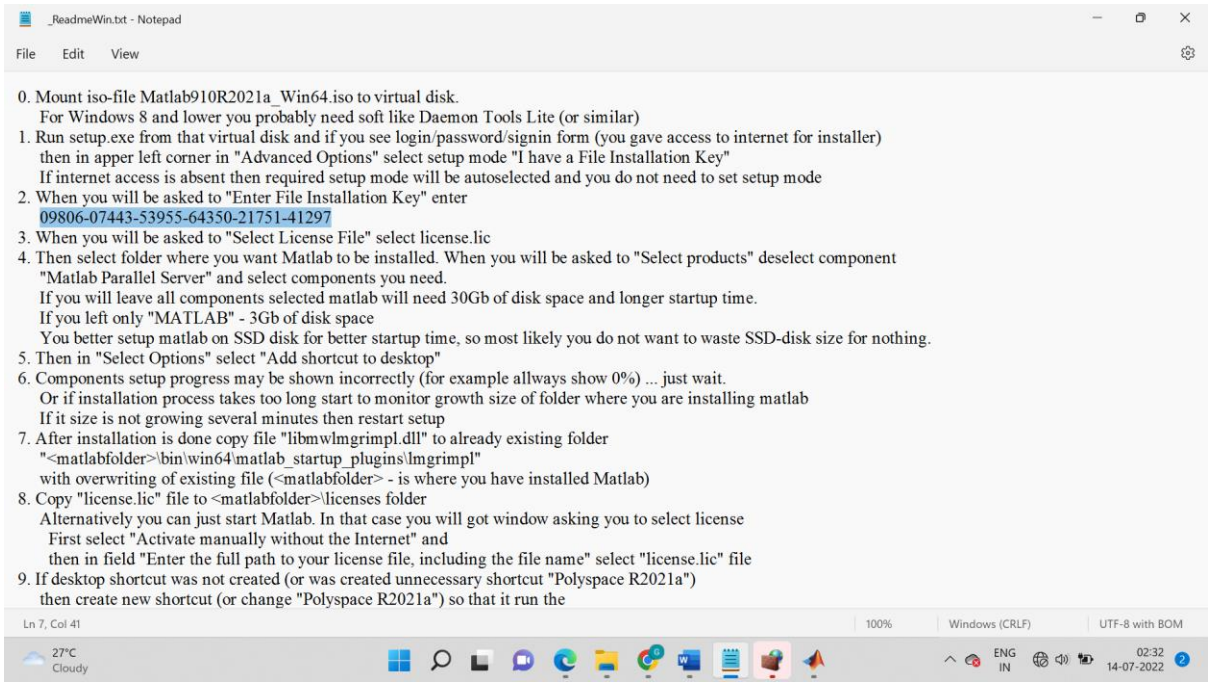
**Step 2: Make a folder named MATLAB on desktop and extract all the zip files from the link to the folder on desktop:**



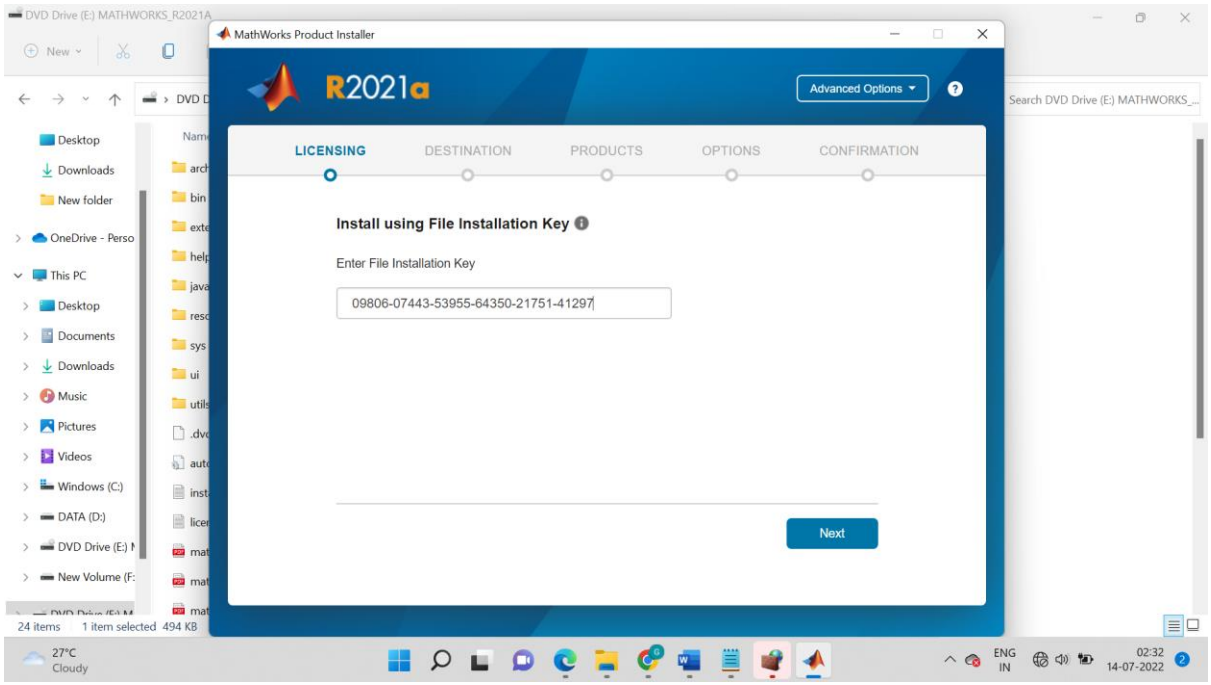

**Step 3: Now open the ReadmeWin text file from the desktop MATLAB folder:**



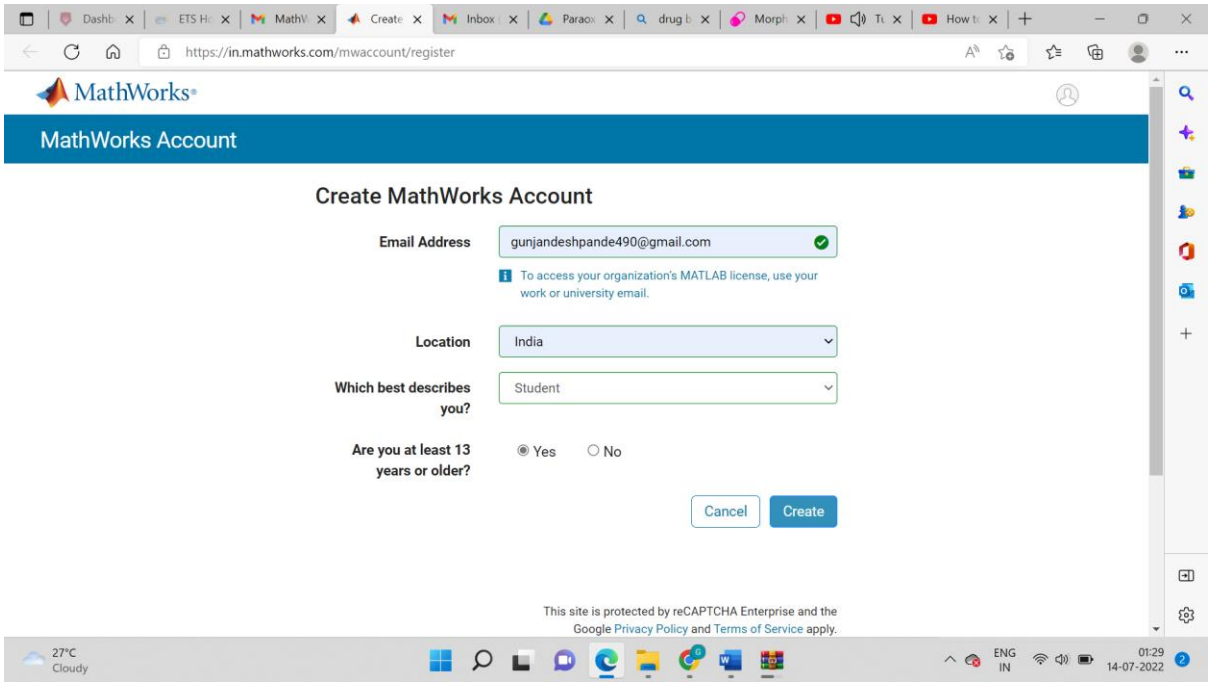
**Step 4: Now copy the installation key from the text file:**



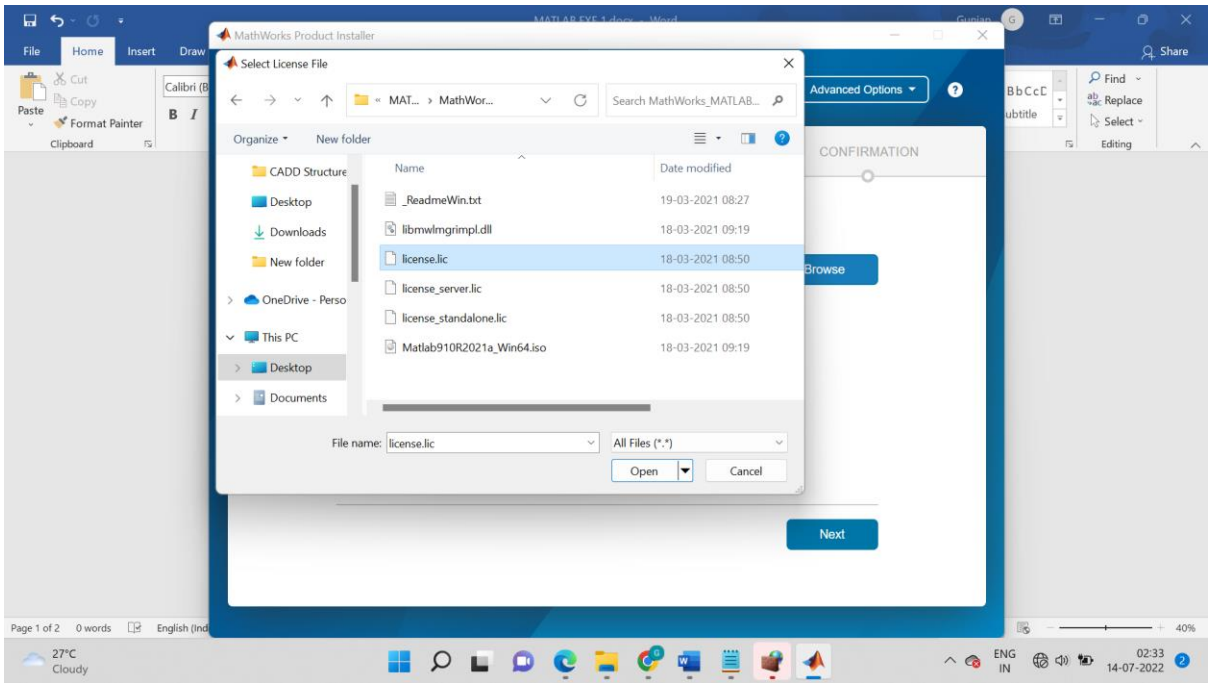

**Step 5: You have to turn off your internet and open the MathWorks Product Installer and paste the installation key:**



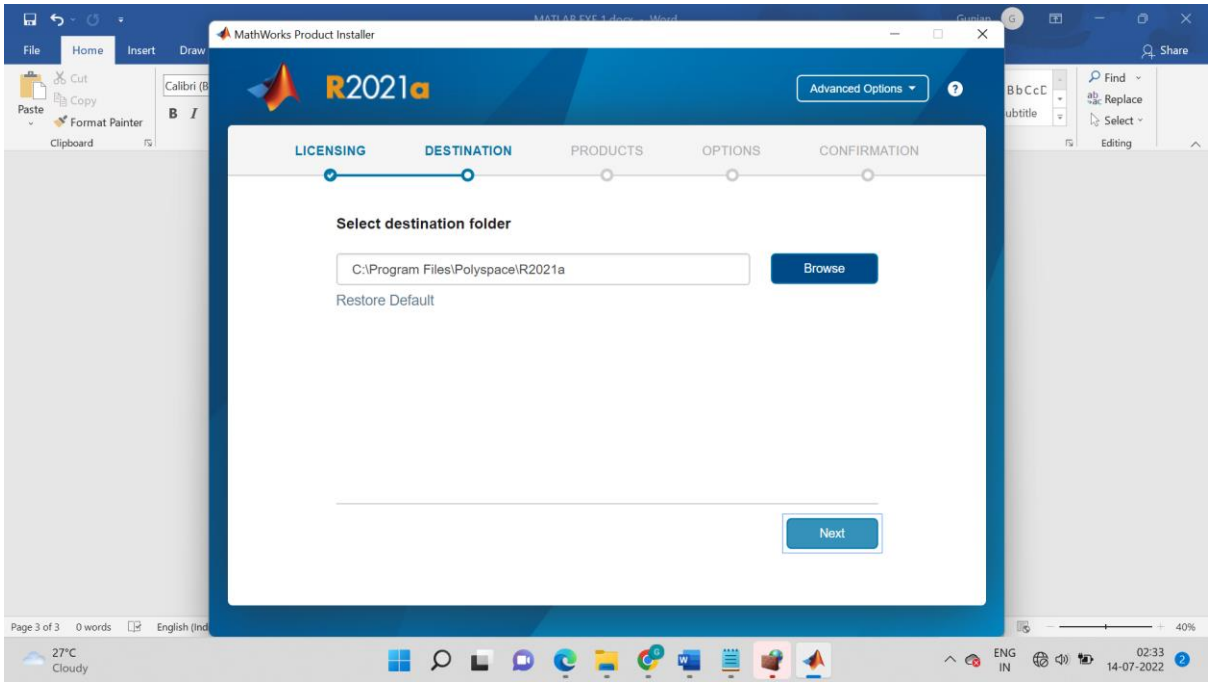
## Step 6: Now create a Mathworks account :



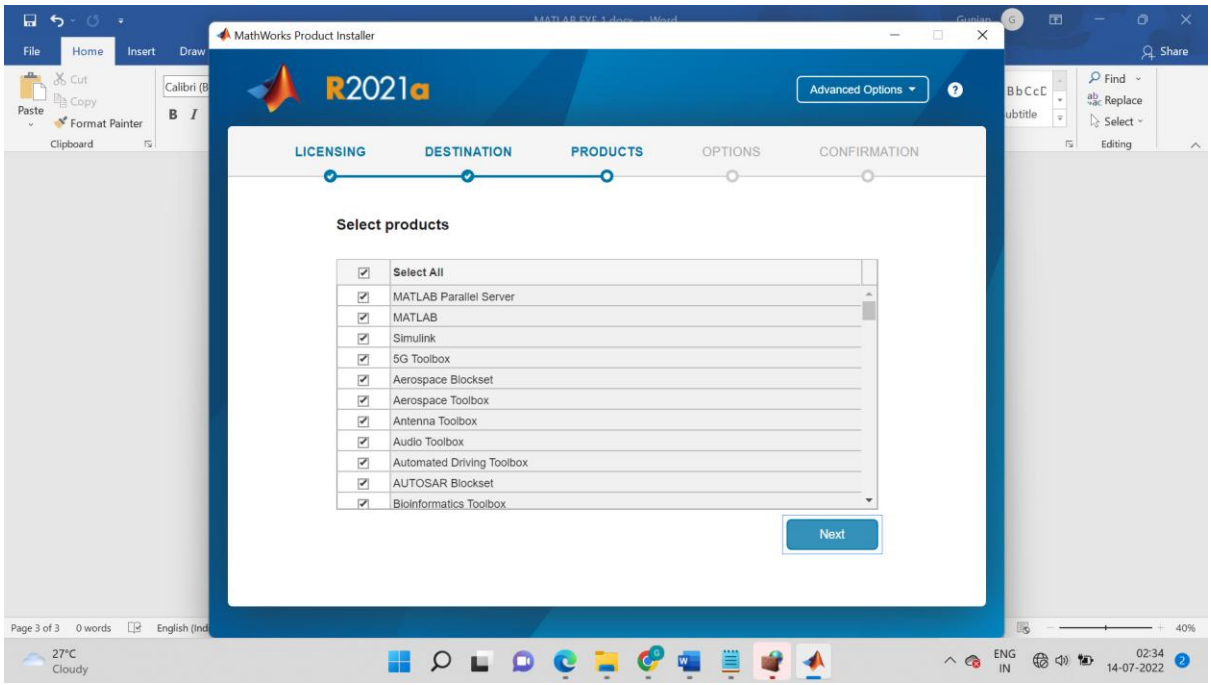

## Step 7: Now go add the license file in the account :



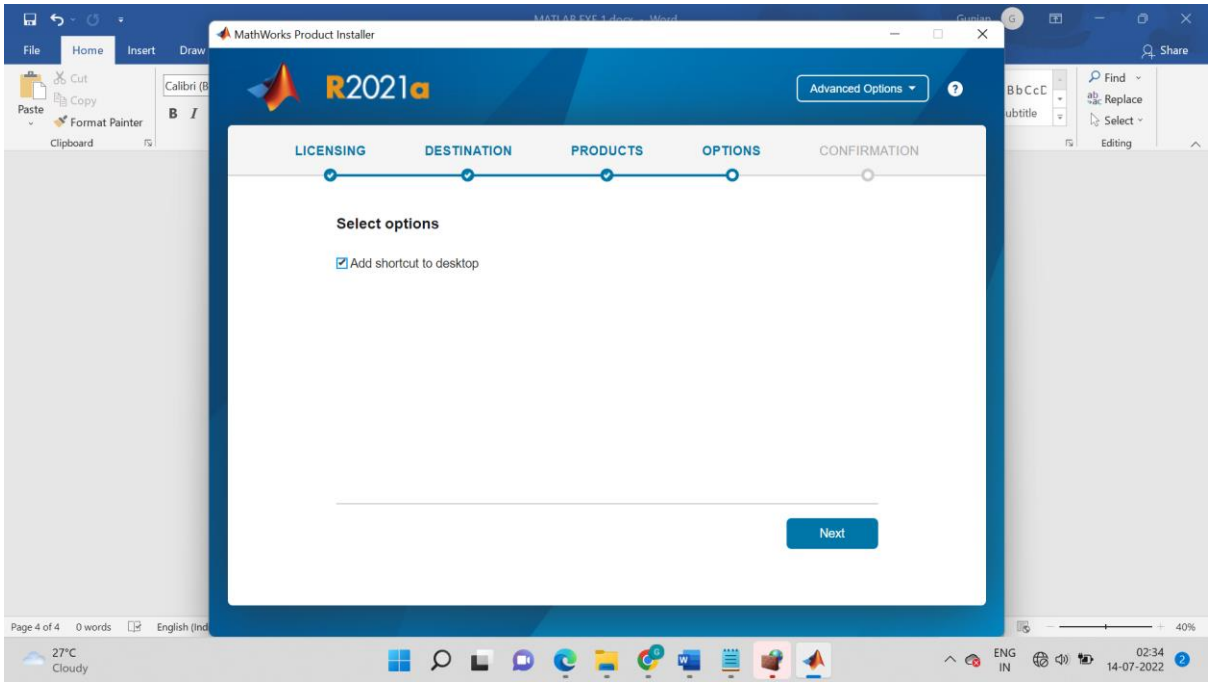
## Step 8: Copy paste your R2021a file destination:



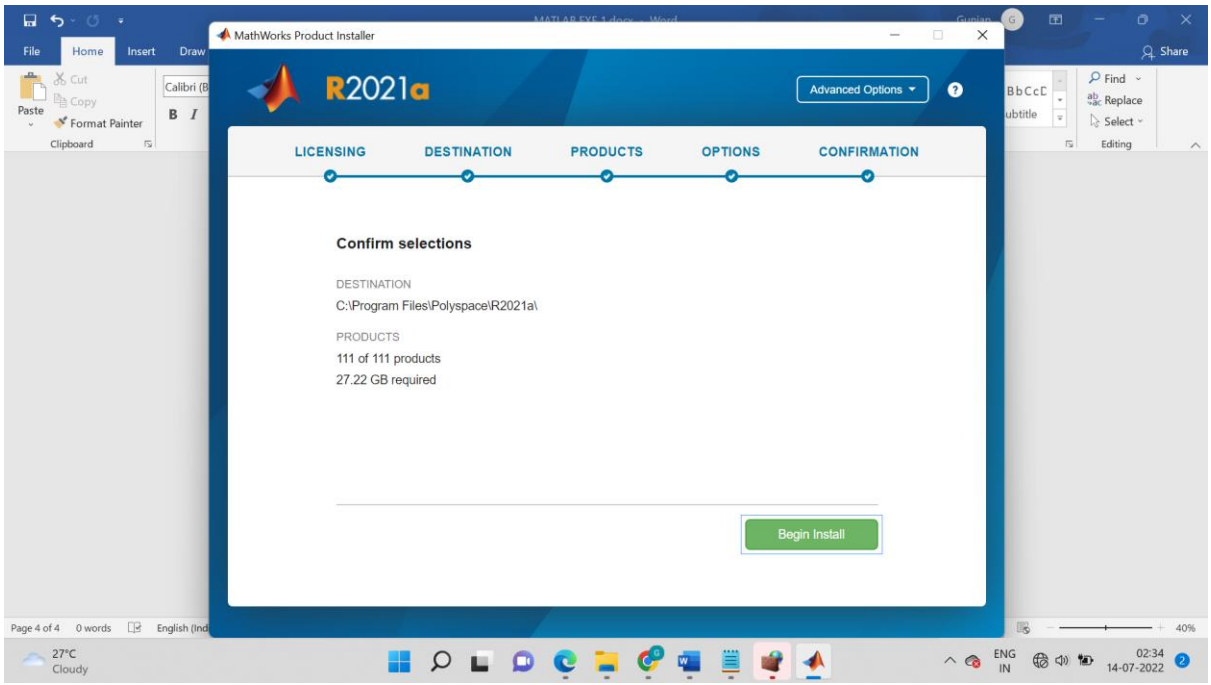

## Step 9: Now select all the products so that it gets downloaded:



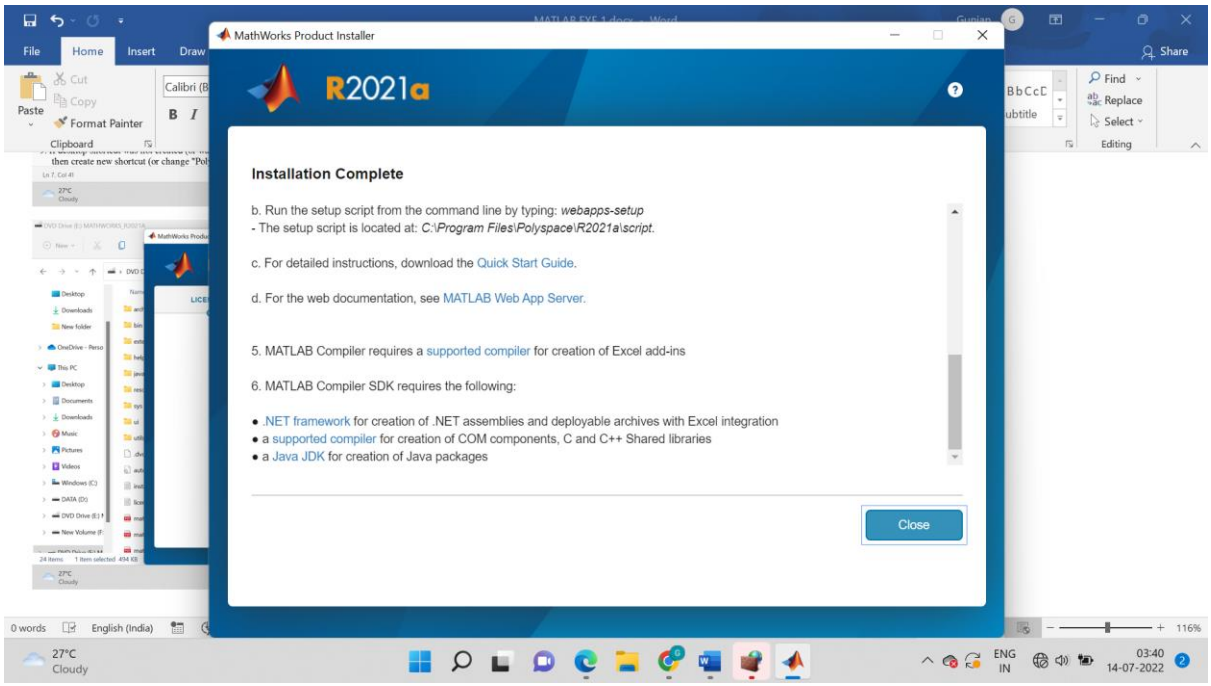
**Step 10: Tick the box so that MATLAB gets added to the desktop:**



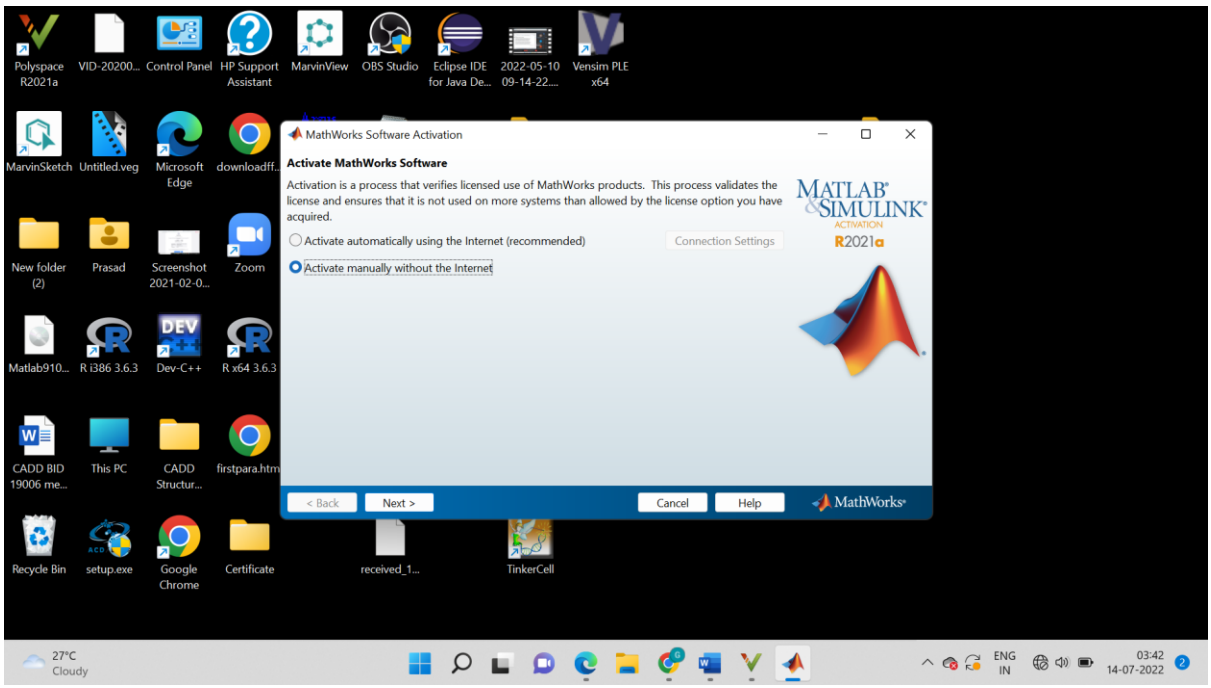

**Step 11: Click on begin installation:**

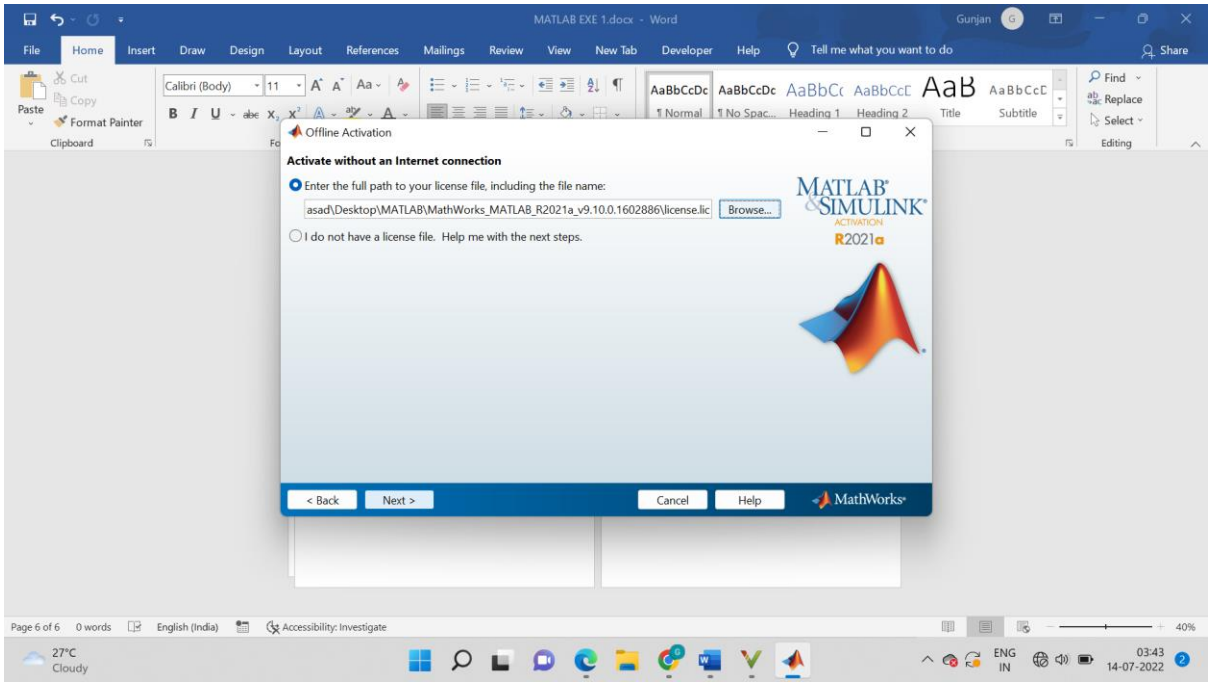


**Step 12: Now the installation is completed so click on close:**

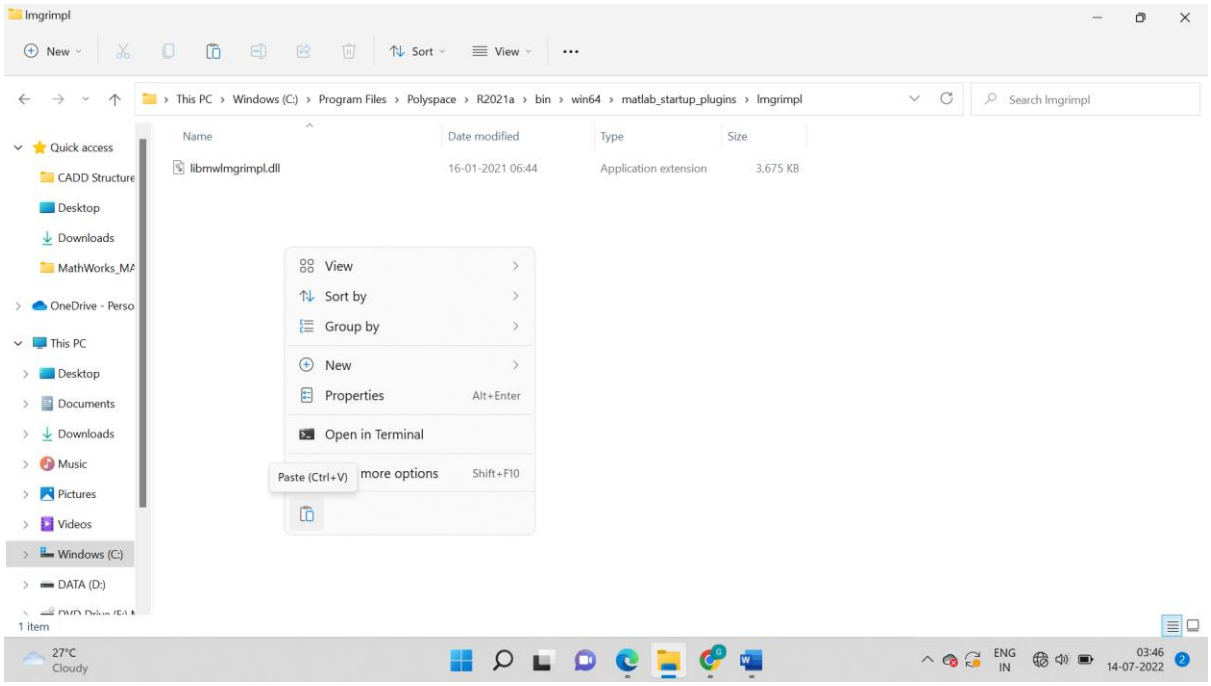


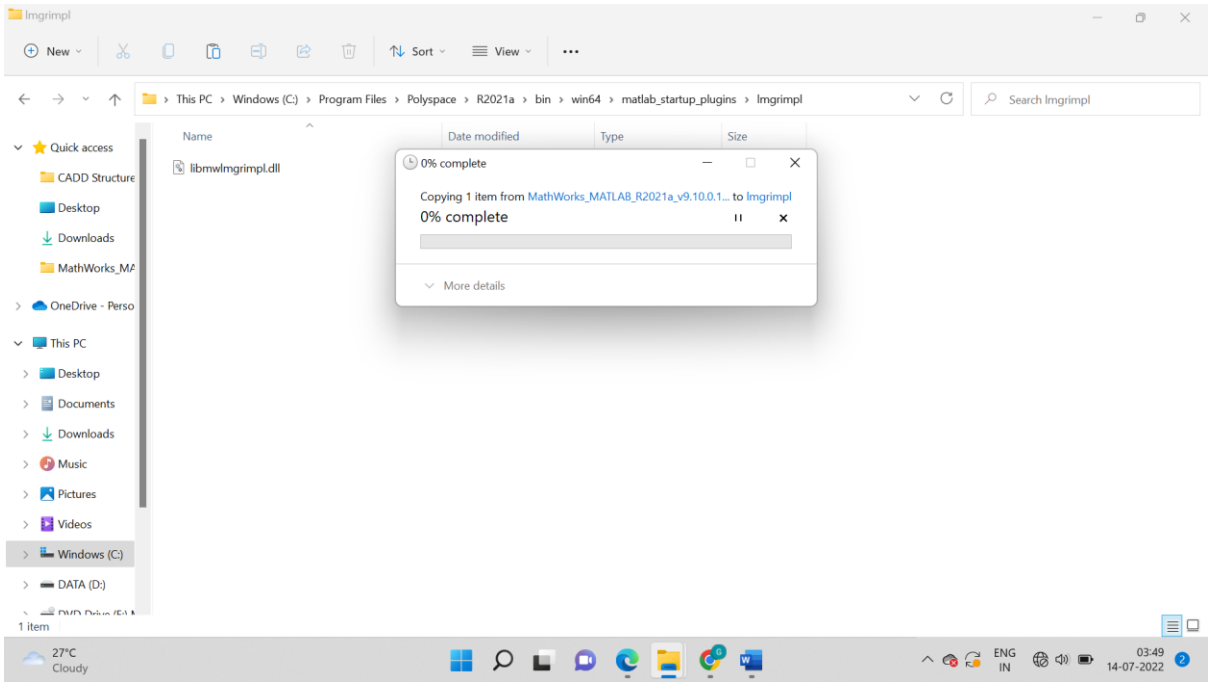

**Step 13: Click on activate manually without the internet:**

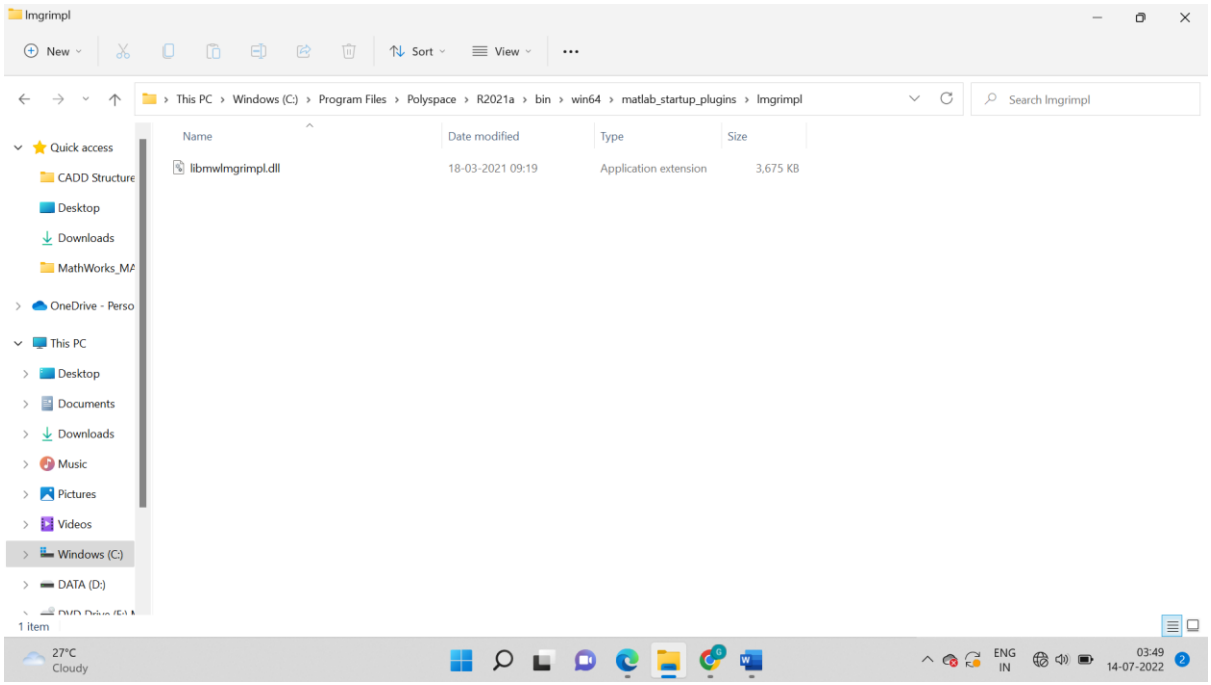


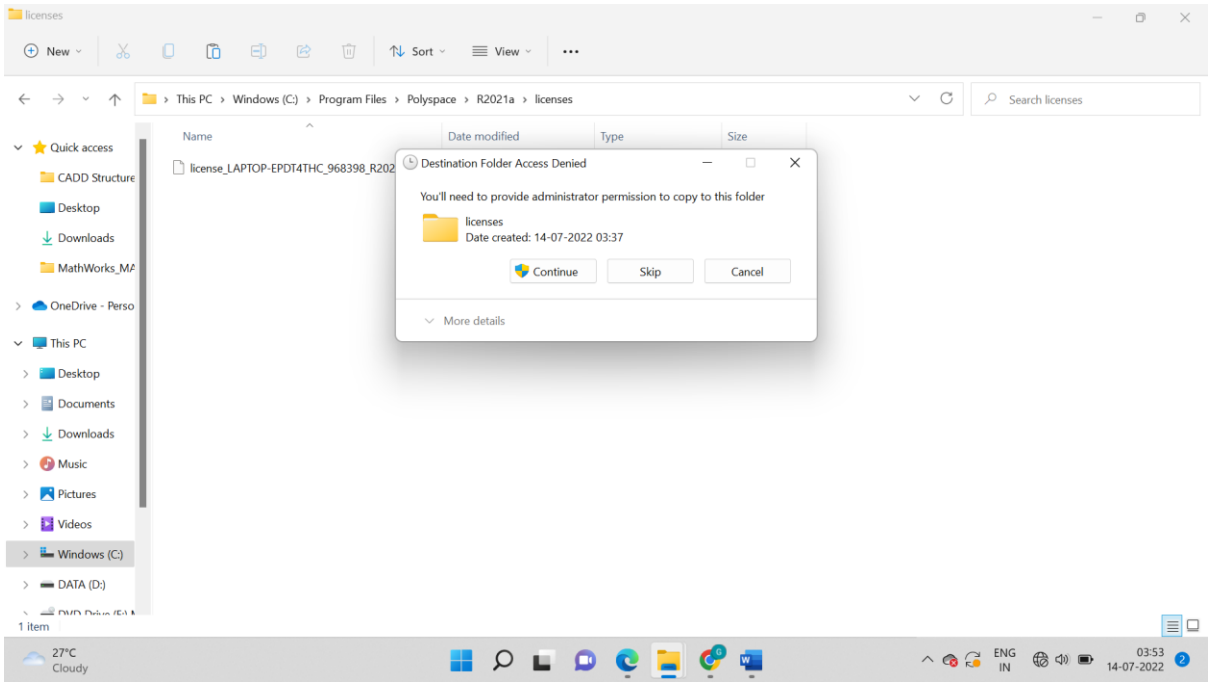


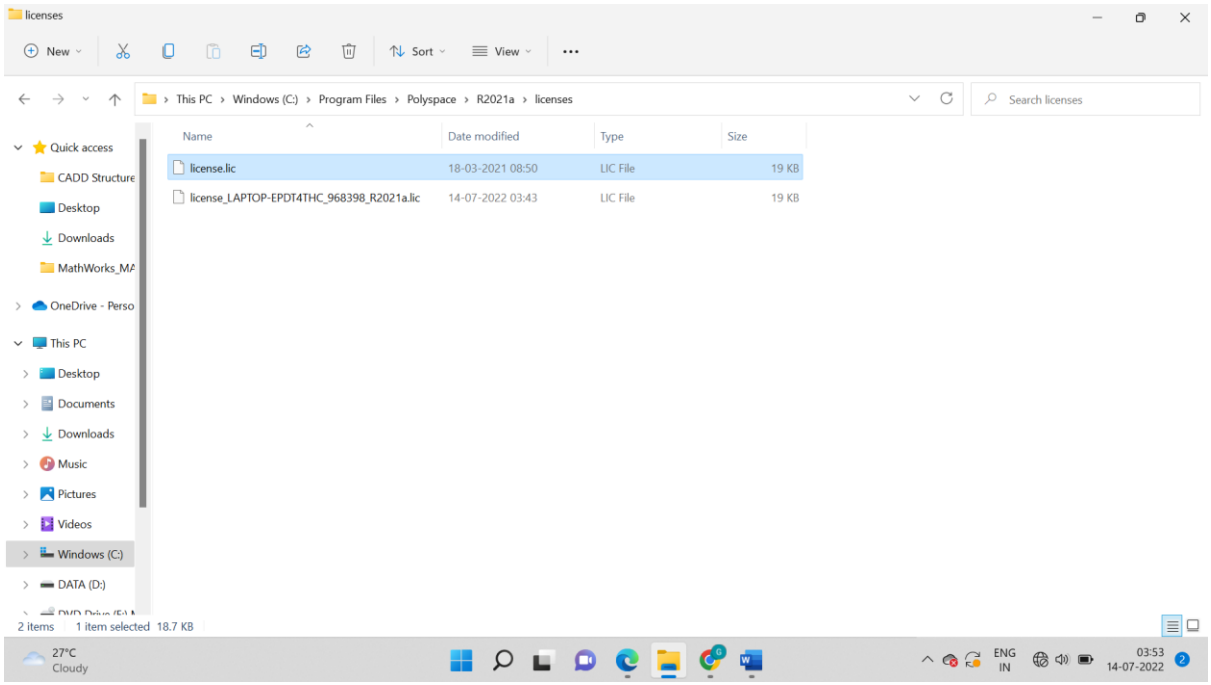

**Step 14: Go to "C:\Users\Prasad\Desktop\MATLAB\MathWorks_MATLAB_R2021a_v9.10.0.1602886\libmwlmgrimpl.dll" and paste the licenses file:**



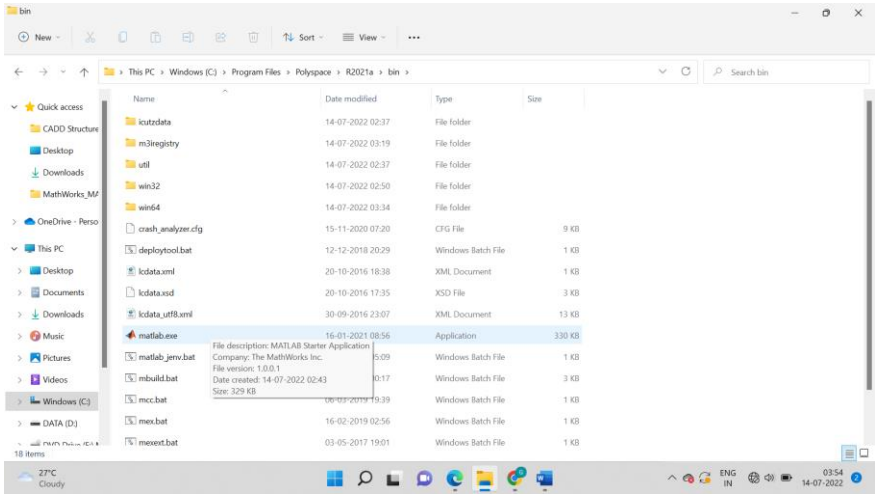

lmgrimpl
New
Sort
View
This PC › Windows (C:) › Program Files › Polyspace › R2021a › bin › win64 › matlab_startup_plugins › lmgrimpl
Search lmgrimpl
Quick access
CADD Structure
Desktop
Downloads
MathWorks_MA
OneDrive - Perso
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
DATA (D:)
Name
Date modified
Type
Size
libmwlmgrimpl.dll
0% complete
Copying 1 item from MathWorks_MATLAB_R2021a_v9.10.0.1... to lmgrimpl
0% complete
More details
1 item
27°C
Cloudy
ENG
IN
03:49
14-07-2022


lmgrimpl
New
Sort
View
This PC › Windows (C:) › Program Files › Polyspace › R2021a › bin › win64 › matlab_startup_plugins › lmgrimpl
Search lmgrimpl
Quick access
CADD Structure
Desktop
Downloads
MathWorks_MA
OneDrive - Perso
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
DATA (D:)
Name
Date modified
Type
Size
libmwlmgrimpl.dll
18-03-2021 09:19
Application extension
3,675 KB
1 item
27°C
Cloudy
ENG
IN
03:49
14-07-2022

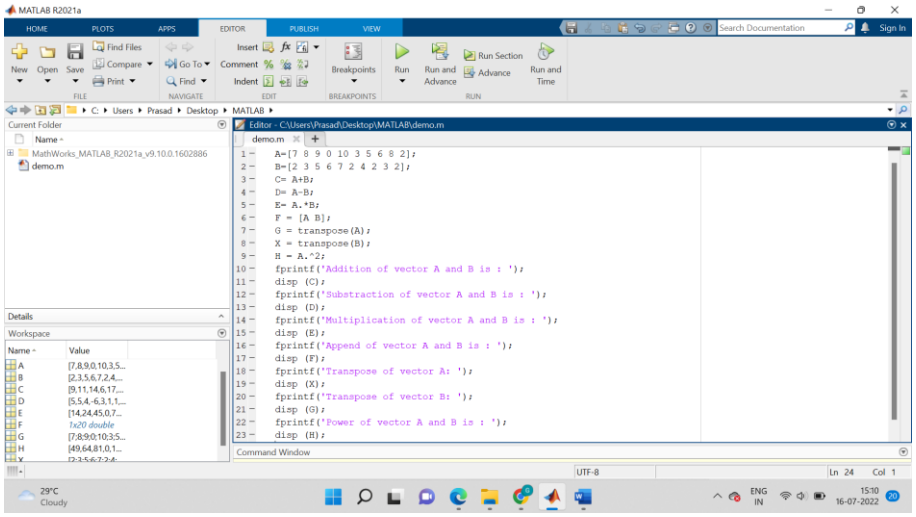

licenses
New
Sort
View
This PC › Windows (C:) › Program Files › Polyspace › R2021a › licenses
Search licenses
Quick access
CADD Structure
Desktop
Downloads
MathWorks_MA
OneDrive - Perso
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
DATA (D:)
Name
Date modified
Type
Size
license_LAPTOP-EPDT4THC_968398_R202
Destination Folder Access Denied
You'll need to provide administrator permission to copy to this folder
licenses
Date created: 14-07-2022 03:37
Continue
Skip
Cancel
More details
1 item
27°C
Cloudy
ENG
IN
03:53
14-07-2022

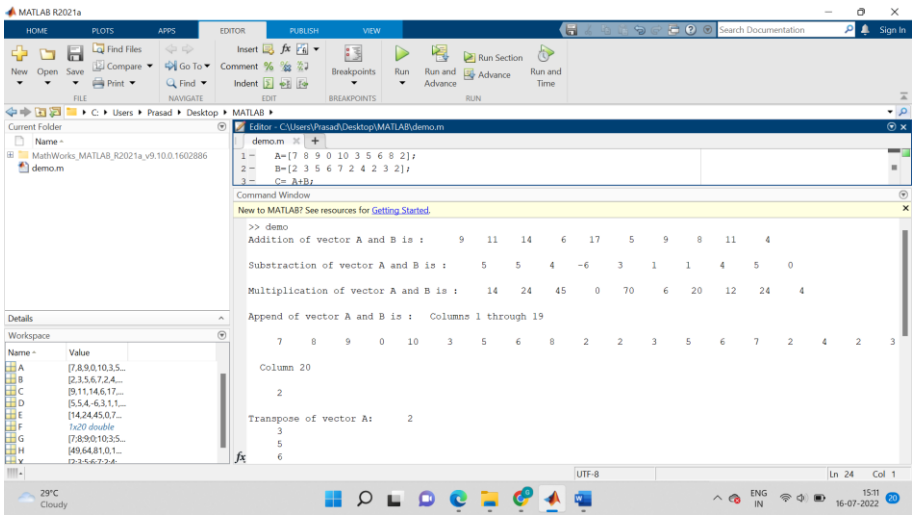

licenses
New
Sort
View
This PC › Windows (C:) › Program Files › Polyspace › R2021a › licenses
Search licenses
Quick access
CADD Structure
Desktop
Downloads
MathWorks_MA
OneDrive - Perso
This PC
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
DATA (D:)
Name
Date modified
Type
Size
license.lic
18-03-2021 08:50
LIC File
19 KB
license_LAPTOP-EPDT4THC_968398_R2021a.lic
14-07-2022 03:43
LIC File
19 KB
2 items
1 item selected 18.7 KB
27°C
Cloudy
ENG
IN
03:53
14-07-2022

**Step 15: Go to C:\Program Files\Polyspace\R2021a\bin and open the matlab.exe application:**



**Step 16: Try out some problems and save the file in .m format and run it:**





# 1-D and 2-D Arrays

## Vectors of MATLAB:

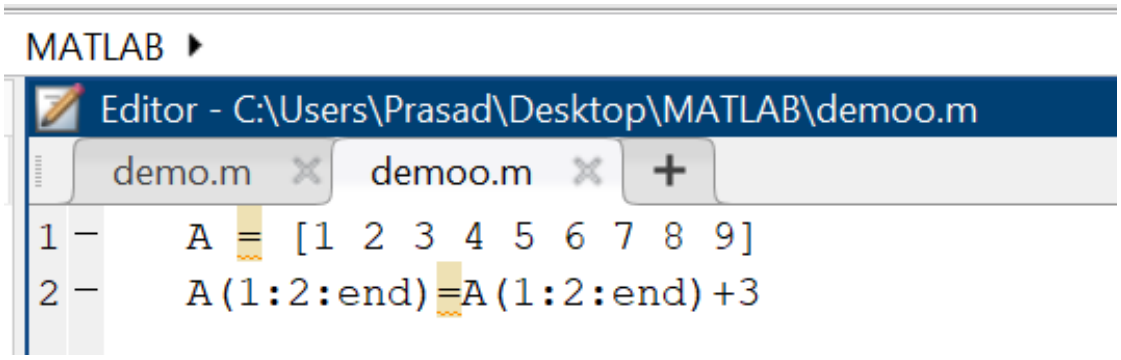
Vectors of MATLAB:

1. A = [2 5 1 6 7 3 9 10], add 3 to the odd indexed numbers.

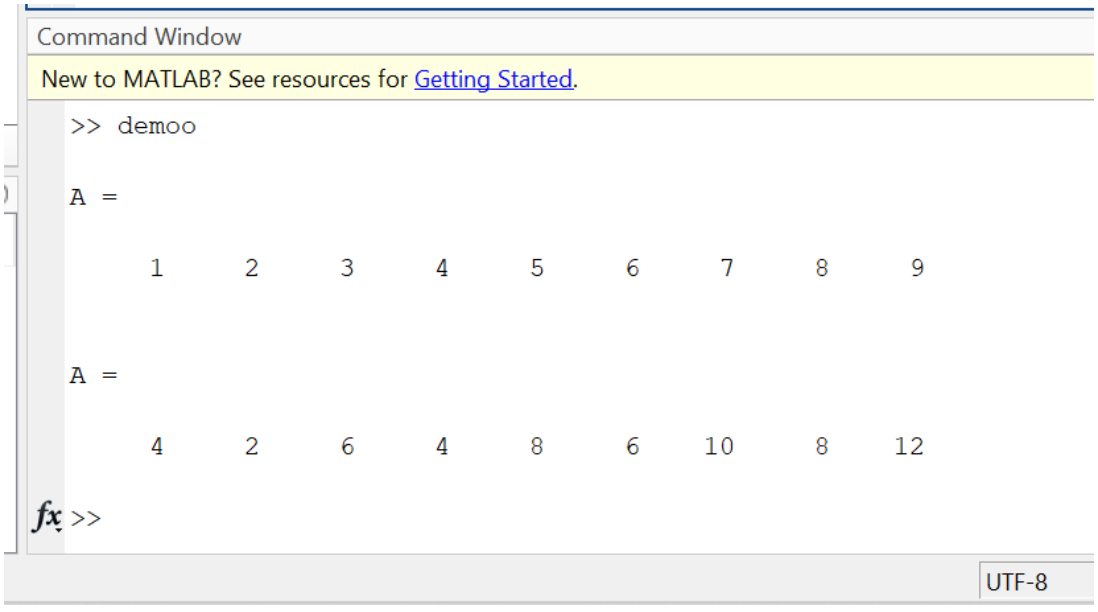
**Explanation:**

So,we have to create a row vector now, using parenthesis and colon operator access the index required that and add 3.

**Code:**




**Output:**

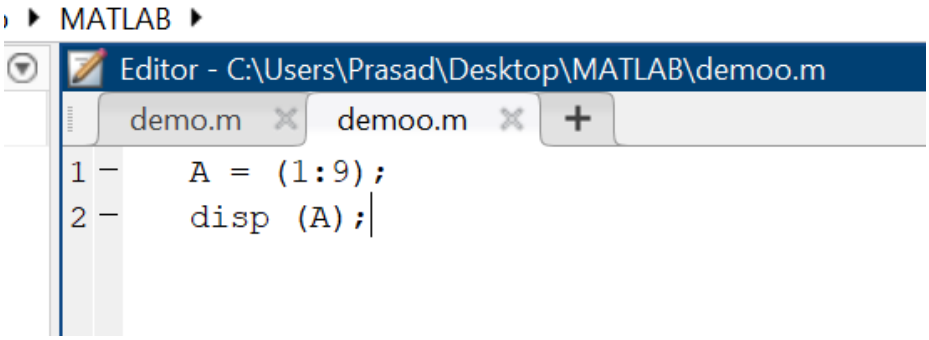


2. Create a vector of row vector of size 9.

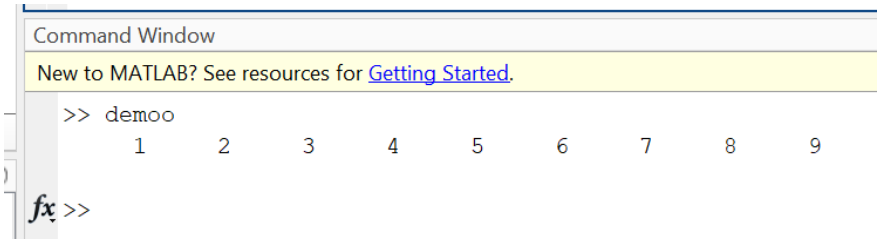
Explanation:

So,I have used colon notation and a row vector of range 1 to 9 is created.

**Code:**




**Output:**

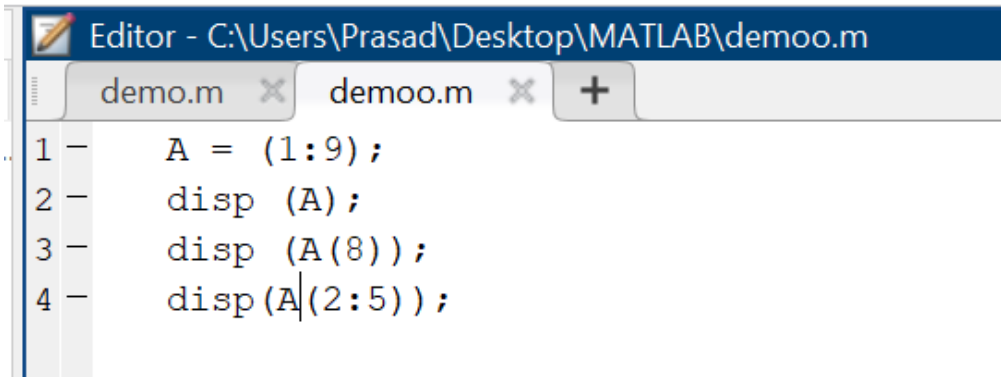


3. From the above array access the following elements:

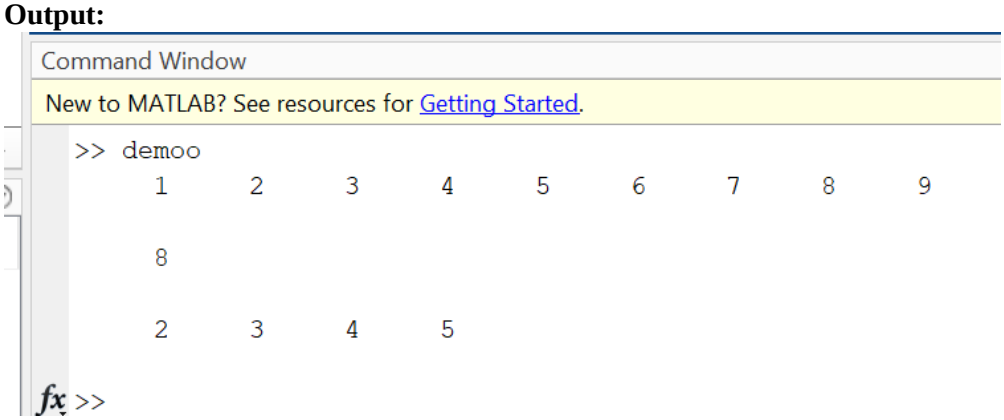
   i. Any one element of a specific position.

Explanation:
So by using colon notation I created a row vector of range 1 to 9 is created and now by using parenthesis just access the required elements.

**Code:**

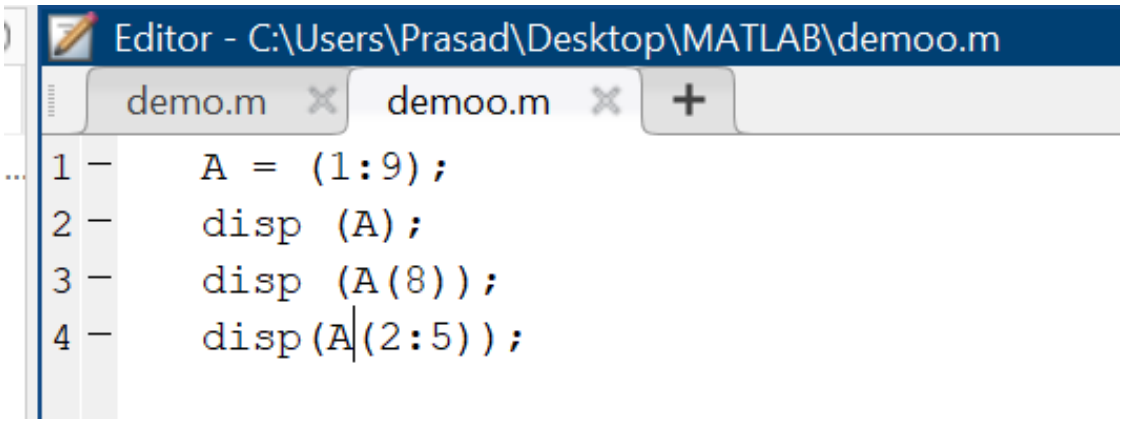



**Output:**

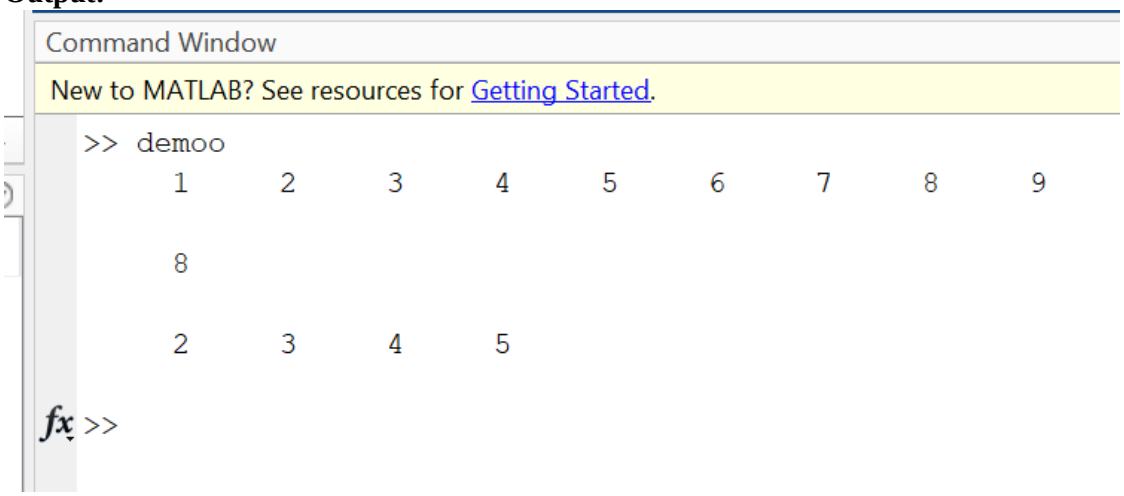


**ii. Range of any four elements.**

**Code:**




**Output:**



4. From the array from question 1. delete the any one element and determine the length and size of the array after deletion.

**Explanation:**

So by using colon notation I have created a row vector of range 1 to 9 is created now just one another vector is created excluding the element that is deleted now the function length() is used to just calculate the number of elements present in the vector.

**Code:**

MATLAB

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
1 -    A = (1:9);
2 -    disp (A);
3 -    fprintf("\nLength : %d\nVector:",length(A));
4 -    disp(A);
5 -    B=A([1:4, 6:end]);
6 -    fprintf("\ndeletion",length(B));
7 -    disp(B);
```

**Output:**

Command Window

New to MATLAB? See resources for Getting Started.

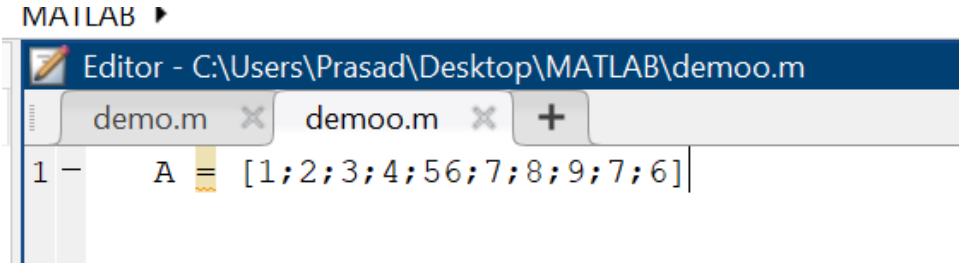
```
>> demoo
     1     2     3     4     5     6     7     8     9


Length : 9
Vector:     1     2     3     4     5     6     7     8     9


deletion     1     2     3     4     6     7     8     9

fx >>
```

5. **Create a vector of column vector of size 10.**

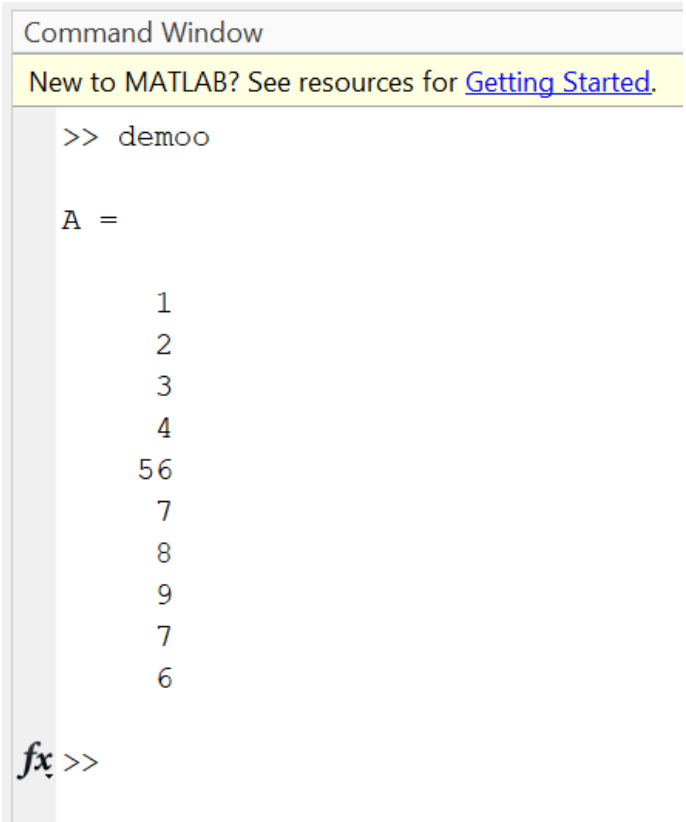
**Explanation:**

So I have created a column vector by using semicolons while initialization.

Code:




Output:

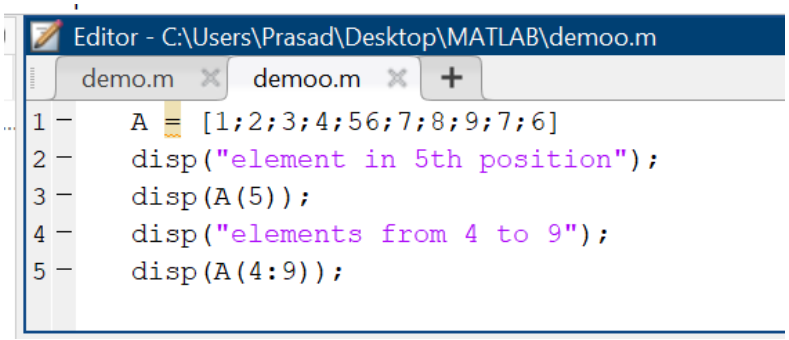


6. From the above array access the following elements:
    i. Any one element of a specific position.

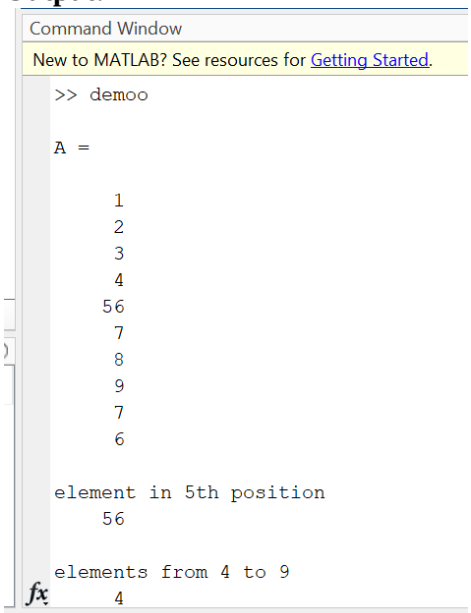
Explanation:

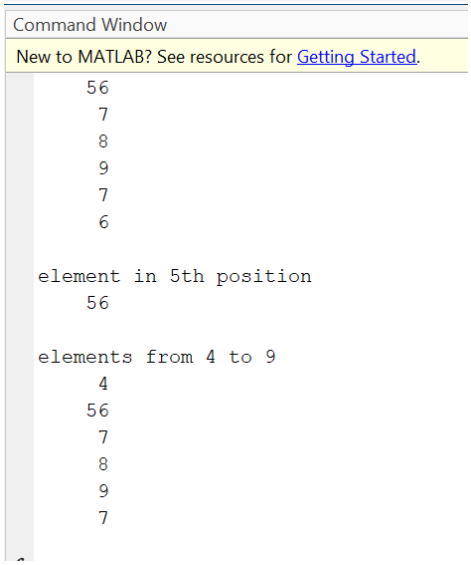
So I created a column vector by using semicolons while initialization now the parenthesis and colon notations that is the range operator is used to access the needed elements.

**Code:**




**Output:**



**ii. Range of any four elements.**

**Code:**

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
A = [1;2;3;4;56;7;8;9;7;6]
disp("element in 5th position");
disp(A(5));
disp("elements from 4 to 9");
disp(A(4:9));
```

**Output:**

New to MATLAB? See resources for Getting Started.

```
    56
     7
     8
     9
     7
     6

element in 5th position
    56

elements from 4 to 9
     4
    56
     7
     8
     9
     7

fx >>
```

7. From the array from question 4. delete the any one element and determine the length and size of the array after deletion.

**Explanation:**

So, I have created a column vector by using semicolons and just while I was doing initialization. another vector is created that is excluding the element to be deleted now the function length () is used which calculates the number of elements present in the vector.

**Code:**

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

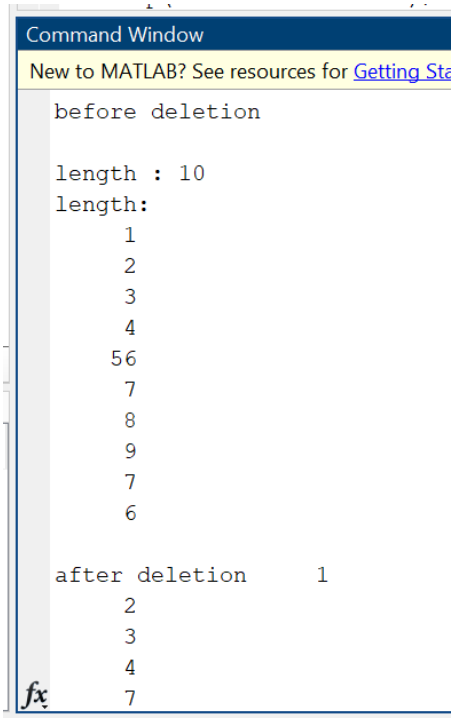
```
A = [1;2;3;4;56;7;8;9;7;6]
disp("before deletion");
fprintf("\nlength : %d\nlength:\n",length(A));
disp(A);
B = A([1:4, 6:end]);
fprintf("after deletion",length(B));
disp(B);
```

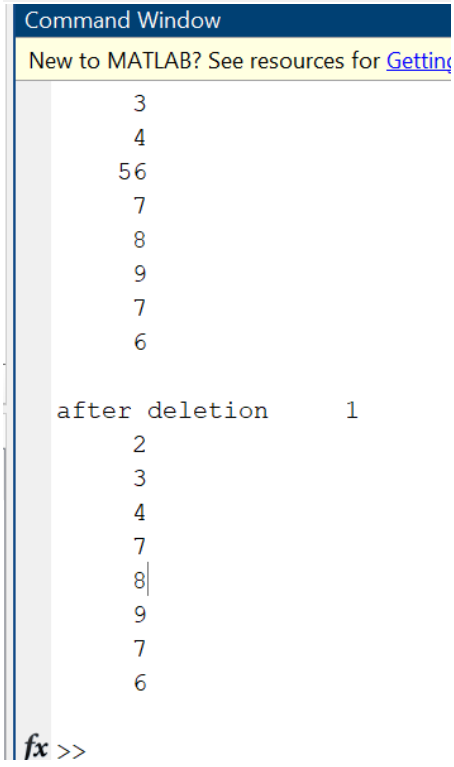
**Output:**

Command Window

New to MATLAB? See resources for

```
>> demoo

A =

     1
     2
     3
     4
    56
     7
     8
     9
     7
     6

before deletion

length : 10
length:
     1
```


Command Window
New to MATLAB? See resources for Getting Sta
before deletion
length : 10
length:
1
2
3
4
56
7
8
9
7
6
after deletion 1
2
3
4
7


Command Window
New to MATLAB? See resources for Gettinc
3
4
56
7
8
9
7
6
after deletion 1
2
3
4
7
8
9
7
6
>>

8. A = [2 5 1 6 7 3 9 10 11 15 14], add 3 to the odd indexed numbers.

Explanation:

So, we have to create a row vector by using parenthesis and colon operator we can access the index required and add 3 as shown:

**Code:**

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
A = [2  5  1  6  7  3  9  10  11  15  14];
disp("vector before:")
disp(A);
A(1:2:end)=A(1:2:end)+3;
disp("After addition of 3");
disp(A)
```

**Output:**

Command Window

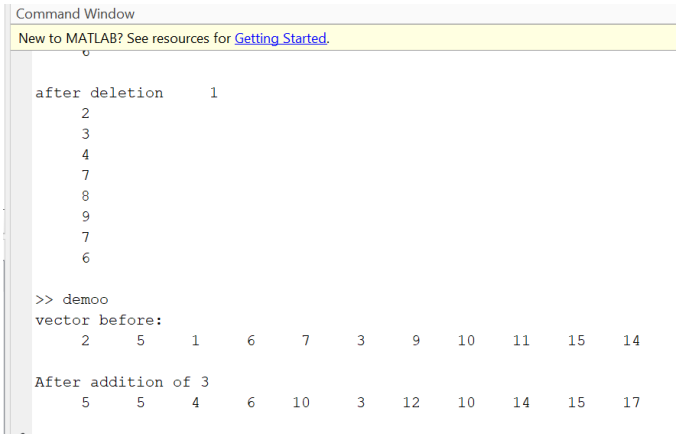
New to MATLAB? See resources for Getting Started.

```
>> demoo

A =

     1
     2
     3
     4
    56
     7
     8
     9
     7
     6

before deletion

length : 10
length:
     1
     2
```

Command Window

New to MATLAB? See resources for Getting Started

```
before deletion

length : 10
length:
     1
     2
     3
     4
    56
     7
     8
     9
     7
     6

after deletion     1
     2
     3
     4
```

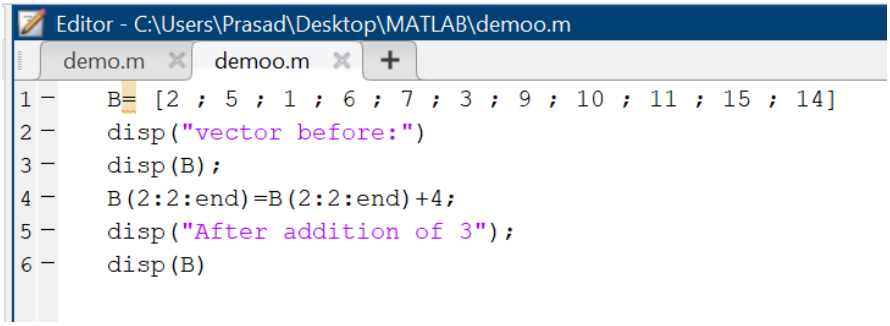

Command Window
New to MATLAB? See resources for Getting Started.
after deletion 1
2
3
4
7
8
9
7
6
>> demoo
vector before:
2 5 1 6 7 3 9 10 11 15 14
After addition of 3
5 5 4 6 10 3 12 10 14 15 17

9. B= [2 ; 5 ; 1 ; 6 ; 7 ; 3 ; 9 ; 10 ; 11 ; 15 ; 14], add 4 to the even indexed numbers.

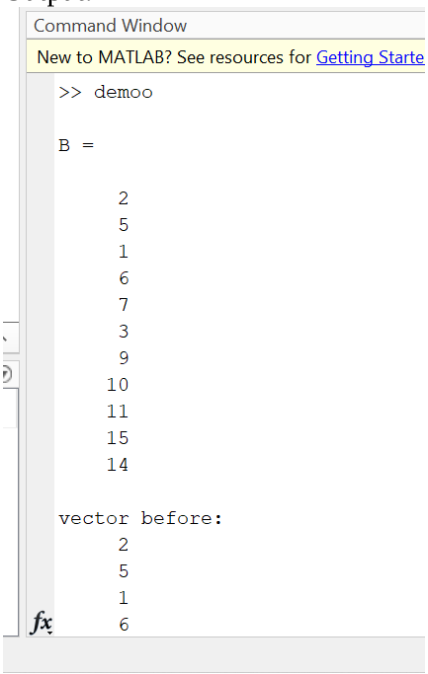
Explanation:

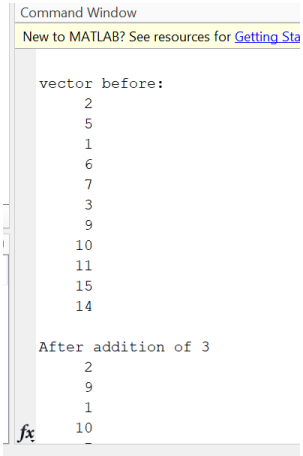
So I have created a row vector by just using parenthesis and colon operator that access the index required and add 4.

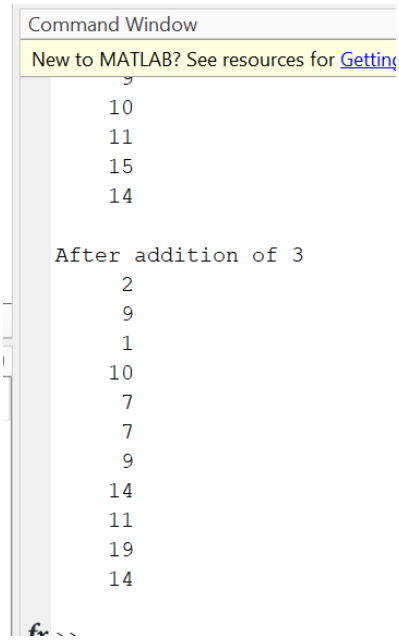
Code:




Output:



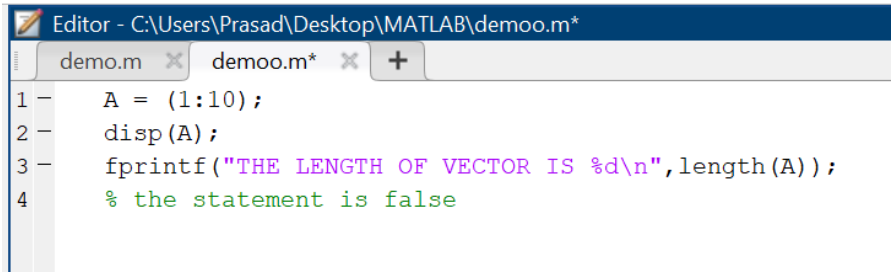

Command Window
New to MATLAB? See resources for Getting
10
11
15
14
After addition of 3
2
9
1
10
7
7
9
14
11
19
14

10. If the vector is of size 1 X 10, then the length of the vector will be more than 10. Justify whether the statement is true or false with the help of an example.

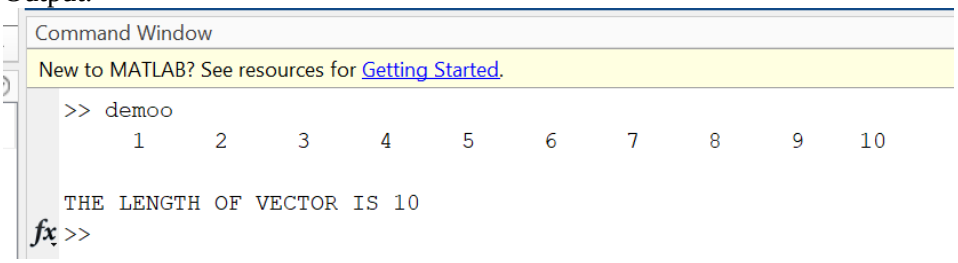
Explanation:

So I have created a row vector from range 1 to 10 As we know a range function take both the start and end point into consideration, so a vector of length no less that 10 is created.

Code:




Output:



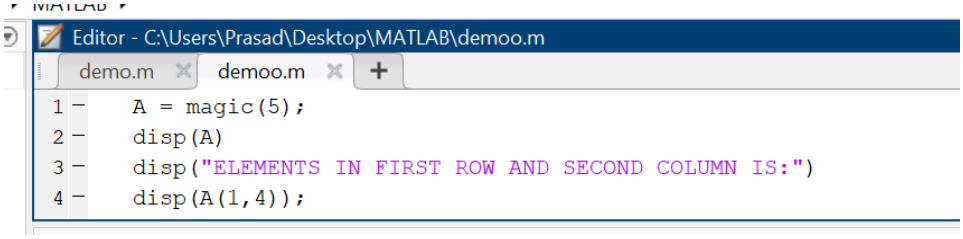
**Matrices of MATLAB:**

1. Create a matrix of size 5.

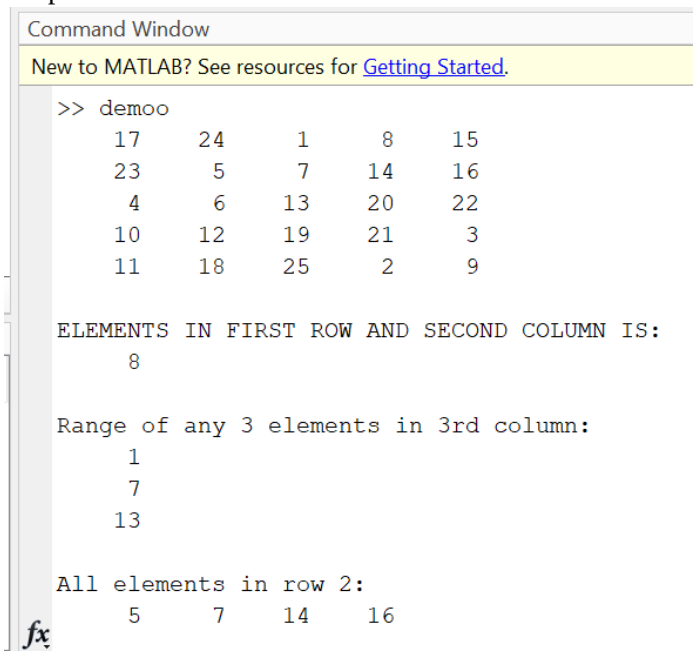
Explanation:

A magic matrix is created that is of dimension 5x5.

Code:




Output:



2. From the above array access the following elements:
    i. Any one element of a specific position.
    ii. Range of any four elements in a row.
    iii. Range of any three elements in a column.
    iv. All the elements of two rows.
    v. All the elements of any 4 columns.
    vi. All the elements in a column vector.

Explanation:

So, I have created a magic matrix that is of dimension 5x5 also , we have parenthesis and range operator ( : ) that we can use to access the required elements.

Code:

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
A = magic(5);
disp(A)
disp("ELEMENTS IN FIRST ROW AND SECOND COLUMN IS:")
disp(A(1,4));
disp("Range of any 3 elements in 3rd column:");
disp(A(1:3,3));
disp("All elements in row 2:");
disp(A(2,2:5));
disp("all elements of 4 columns ");
disp(A(2,:));
disp("ALL ELEMENTS OF COLUMN VECTOR:");
disp(A(:));
```

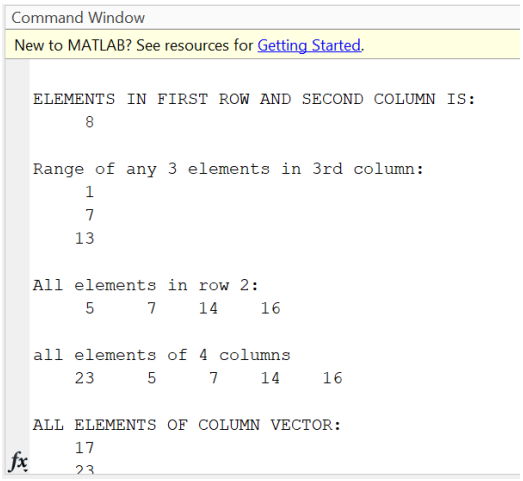
Output:

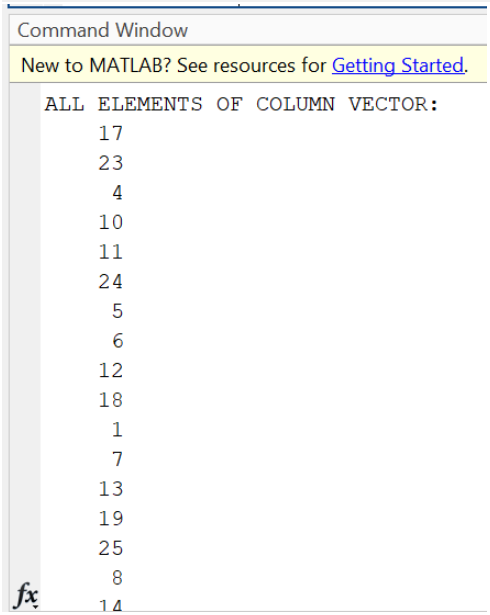
Command Window

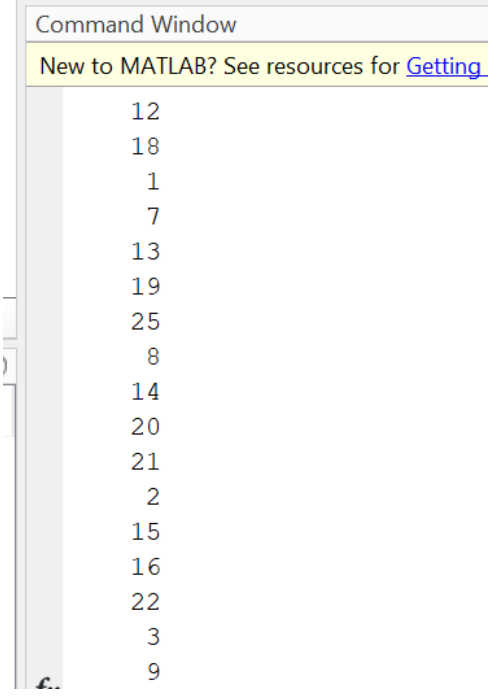
New to MATLAB? See resources for Getting Started.

```
>> demoo
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

ELEMENTS IN FIRST ROW AND SECOND COLUMN IS:
     8

Range of any 3 elements in 3rd column:
     1
     7
    13

All elements in row 2:
     5     7    14    16
```

fx


Command Window
New to MATLAB? See resources for Getting Started.
ELEMENTS IN FIRST ROW AND SECOND COLUMN IS:
8
Range of any 3 elements in 3rd column:
1
7
13
All elements in row 2:
5 7 14 16
all elements of 4 columns
23 5 7 14 16
ALL ELEMENTS OF COLUMN VECTOR:
17
23


Command Window
New to MATLAB? See resources for Getting Started.
ALL ELEMENTS OF COLUMN VECTOR:
17
23
4
10
11
24
5
6
12
18
1
7
13
19
25
8
14


Command Window
New to MATLAB? See resources for Getting
12
18
1
7
13
19
25
8
14
20
21
2
15
16
22
3
9

3. From the array from question 1. delete the any one column and determine the length and size of the array after deletion.

Explanation:

So, I have created a matrix of dimension 5x5 now, parenthesis and range operator (:) is used that will access the required element and declared it as empty.

Code:

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
X= magic(5);
disp (X);
disp ("Before deletion:");
disp (size (X));
disp (length (X));
X(:,3)=[]
disp("After deleting coloumn 3 now the size and length are:")
disp (size (X));
disp (length (X));
```

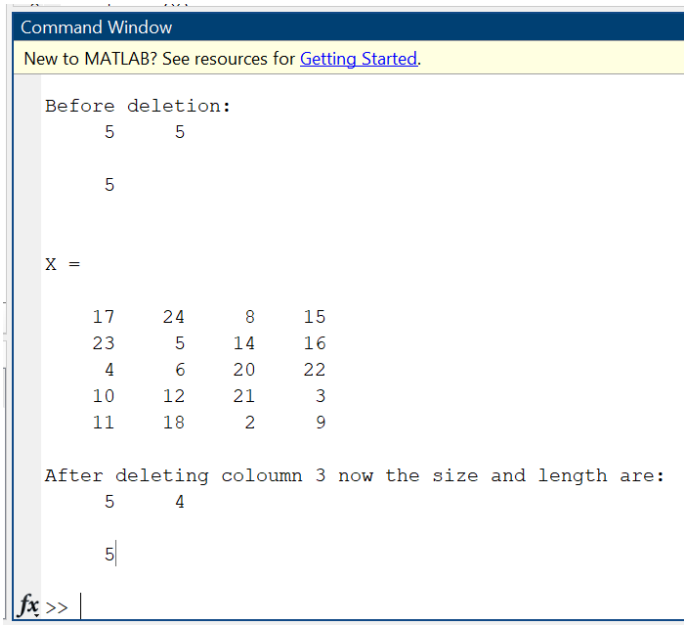
Command Window

Output:

Command Window

New to MATLAB? See resources for Getting Started.

```
>> demoo
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

Before deletion:
     5     5

     5


X =

    17    24     8    15
    23     5    14    16
     4     6    20    22
    10    12    21     3
    11    18     2     9
```


Command Window
New to MATLAB? See resources for Getting Started.
Before deletion:
5 5
5
X =
17 24 8 15
23 5 14 16
4 6 20 22
10 12 21 3
11 18 2 9
After deleting coloumn 3 now the size and length are:
5 4
5
fx >>

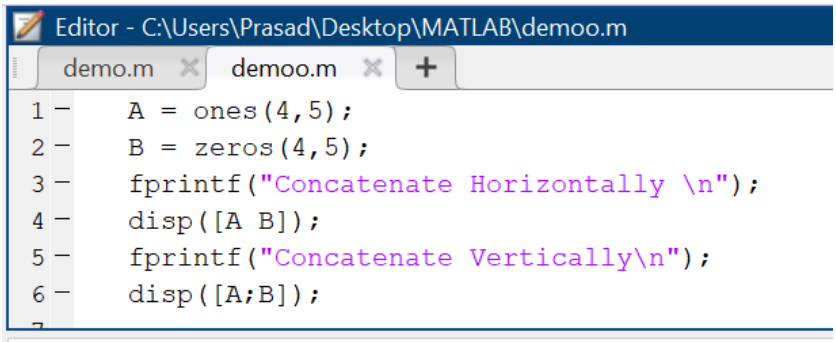
4. Create a 2-D array of size of 4 X 5 to perform the following operations:

Explanation:

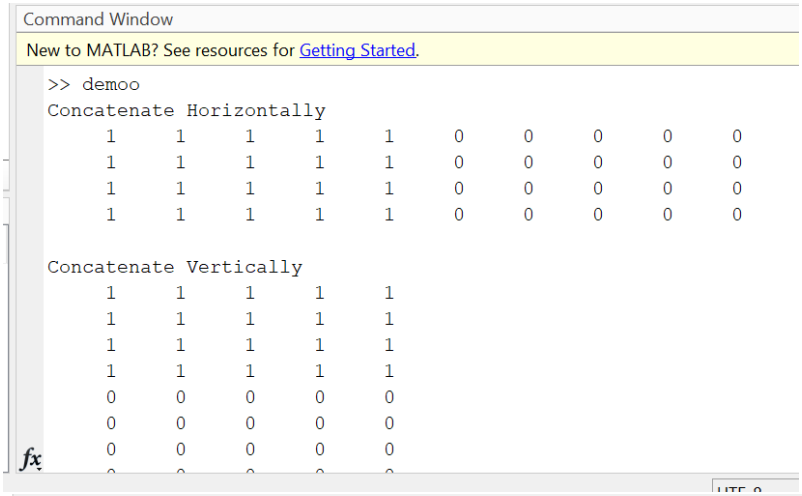
So I have created two matrices and concatenated accordingly by using [ ].

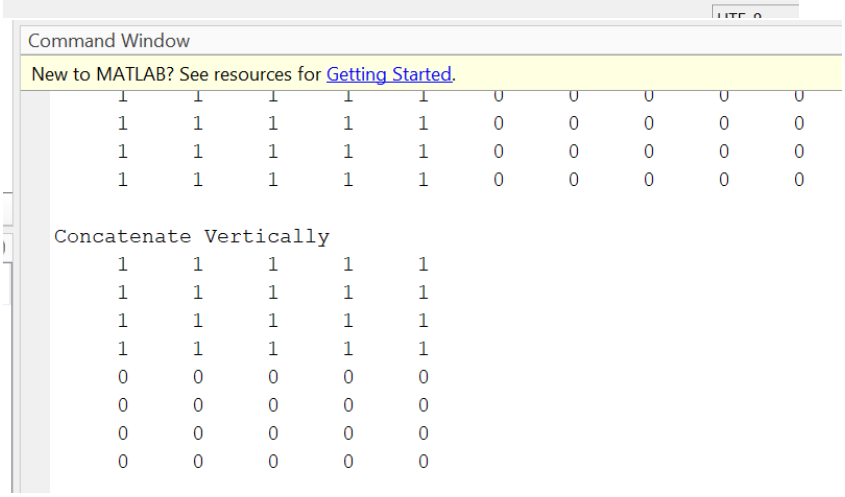
i. Concatenate Horizontally
ii. Concatenate Vertically

Code:




Output:



# Multi-Dimensional Arrays and Colon Notation

## Arrays of MATLAB:

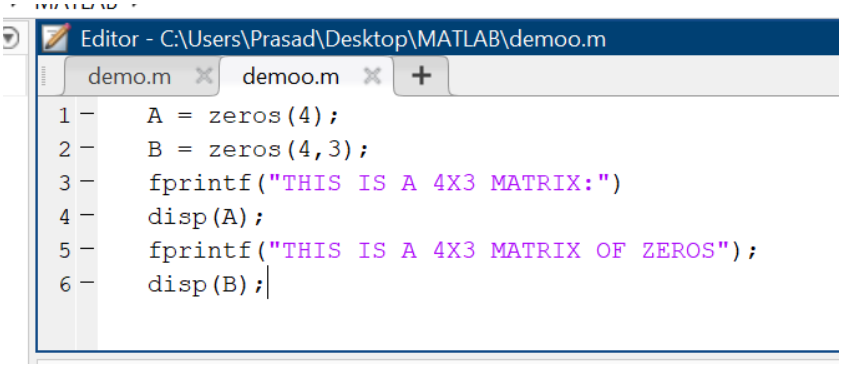
1. Create the following 2-D arrays.

Explanation:

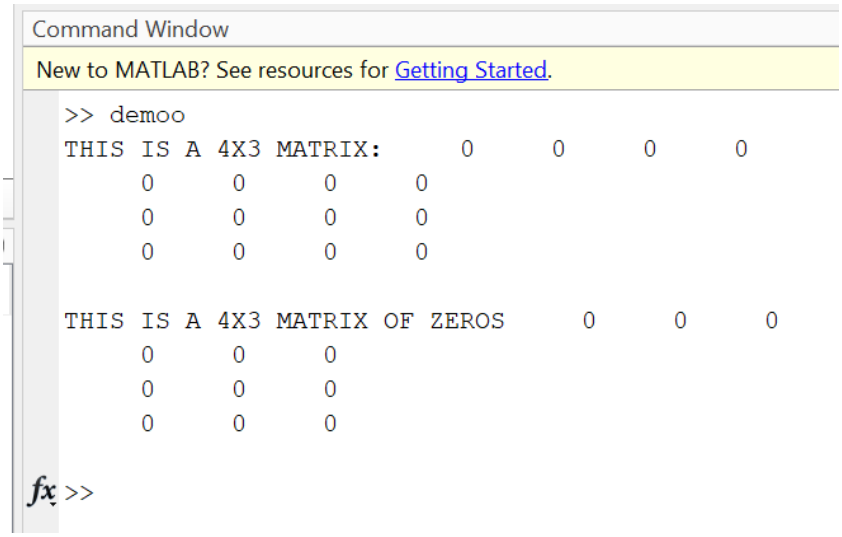
All these functions are used to create the 2D arrays.

a. Arrays containing 0s as elements of size 4 and 4X3.

Code:




Output:



b. Arrays containing 1s as elements of size 4 and 4X3.

Code:

esktop ▸ MATLAB ▸

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
1 -    A = ones(4);
2 -    B = ones(4,3);
3 -    fprintf("THIS IS A 4X4 MATRIX:")
4 -    disp(A);
5 -    fprintf("THIS IS A 4X3 MATRIX OF ONES");
6 -    disp(B);
```

Output:

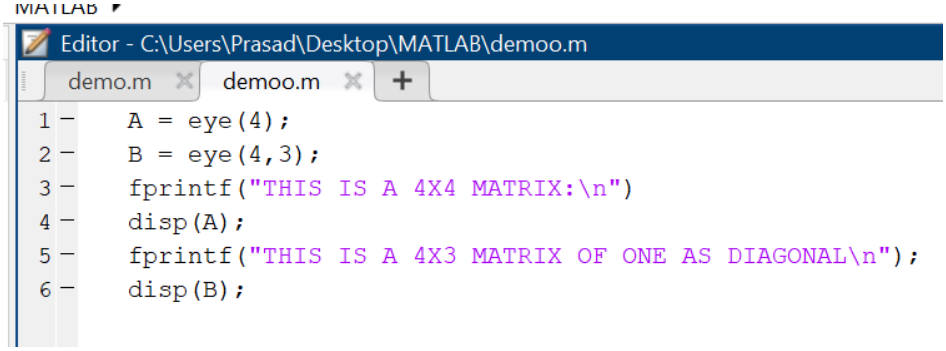
Command Window

New to MATLAB? See resources for Getting Started.

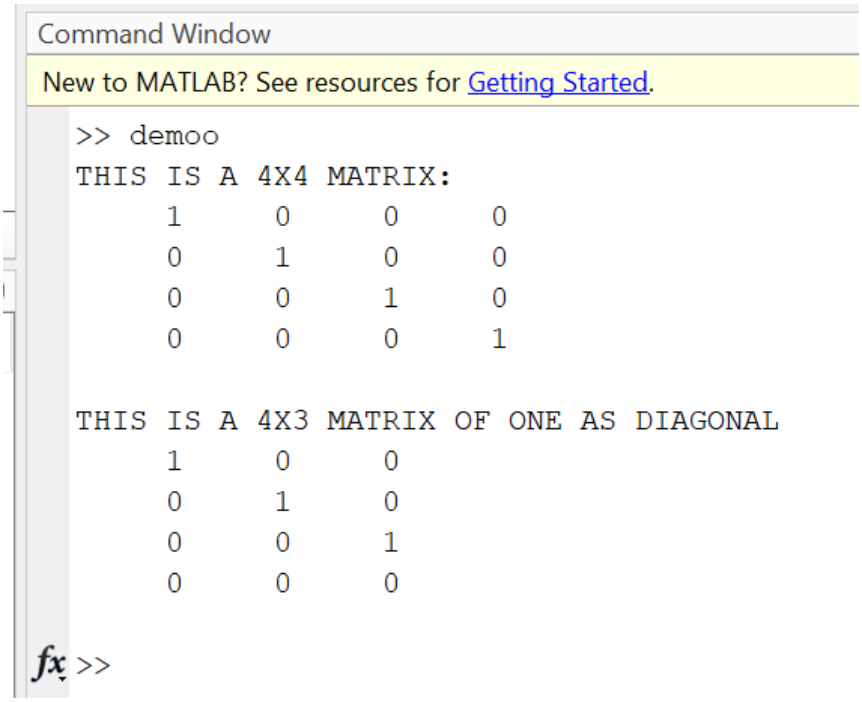
```
>> demoo
THIS IS A 4X4 MATRIX:     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1

THIS IS A 4X3 MATRIX OF ONES     1     1     1
     1     1     1
     1     1     1
     1     1     1

fx >>
```

c. Arrays containing 1s (as diagonal elements) and 0s as elements of size 4 and 4X3.

Code:

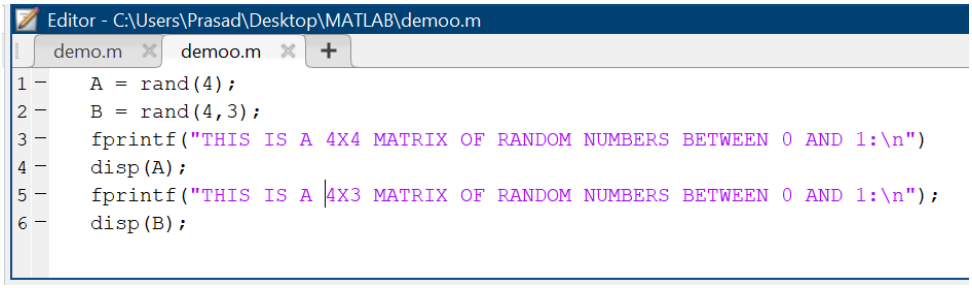



Output:

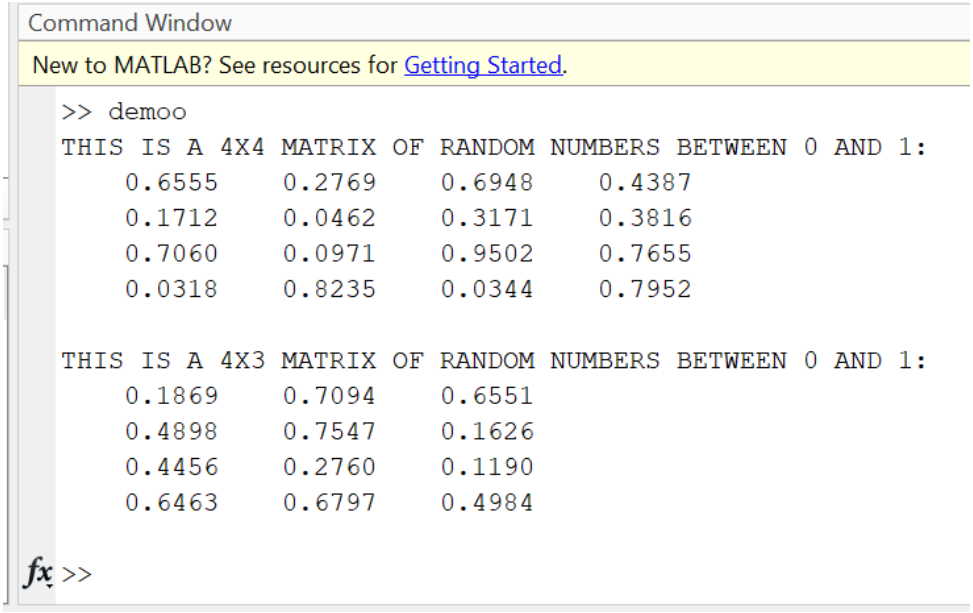


d. Array containing random elements from 0-1 of size 4 and 4X3.

Code:




Output:

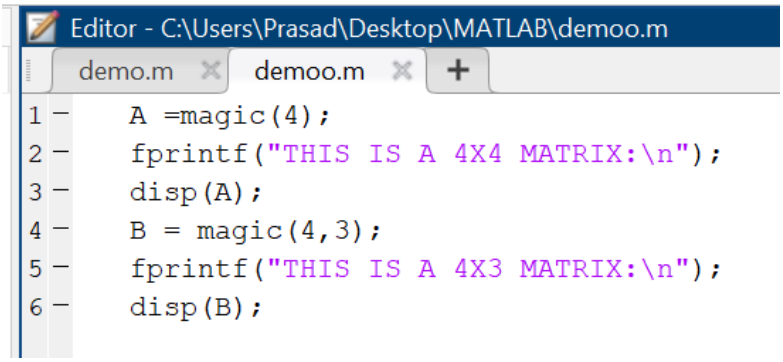


e. Arrays using magic () of size 4 and 4X3.

Magic matrix:

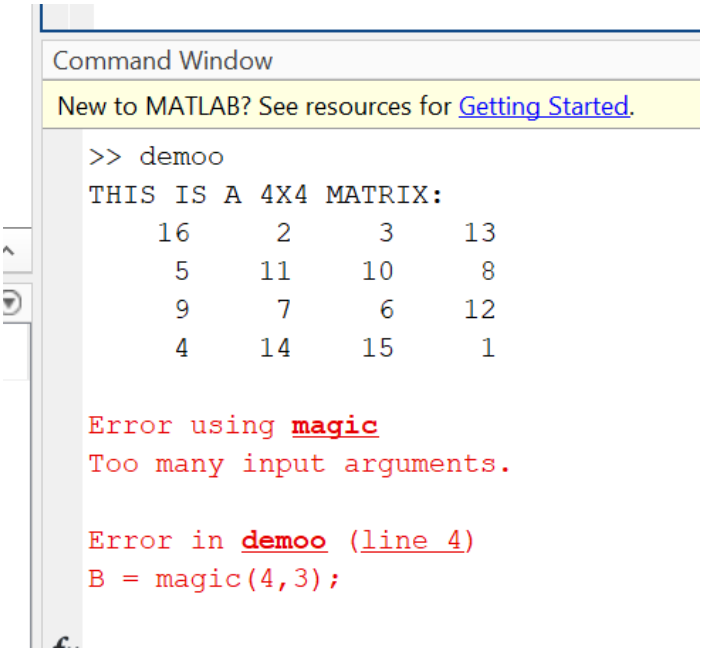
Magic matrix will work only when the number of rows and columns are same.

Code:




Output:

Magic matrix will work only when the number of rows and columns are same.



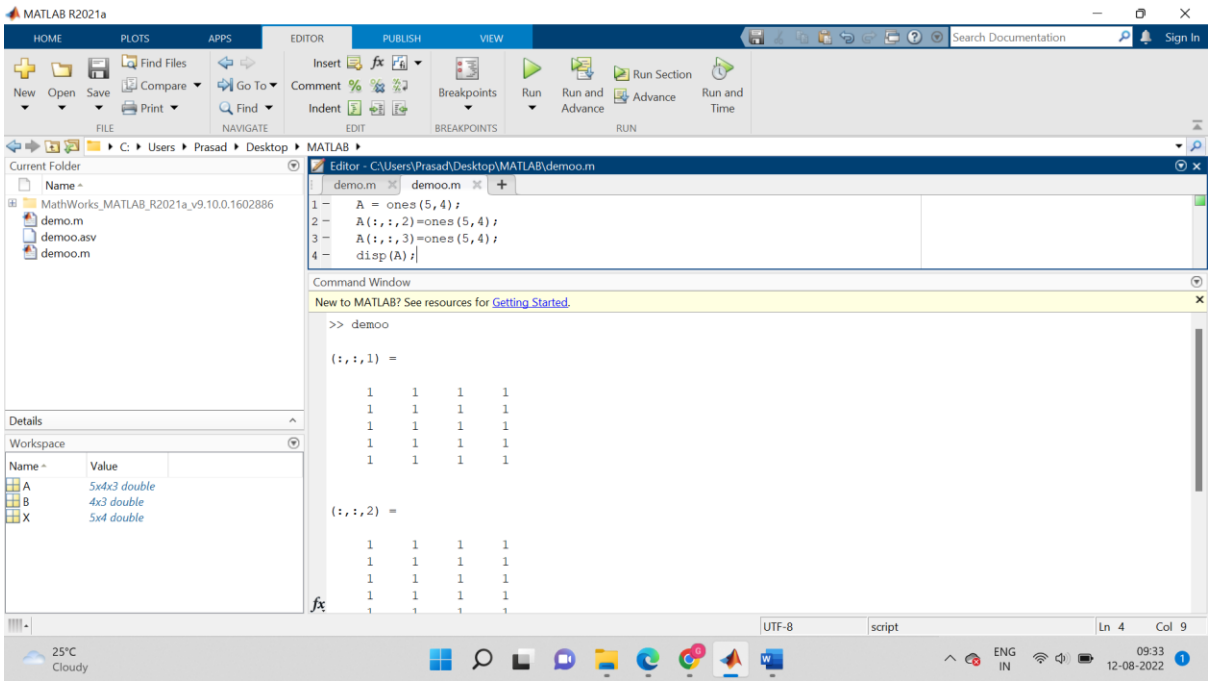
2. Create the following multidimensional arrays of size 5X4 with 3 dimensions:

Explanation:

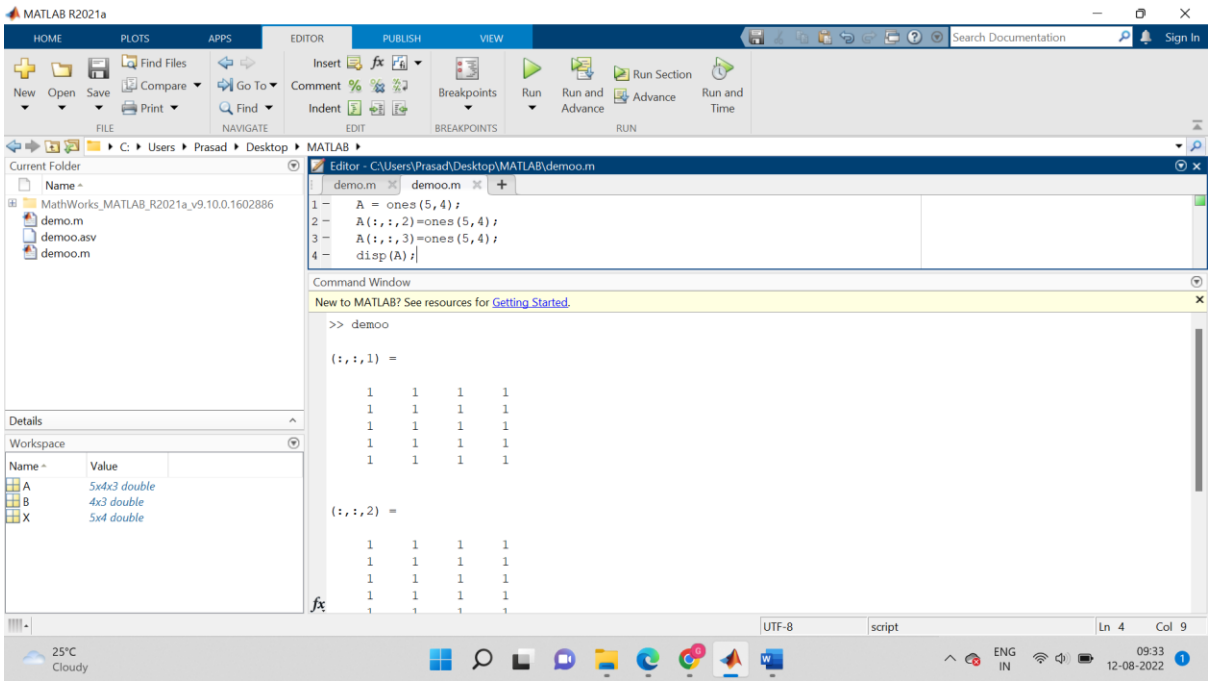
So, I have created an array of required dimension. Now the next dimensions is added, one by one to make it a multidimensional array.

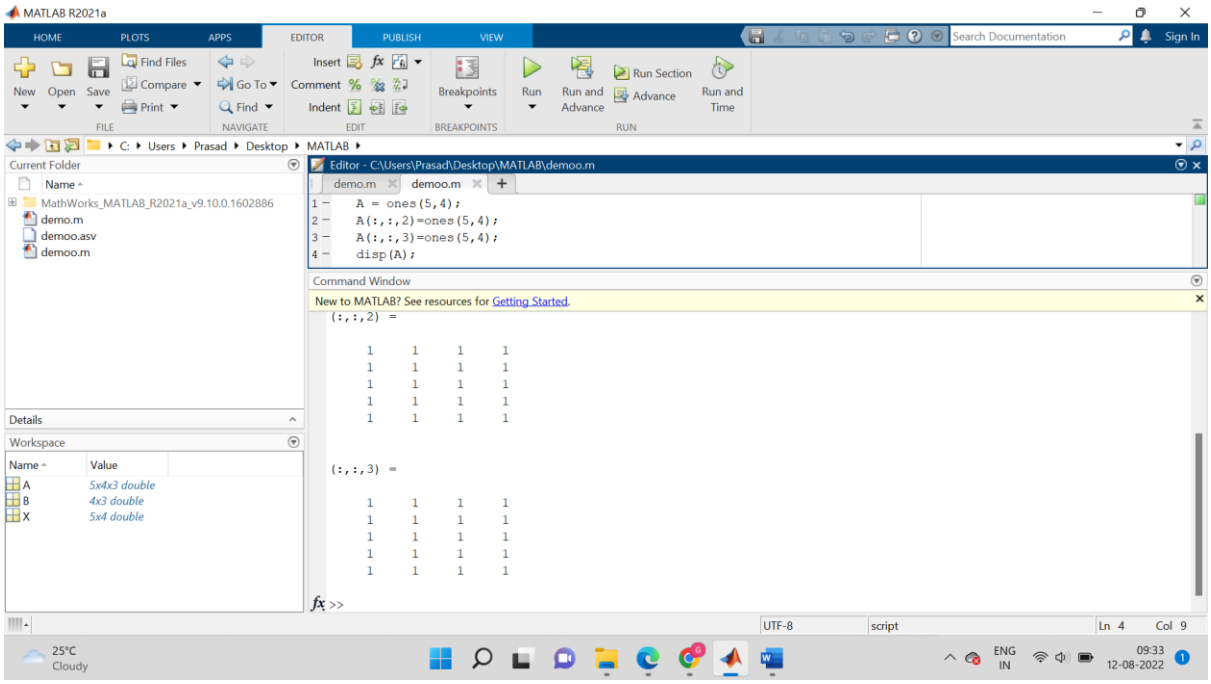
i) ones()

Code:



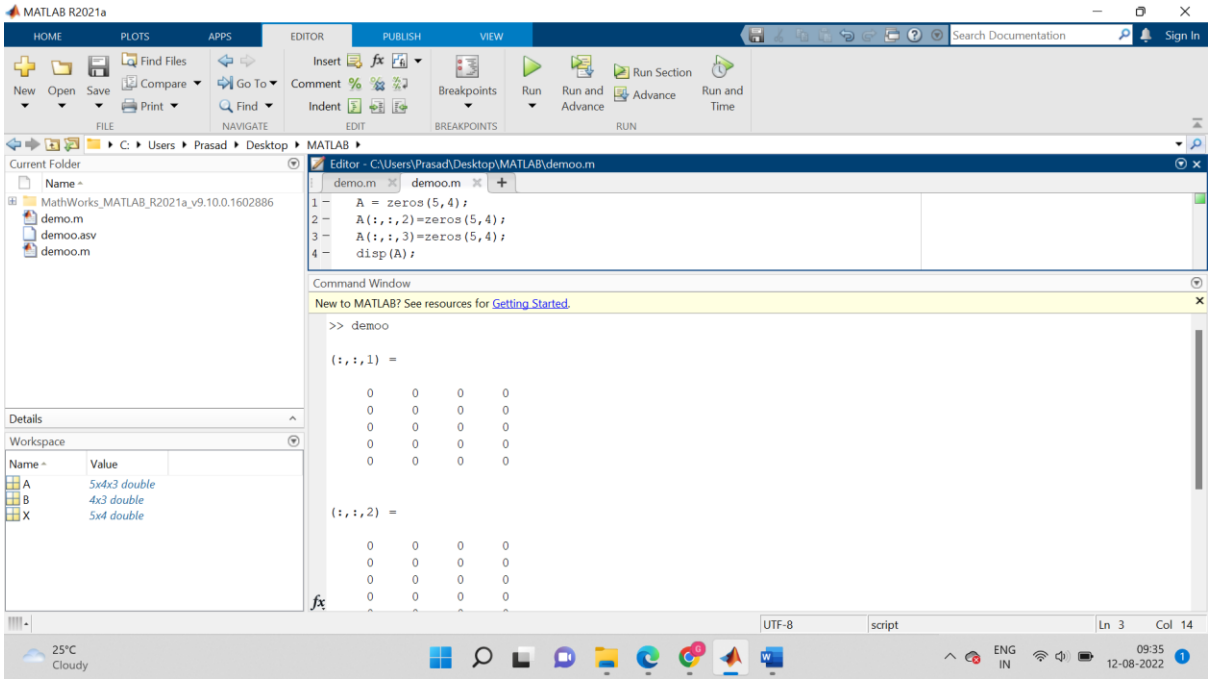

Output:



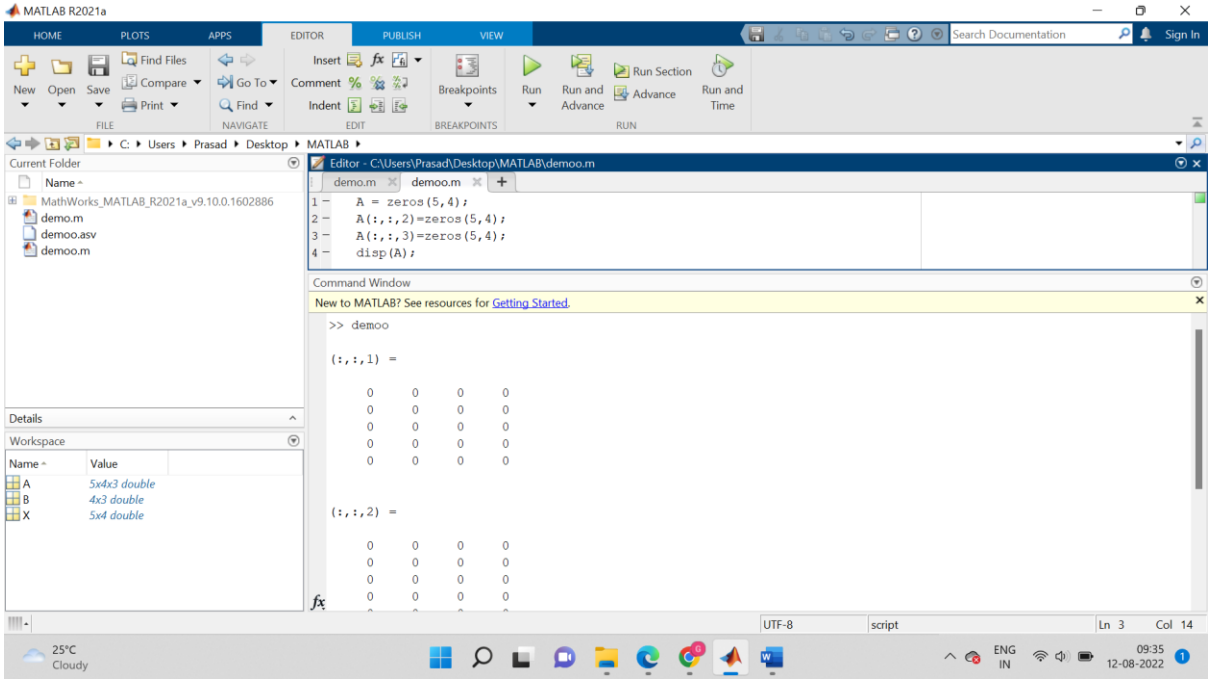


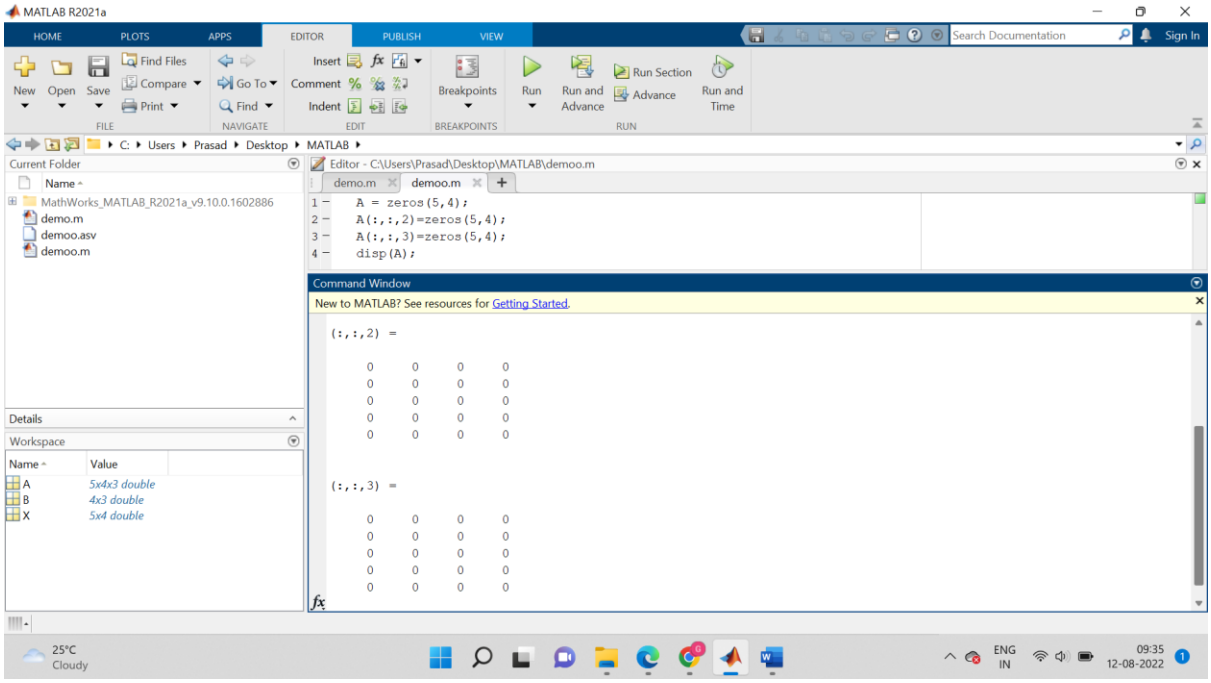

ii) zeros()

Code:



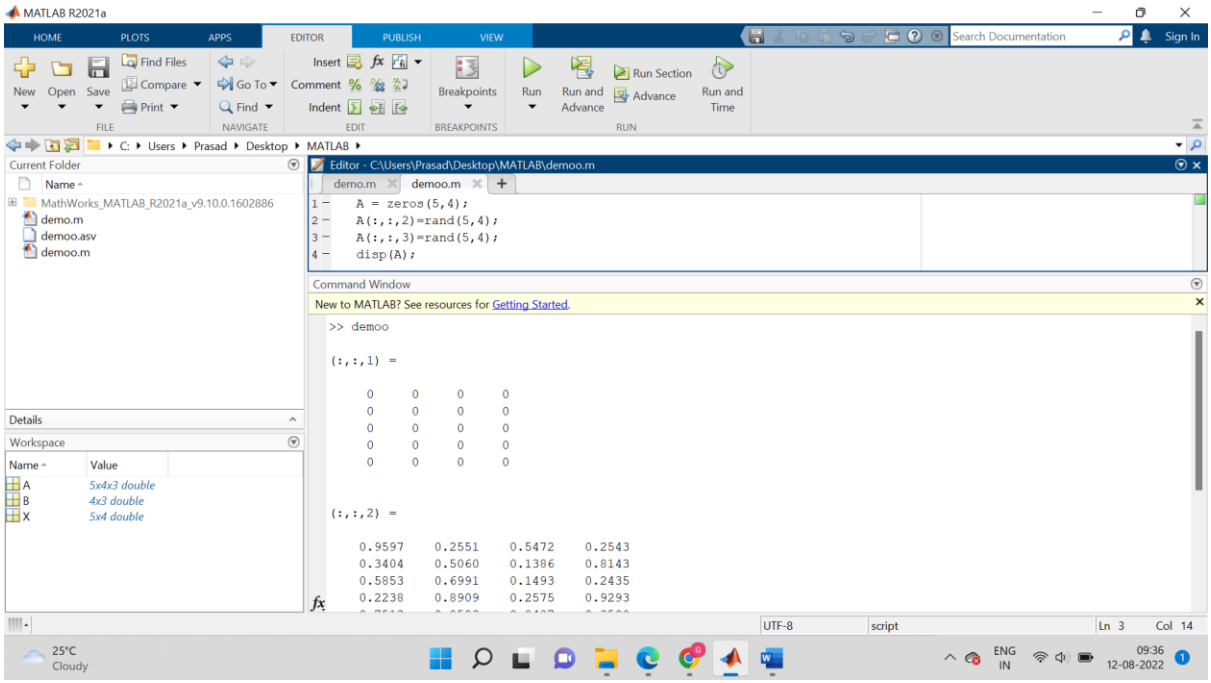
Output:


MATLAB R2021a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ Users ▸ Prasad ▸ Desktop ▸ MATLAB ▸
Current Folder
Name
MathWorks_MATLAB_R2021a_v9.10.0.1602886
demo.m
demoo.asv
demoo.m
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m
demoo.m
1 – A = zeros(5,4);
2 – A(:,:,2)=zeros(5,4);
3 – A(:,:,3)=zeros(5,4);
4 – disp(A);
Command Window
New to MATLAB? See resources for Getting Started.
>> demoo
(:,:,1) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
(:,:,2) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
Details
Workspace
Name
Value
A
5x4x3 double
B
4x3 double
X
5x4 double
UTF-8
script
Ln 3
Col 14
25°C
Cloudy
ENG
IN
09:35
12-08-2022

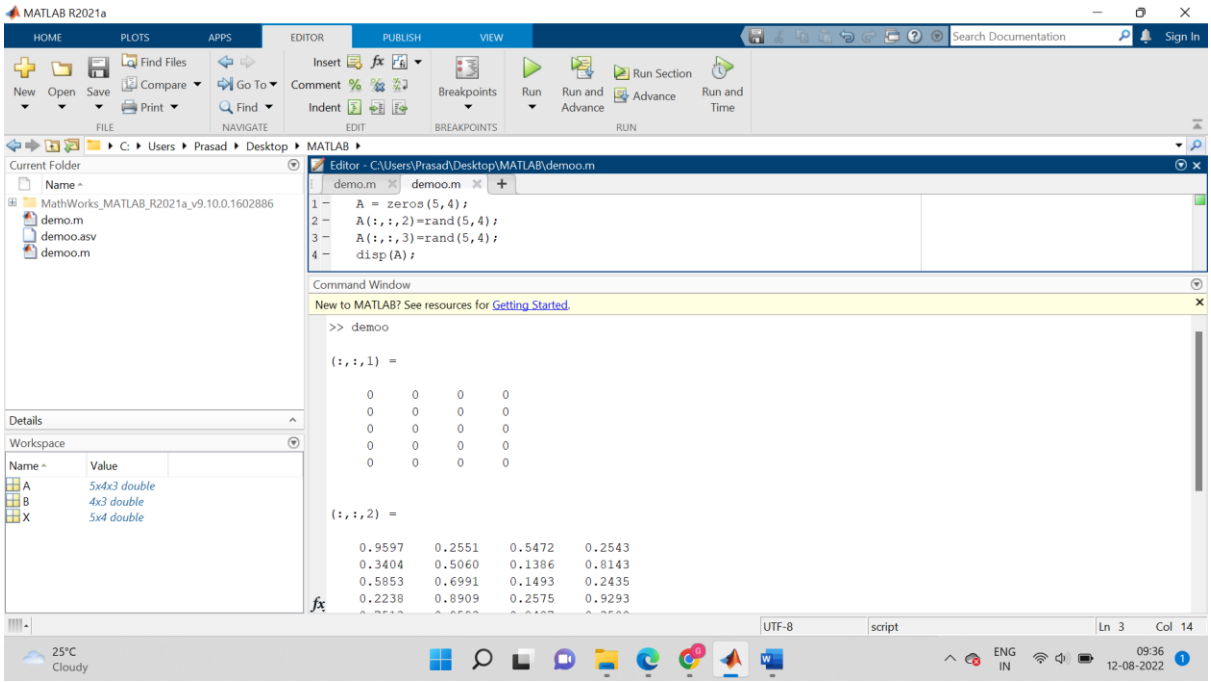

MATLAB R2021a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ Users ▸ Prasad ▸ Desktop ▸ MATLAB ▸
Current Folder
Name
MathWorks_MATLAB_R2021a_v9.10.0.1602886
demo.m
demoo.asv
demoo.m
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m
demoo.m
1 – A = zeros(5,4);
2 – A(:,:,2)=zeros(5,4);
3 – A(:,:,3)=zeros(5,4);
4 – disp(A);
Command Window
New to MATLAB? See resources for Getting Started.
(:,:,2) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
(:,:,3) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
Details
Workspace
Name
Value
A
5x4x3 double
B
4x3 double
X
5x4 double
25°C
Cloudy
ENG
IN
09:35
12-08-2022

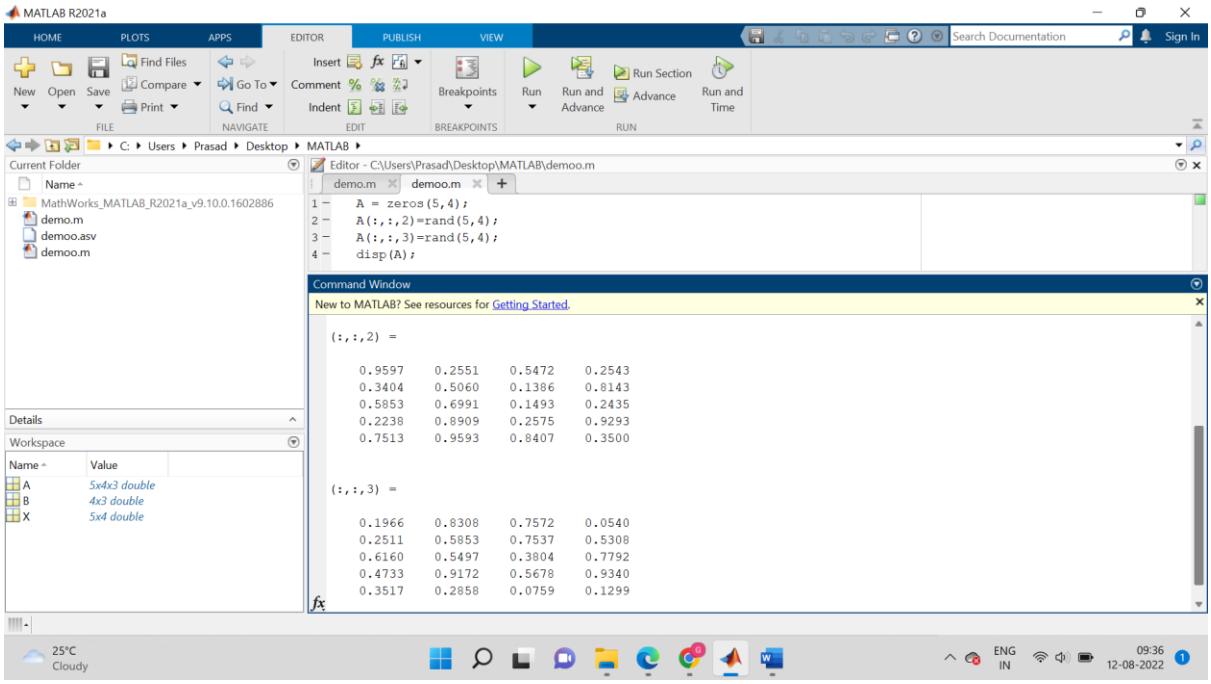
iii) rand()

Code:




Output:



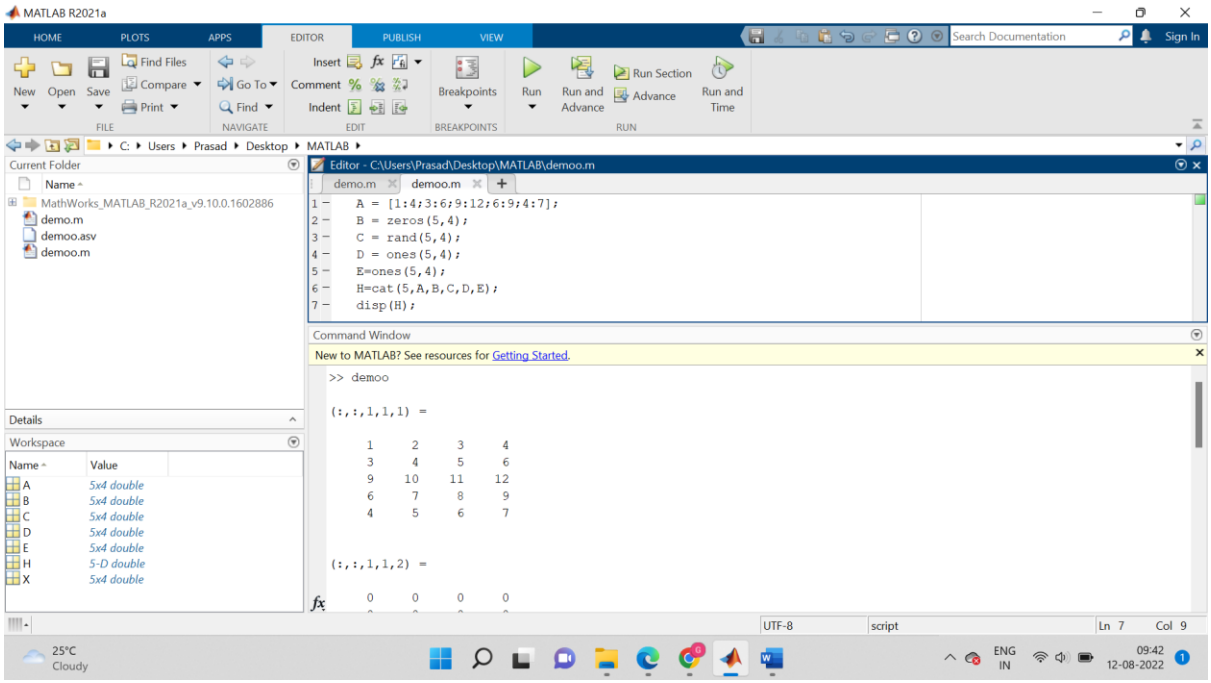

MATLAB R2021a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ Users ▸ Prasad ▸ Desktop ▸ MATLAB ▸
Current Folder
Name
MathWorks_MATLAB_R2021a_v9.10.0.1602886
demo.m
demoo.asv
demoo.m
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m
demoo.m
1 A = zeros(5,4);
2 A(:,:,2)=rand(5,4);
3 A(:,:,3)=rand(5,4);
4 disp(A);
Command Window
New to MATLAB? See resources for Getting Started.
(:,:,2) =
0.9597 0.2551 0.5472 0.2543
0.3404 0.5060 0.1386 0.8143
0.5853 0.6991 0.1493 0.2435
0.2238 0.8909 0.2575 0.9293
0.7513 0.9593 0.8407 0.3500
(:,:,3) =
0.1966 0.8308 0.7572 0.0540
0.2511 0.5853 0.7537 0.5308
0.6160 0.5497 0.3804 0.7792
0.4733 0.9172 0.5678 0.9340
0.3517 0.2858 0.0759 0.1299
Details
Workspace
Name
Value
A 5x4x3 double
B 4x3 double
X 5x4 double
25°C
Cloudy
ENG
IN
09:36
12-08-2022

3. Demonstrate the use of cat () the creation of a multidimensional array of 5X4 with 5 dimensions. Use " : range operator" to create one array, use "[ ] concatenation operator" to create another array, use appropriate special array functions.

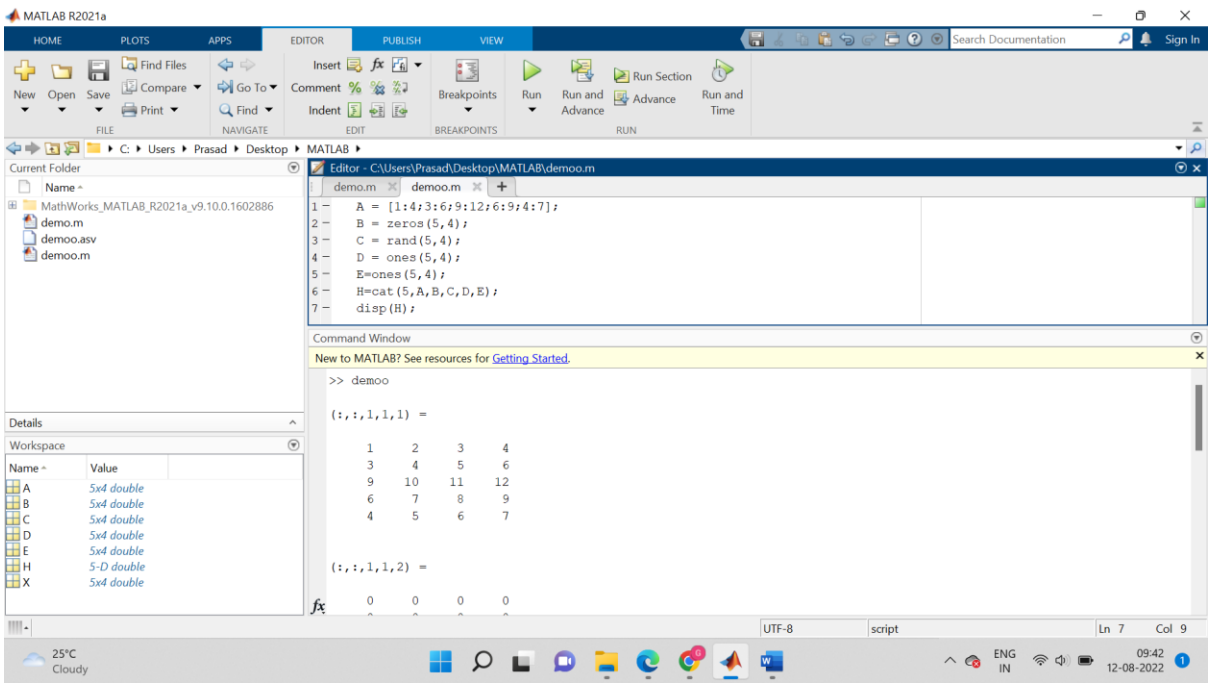
Explanation:

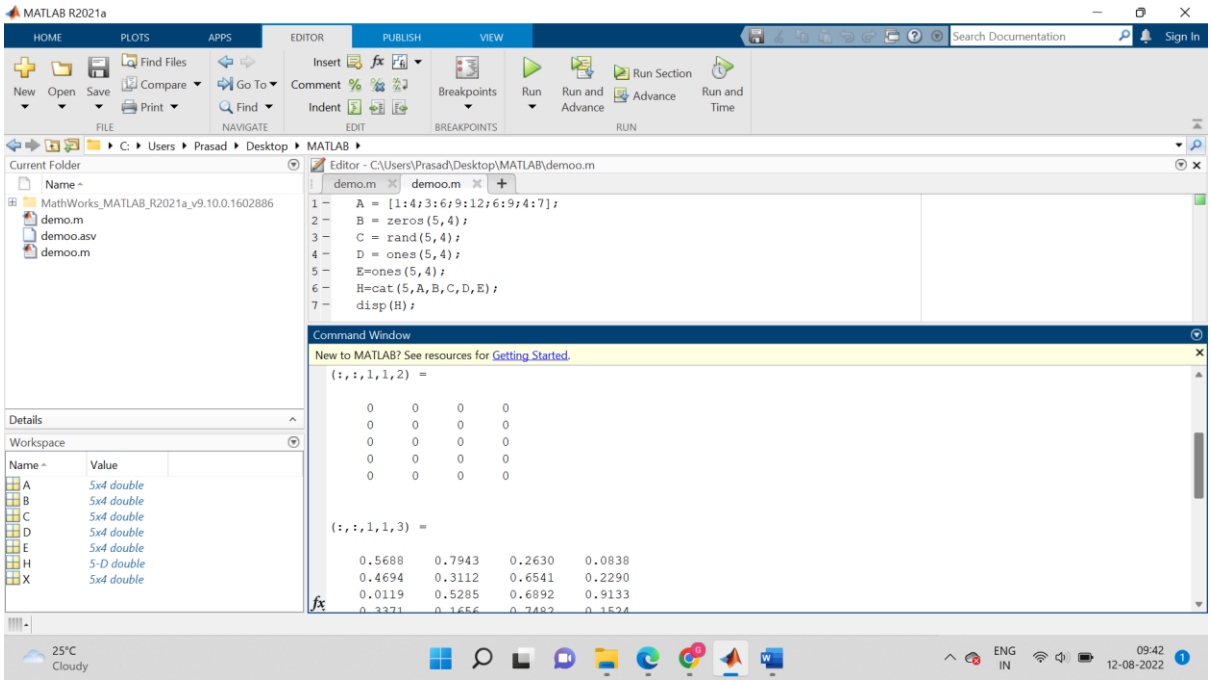
So, I have created 5 arrays of same dimensions. Use cat() function to concatenate them in multidimensional array.

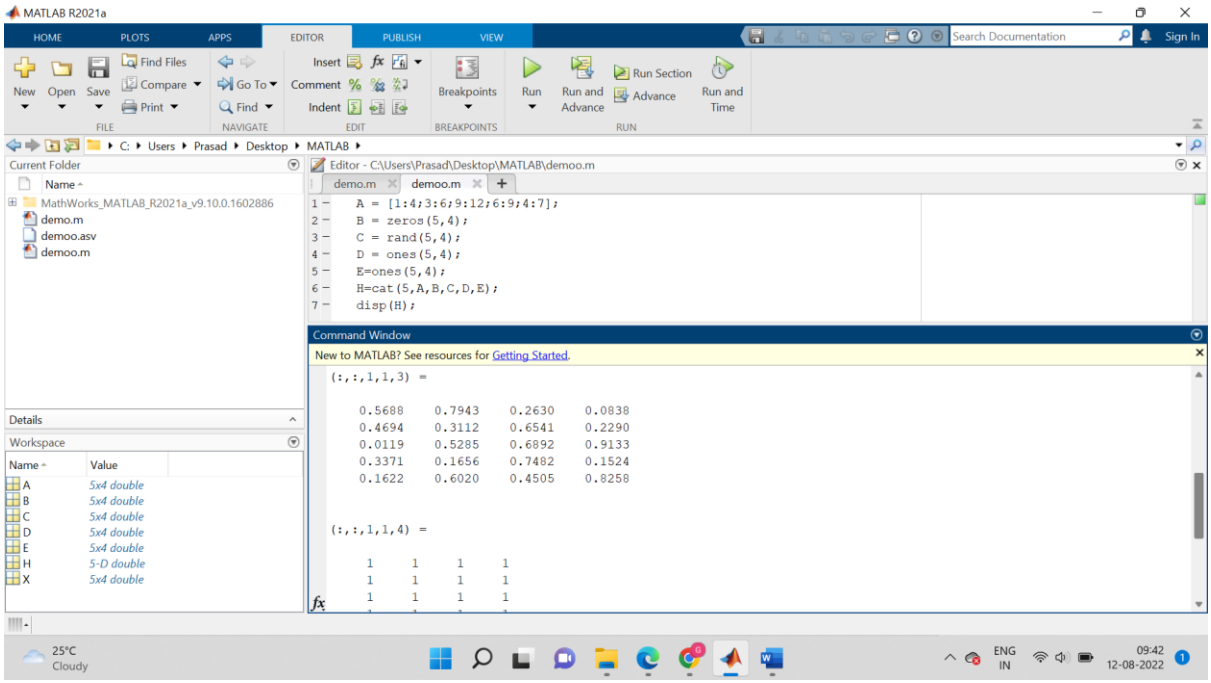
Code:

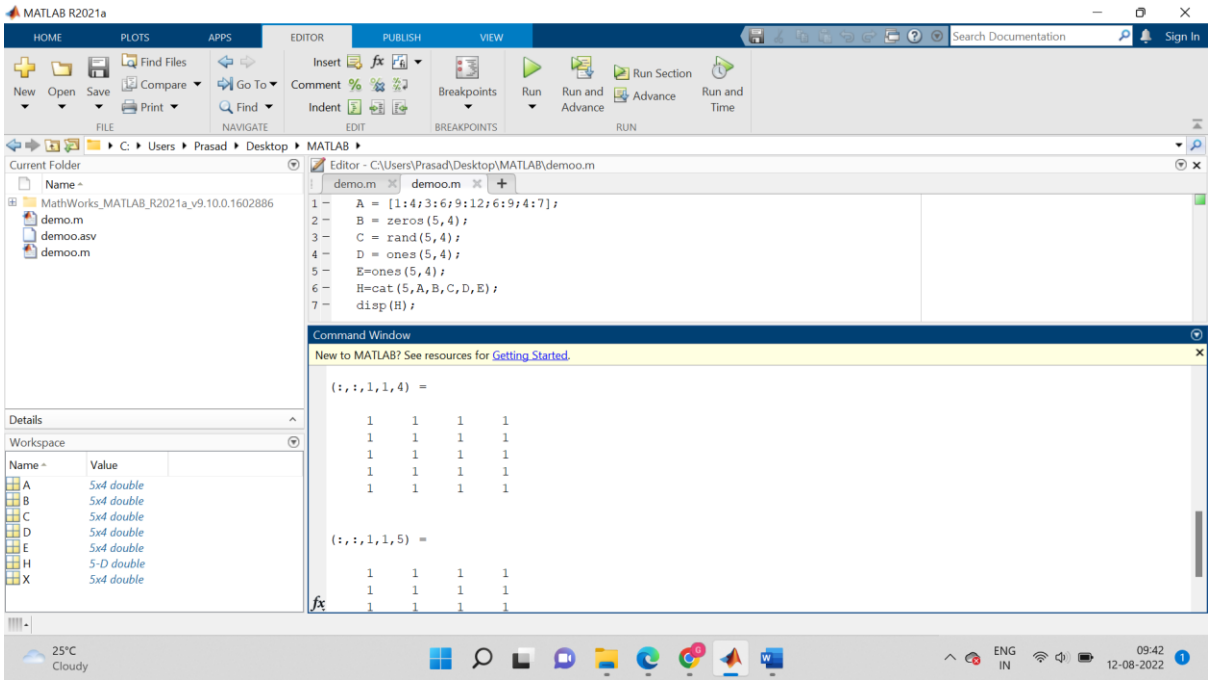


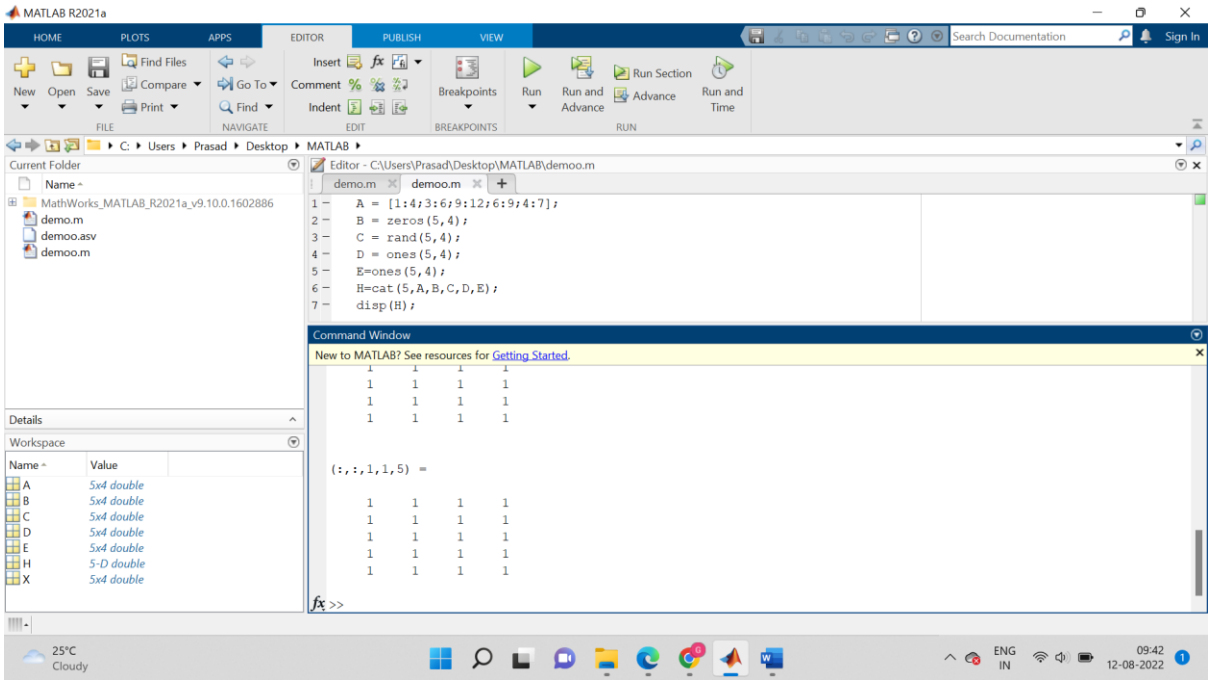

Output:




MATLAB R2021a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ Users ▸ Prasad ▸ Desktop ▸ MATLAB ▸
Current Folder
Name
MathWorks_MATLAB_R2021a_v9.10.0.1602886
demo.m
demoo.asv
demoo.m
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m
demoo.m
1 A = [1:4;3:6;9:12;6:9;4:7];
2 B = zeros(5,4);
3 C = rand(5,4);
4 D = ones(5,4);
5 E=ones(5,4);
6 H=cat(5,A,B,C,D,E);
7 disp(H);
Command Window
New to MATLAB? See resources for Getting Started.
(:,:,1,1,2) =
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
(:,:,1,1,3) =
0.5688 0.7943 0.2630 0.0838
0.4694 0.3112 0.6541 0.2290
0.0119 0.5285 0.6892 0.9133
Details
Workspace
Name
Value
A 5x4 double
B 5x4 double
C 5x4 double
D 5x4 double
E 5x4 double
H 5-D double
X 5x4 double
25°C
Cloudy
ENG
IN
09:42
12-08-2022

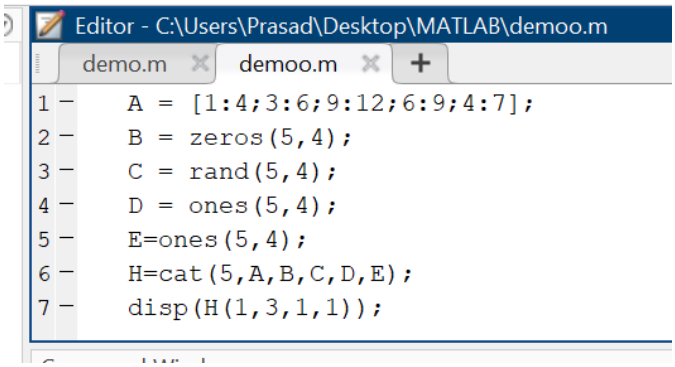

MATLAB R2021a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ Users ▸ Prasad ▸ Desktop ▸ MATLAB ▸
Current Folder
Name
MathWorks_MATLAB_R2021a_v9.10.0.1602886
demo.m
demoo.asv
demoo.m
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m
demoo.m
1 A = [1:4;3:6;9:12;6:9;4:7];
2 B = zeros(5,4);
3 C = rand(5,4);
4 D = ones(5,4);
5 E=ones(5,4);
6 H=cat(5,A,B,C,D,E);
7 disp(H);
Command Window
New to MATLAB? See resources for Getting Started.
(:,:,1,1,3) =
0.5688 0.7943 0.2630 0.0838
0.4694 0.3112 0.6541 0.2290
0.0119 0.5285 0.6892 0.9133
0.3371 0.1656 0.7482 0.1524
0.1622 0.6020 0.4505 0.8258
(:,:,1,1,4) =
1 1 1 1
1 1 1 1
1 1 1 1
Details
Workspace
Name
Value
A 5x4 double
B 5x4 double
C 5x4 double
D 5x4 double
E 5x4 double
H 5-D double
X 5x4 double
25°C
Cloudy
ENG
IN
09:42
12-08-2022


MATLAB R2021a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ Users ▸ Prasad ▸ Desktop ▸ MATLAB ▸
Current Folder
Name
MathWorks_MATLAB_R2021a_v9.10.0.1602886
demo.m
demoo.asv
demoo.m
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m
demoo.m
1 A = [1:4;3:6;9:12;6:9;4:7];
2 B = zeros(5,4);
3 C = rand(5,4);
4 D = ones(5,4);
5 E=ones(5,4);
6 H=cat(5,A,B,C,D,E);
7 disp(H);
Command Window
New to MATLAB? See resources for Getting Started.
(:,:,1,1,4) =
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
(:,:,1,1,5) =
1 1 1 1
1 1 1 1
1 1 1 1
Details
Workspace
Name
Value
A 5x4 double
B 5x4 double
C 5x4 double
D 5x4 double
E 5x4 double
H 5-D double
X 5x4 double
25°C
Cloudy
ENG
IN
09:42
12-08-2022

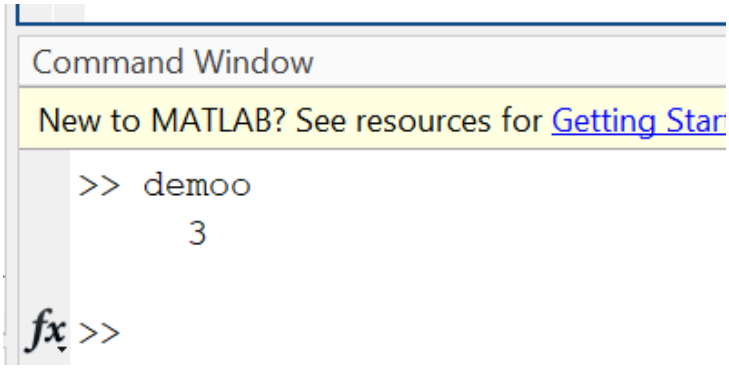

MATLAB R2021a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
C: ▸ Users ▸ Prasad ▸ Desktop ▸ MATLAB ▸
Current Folder
Name
MathWorks_MATLAB_R2021a_v9.10.0.1602886
demo.m
demoo.asv
demoo.m
Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m
demoo.m
1 A = [1:4;3:6;9:12;6:9;4:7];
2 B = zeros(5,4);
3 C = rand(5,4);
4 D = ones(5,4);
5 E=ones(5,4);
6 H=cat(5,A,B,C,D,E);
7 disp(H);
Command Window
New to MATLAB? See resources for Getting Started.
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
(:,:,1,1,5) =
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
>>
Details
Workspace
Name
Value
A 5x4 double
B 5x4 double
C 5x4 double
D 5x4 double
E 5x4 double
H 5-D double
X 5x4 double
25°C
Cloudy
ENG
IN
09:42
12-08-2022

4. From the above array access the following elements:
    i. Any one element of a specific position in any dimension of your choice.

Code:

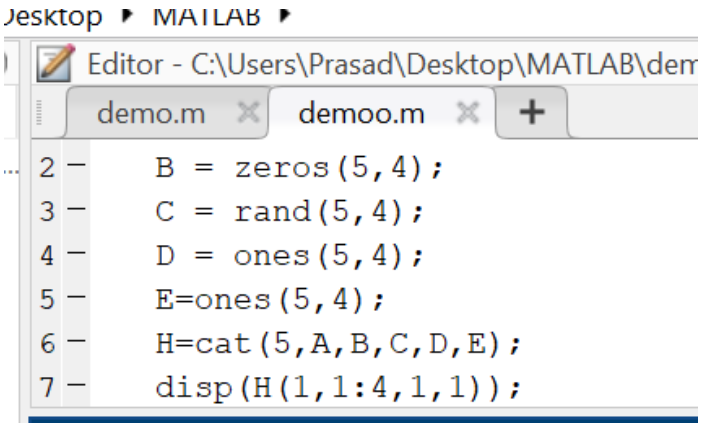



Output:

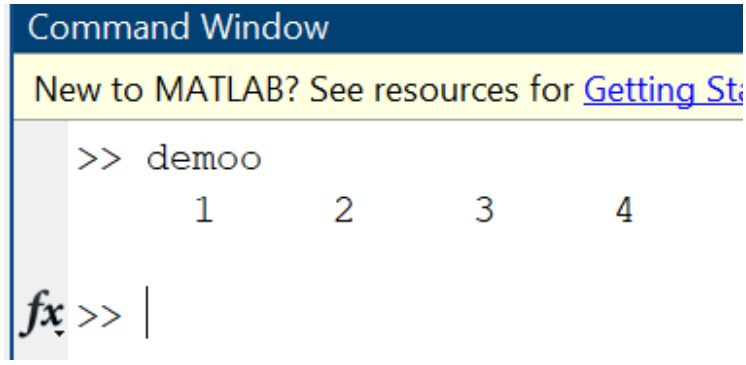


ii. Range of any four elements in a row in any dimension of your choice.

Code:

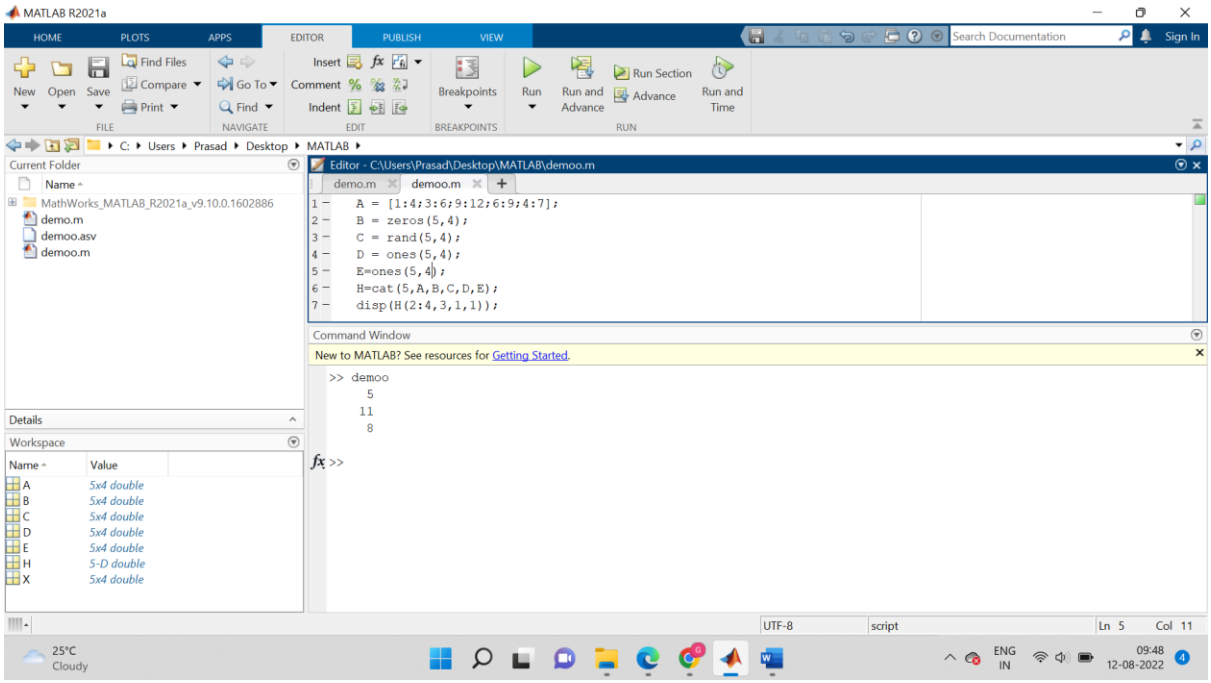



Output:

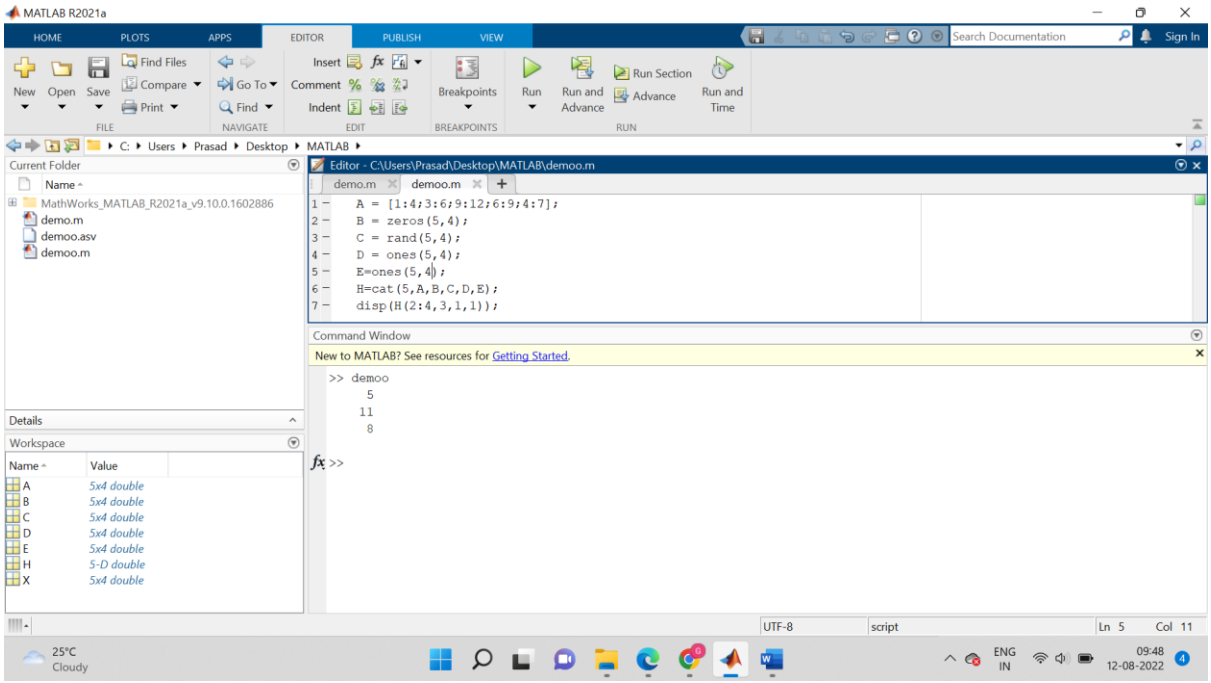


iii. Range of any three elements in a column in any dimension of your choice.

Code:

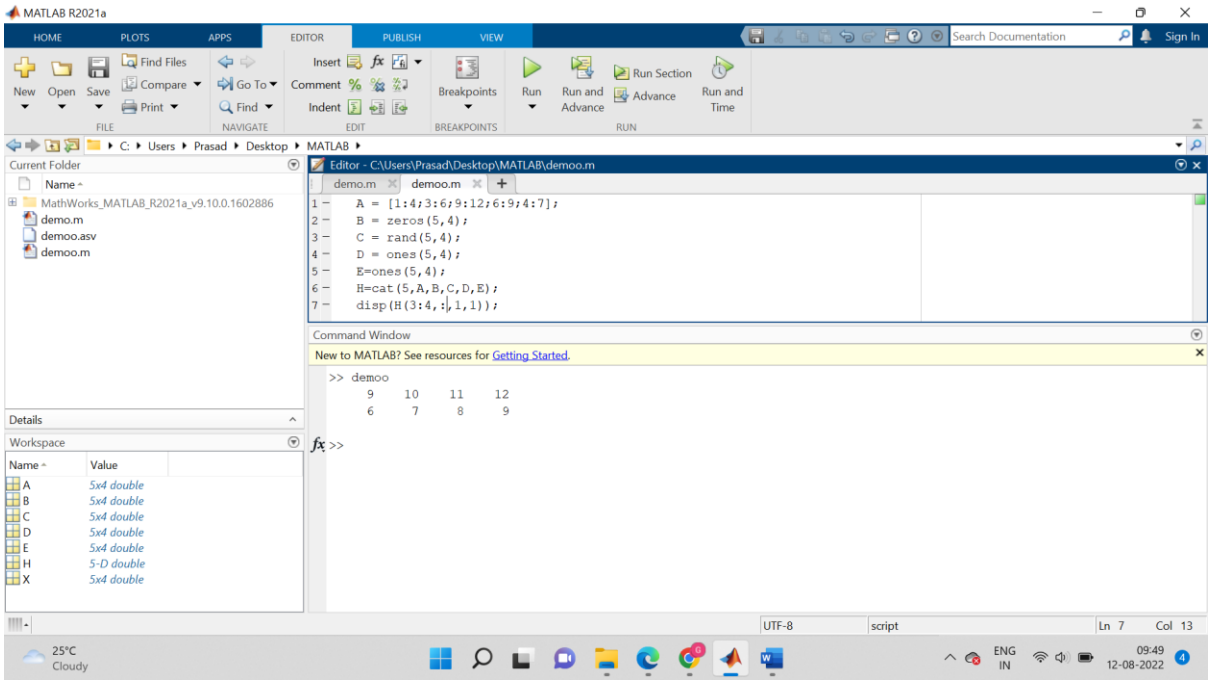



Output:

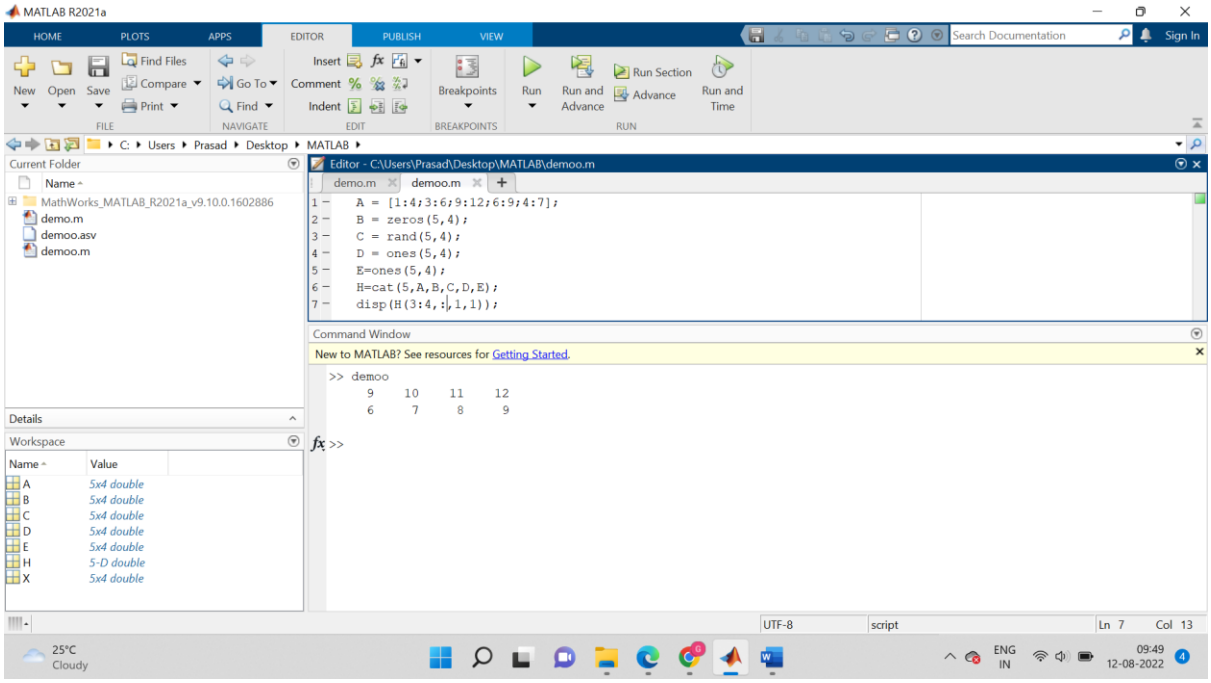


iv. All the elements of two rows in any dimension of your choice.

Code:

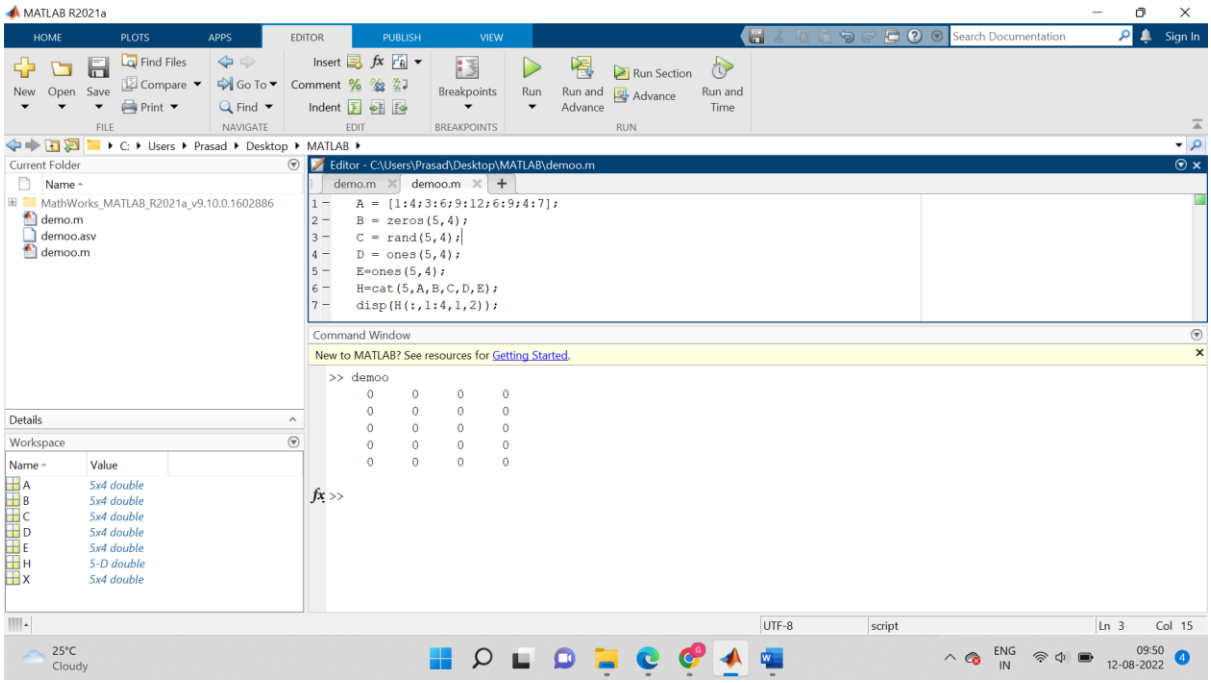



Output:

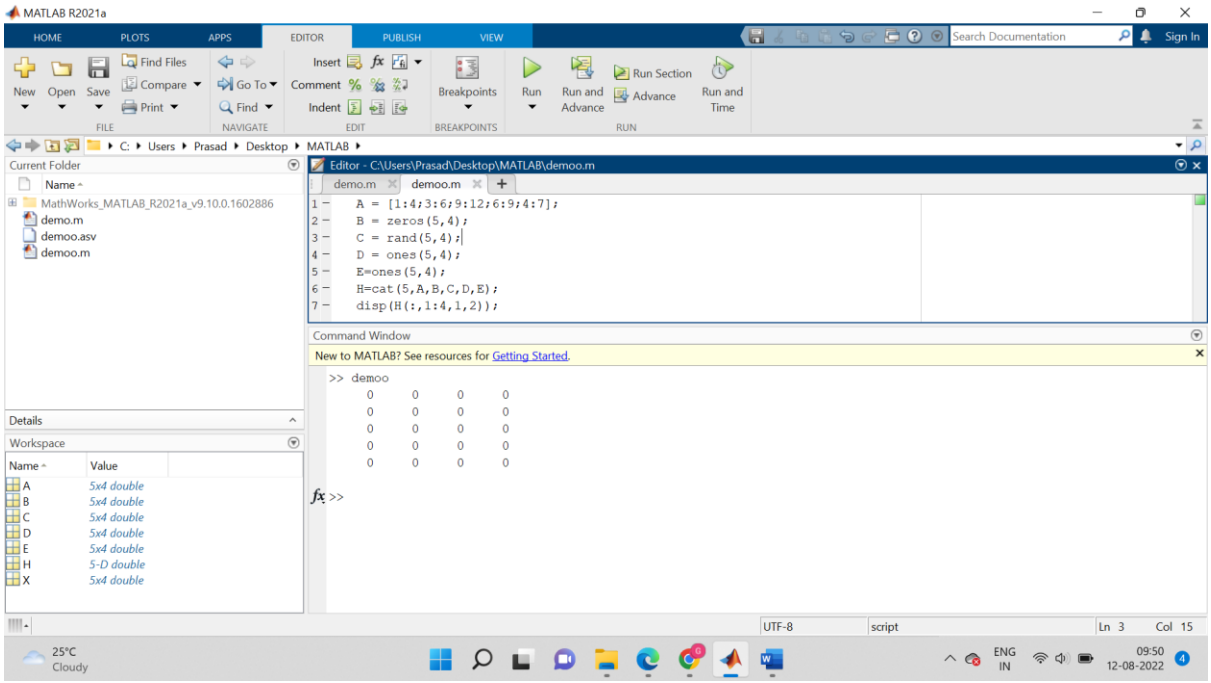


v. All the elements of any 4 columns in any dimension of your choice.

Code:




Output:

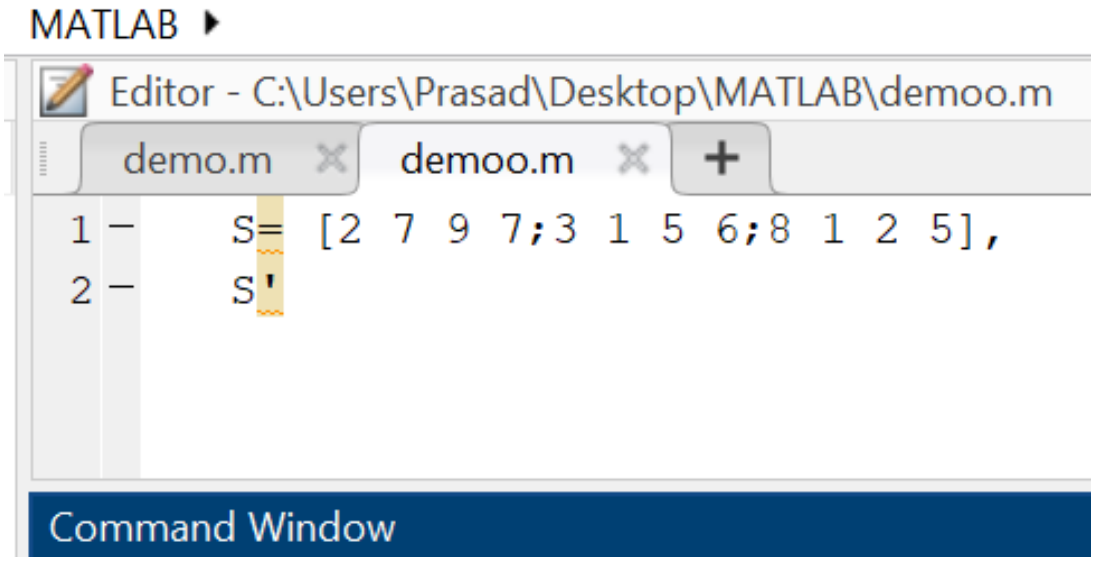


**Colon Notation:**

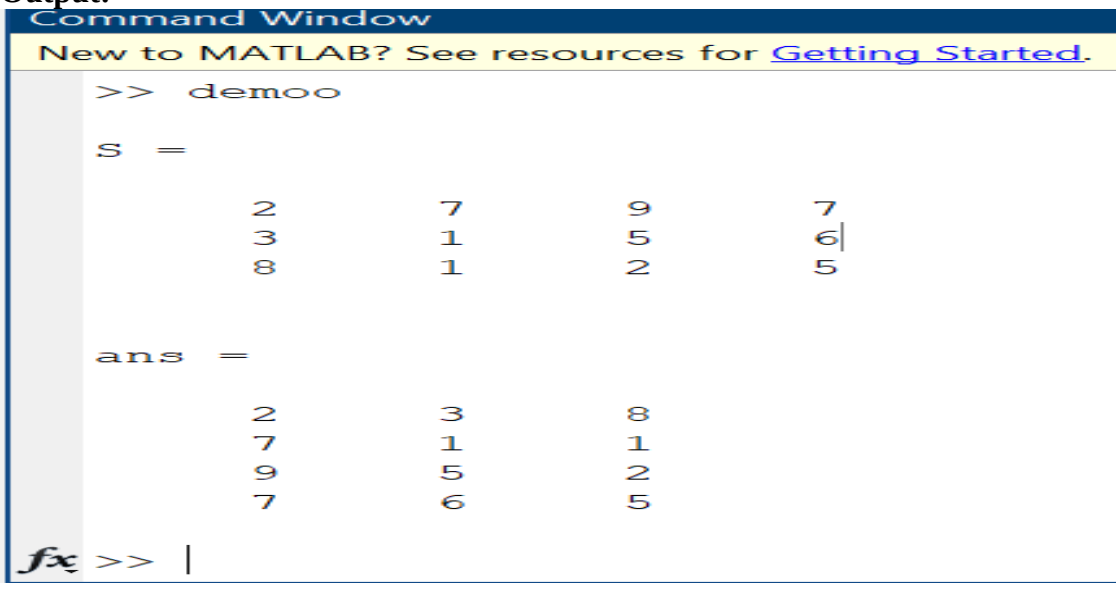
1. If S= [2 7 9 7;3 1 5 6;8 1 2 5], do the following actions
    a. S'

**Explanation:**

**Code:**

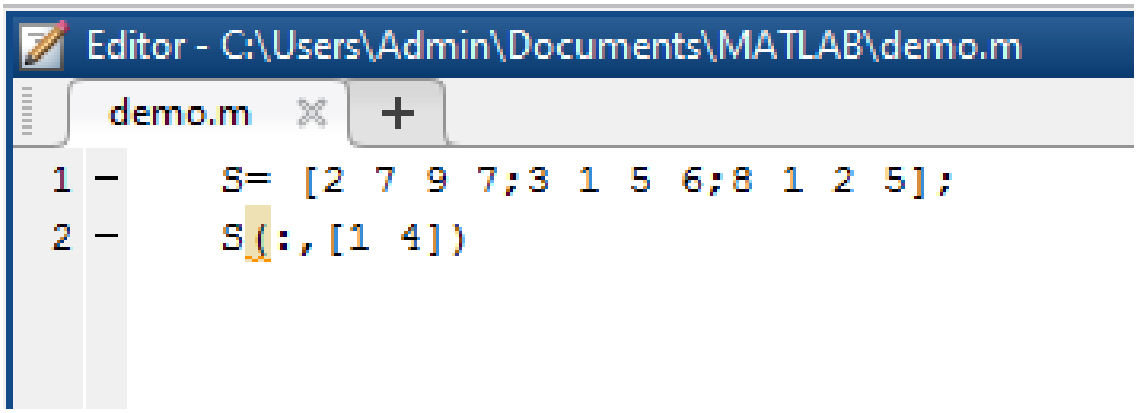



**Output:**

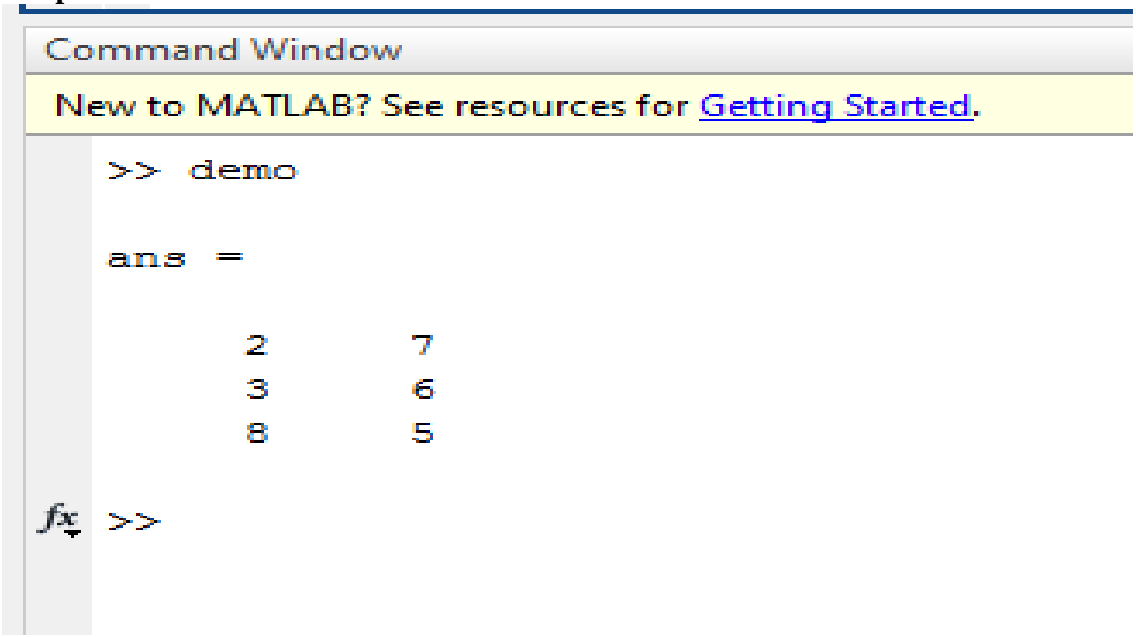


**b. S(:,[1 4])**

**Code:**

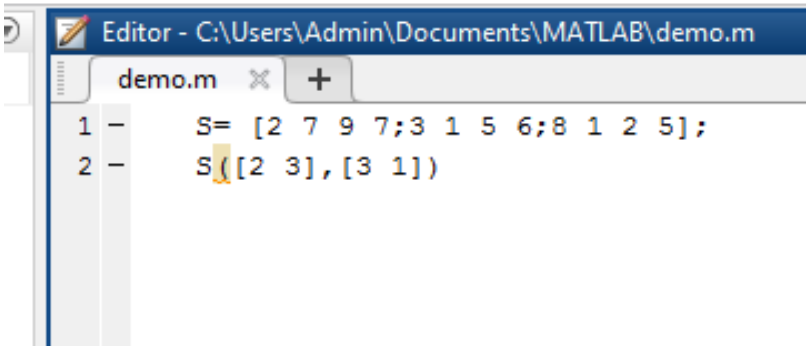



**Output:**

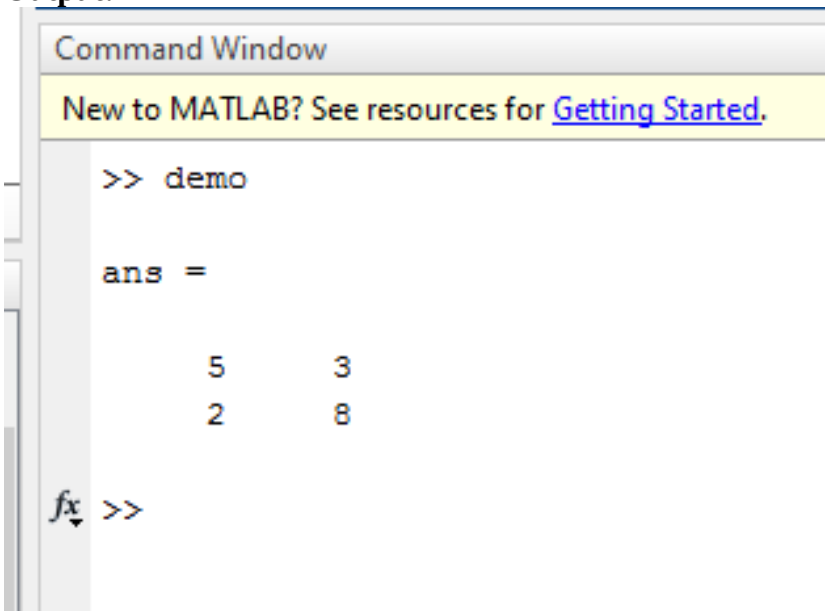


c. **S([2 3],[3 1])**

**Code:**

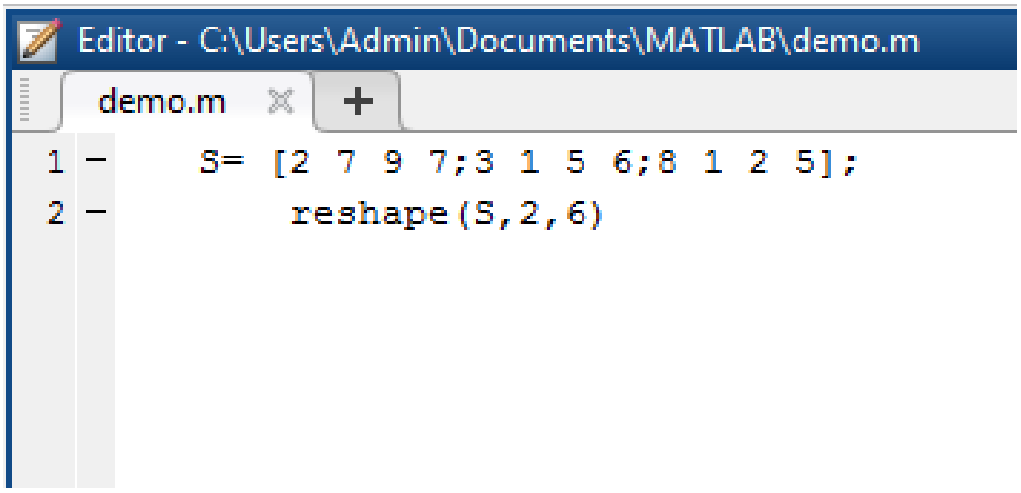



**Output:**

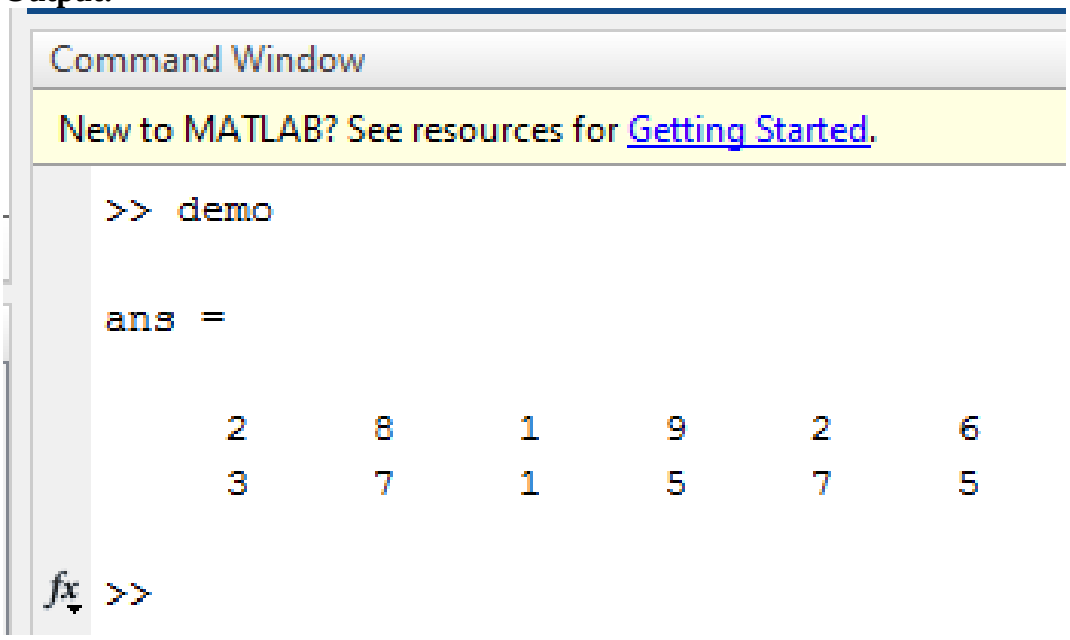


**d. reshape(S,2,6)**

**Code:**

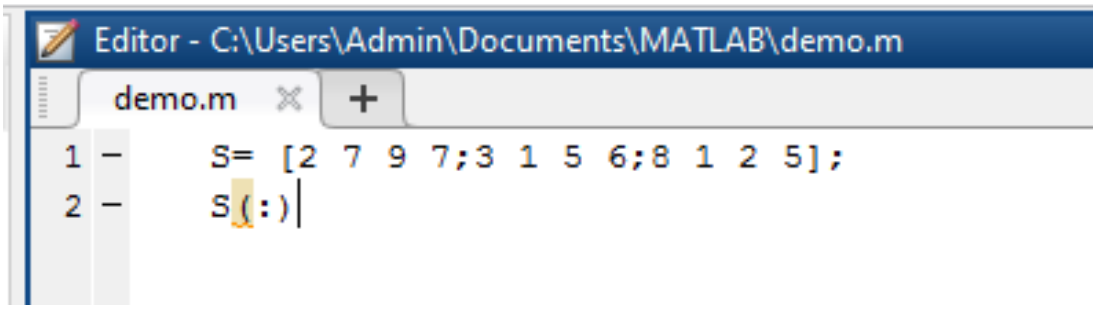



**Output:**

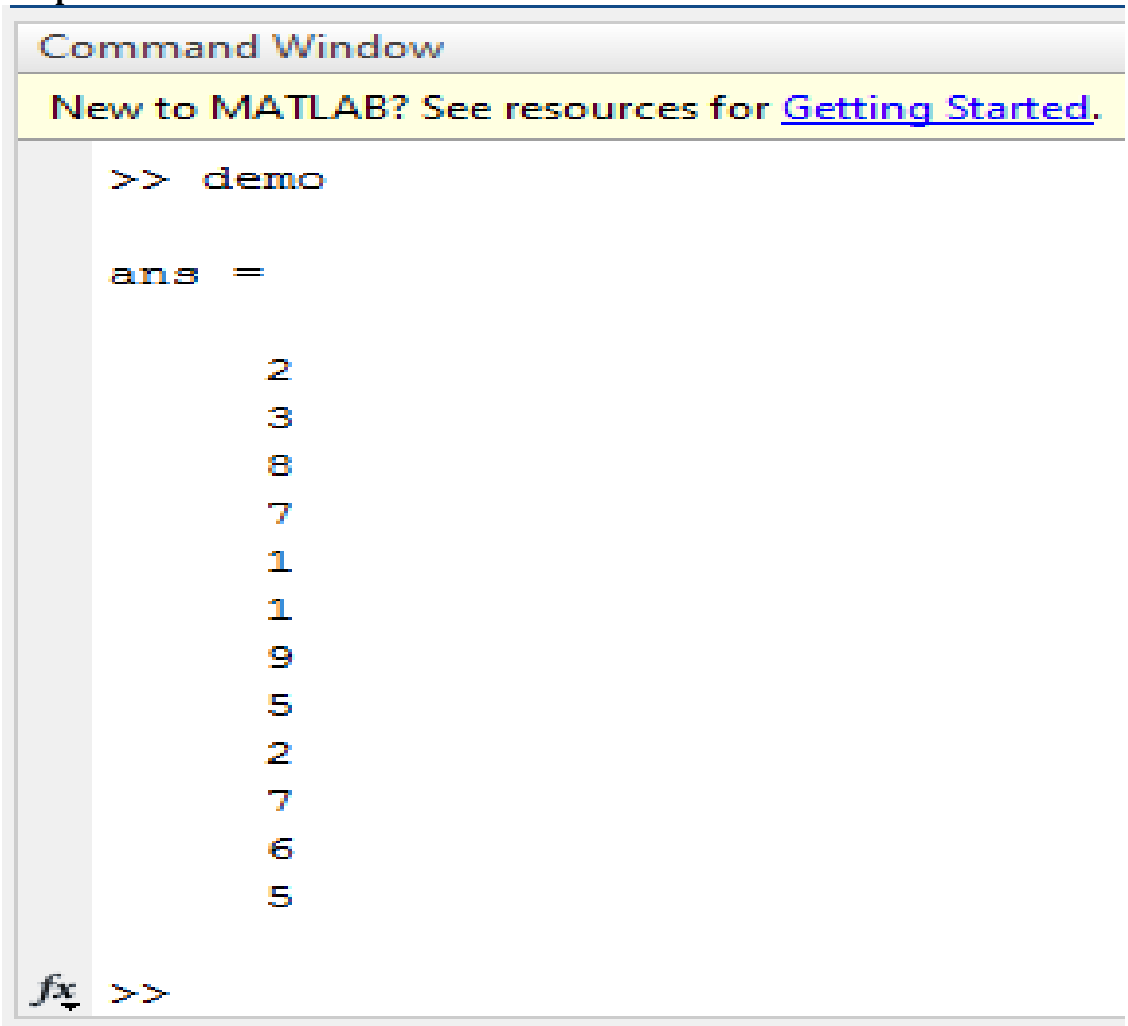


**e. S(:)**

**Code:**

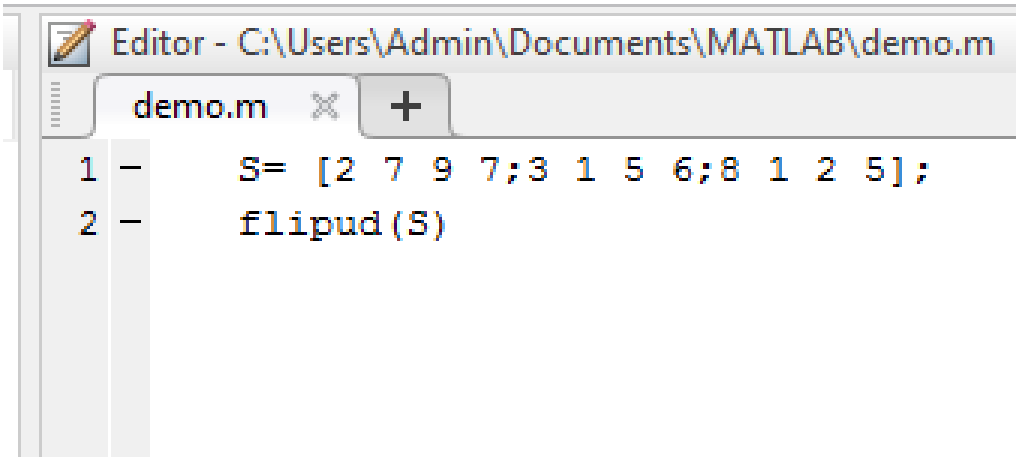



**Output:**

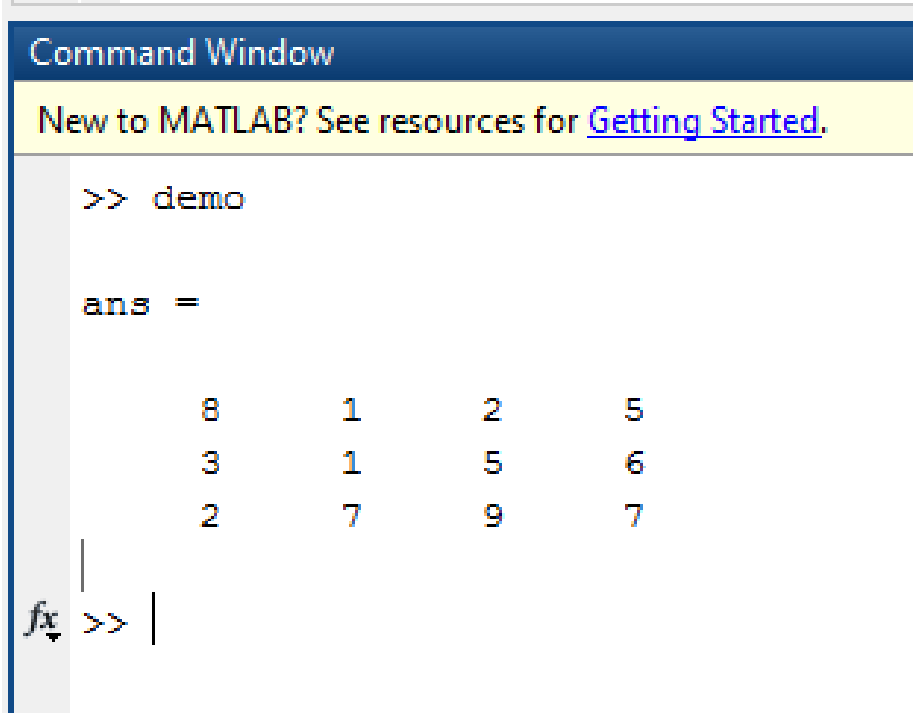


**f. flipud(S)**

**Code:**

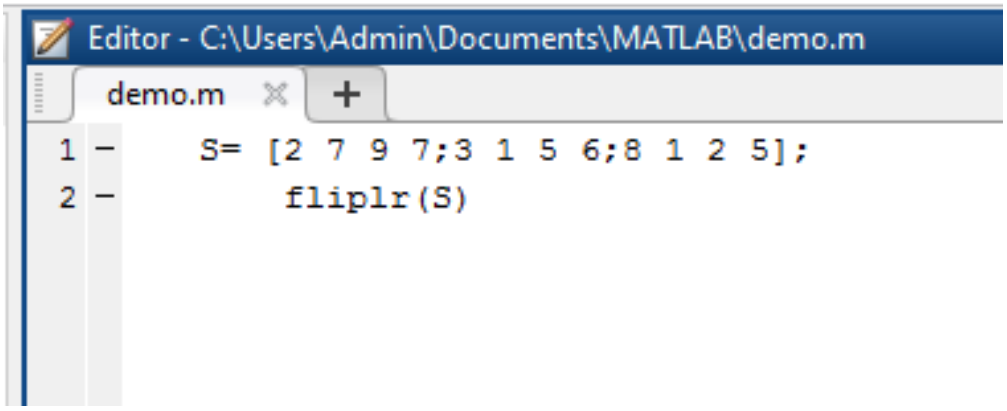



**Output:**



**g. fliplr(S)**

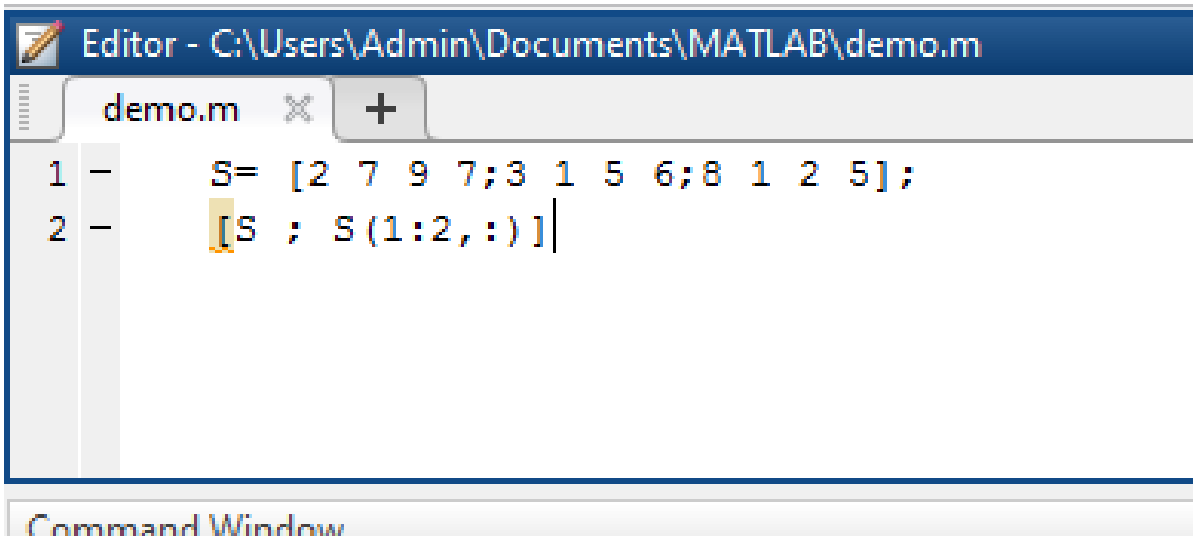
**Code:**



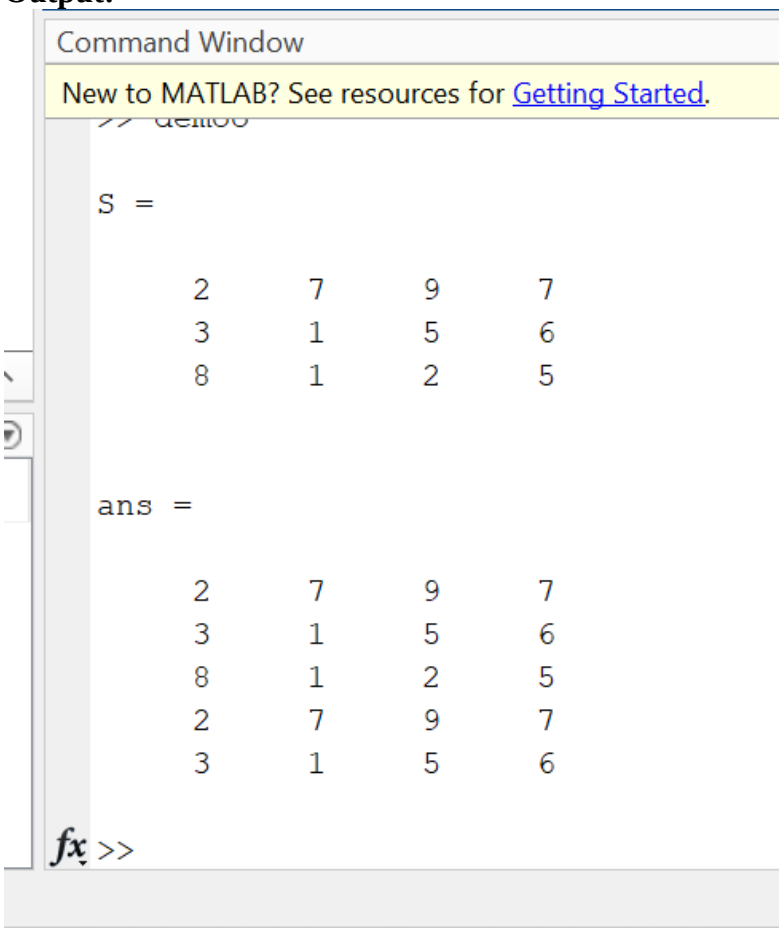

**Output**

**h. [S ; S(1:2,:)]**

**Code:**

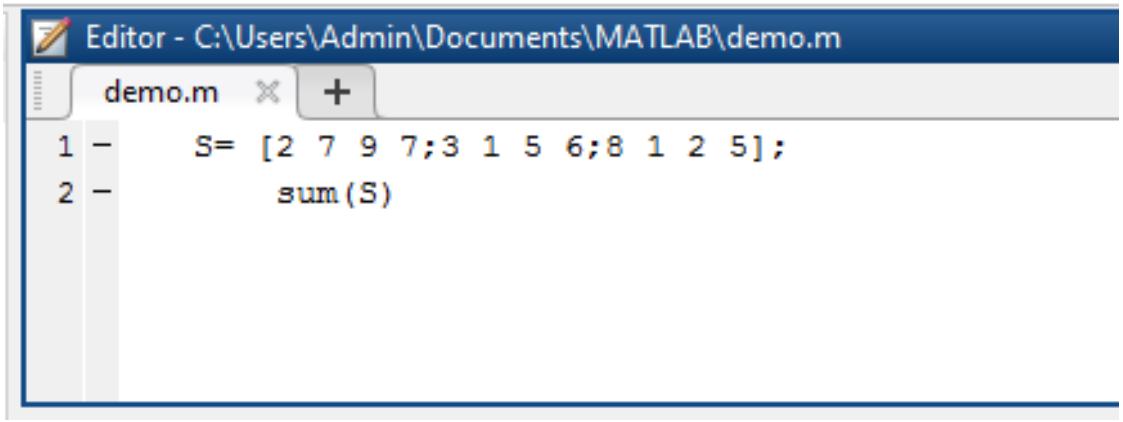



**Output:**

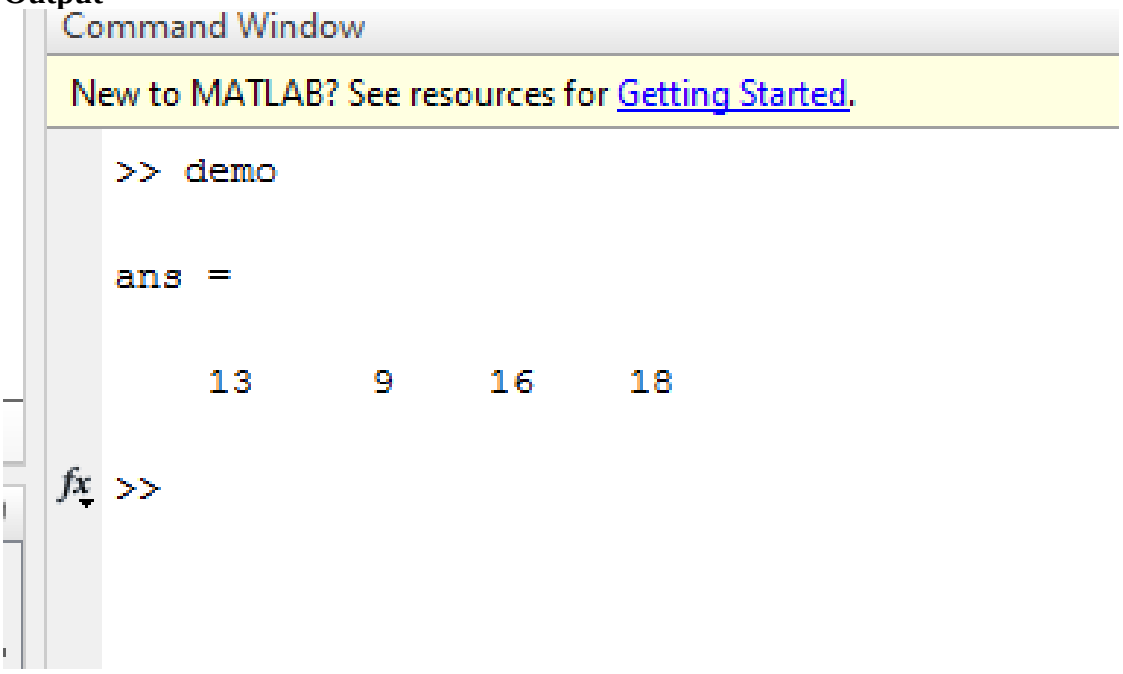


i. **sum(S)**

**Code:**

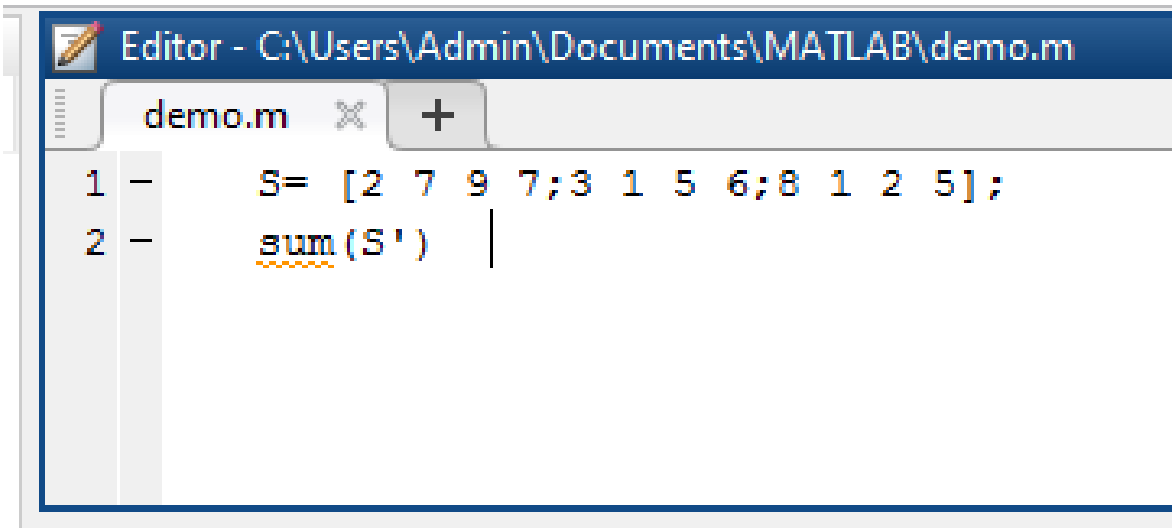



**Output**

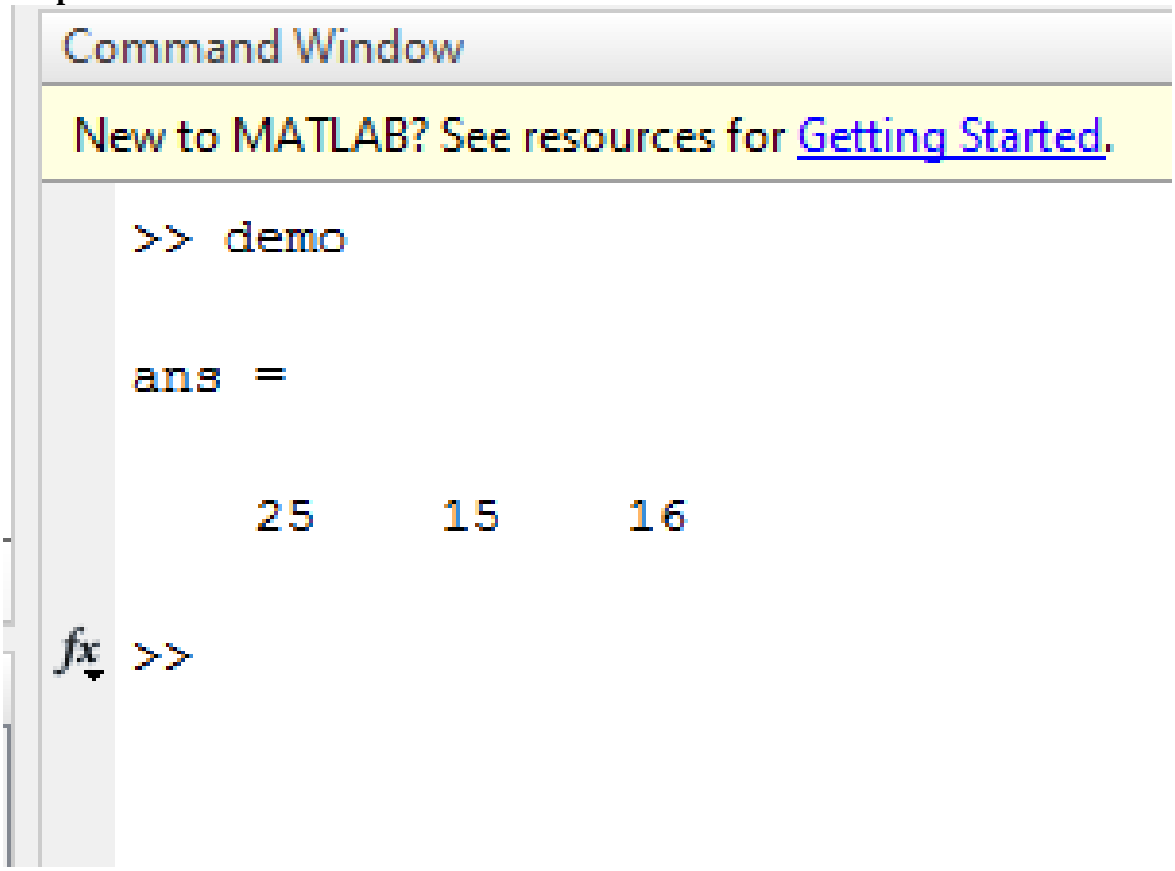


**j. sum(S')**

**Code:**

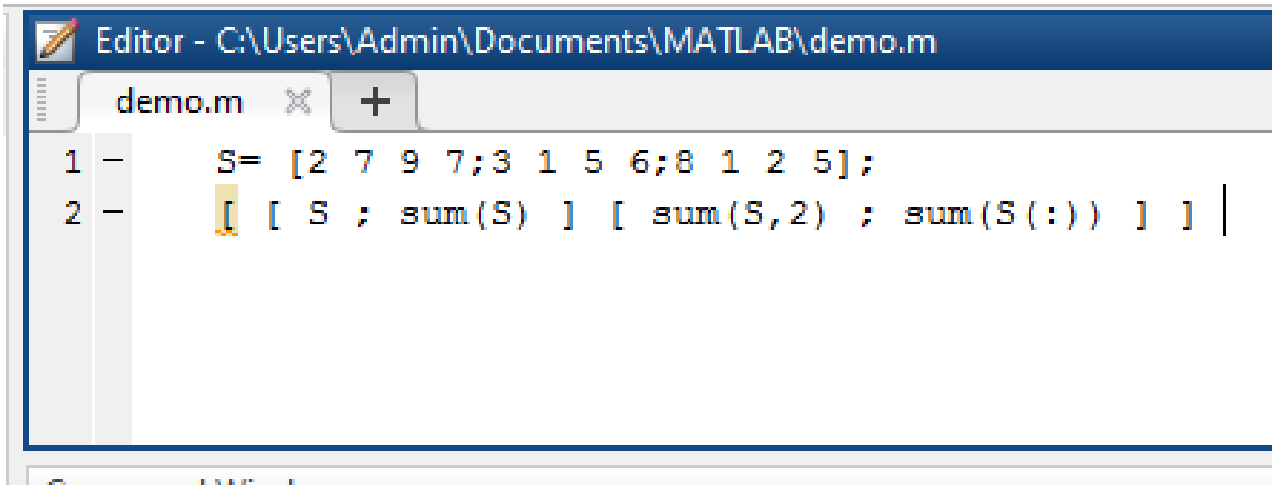



**Output**

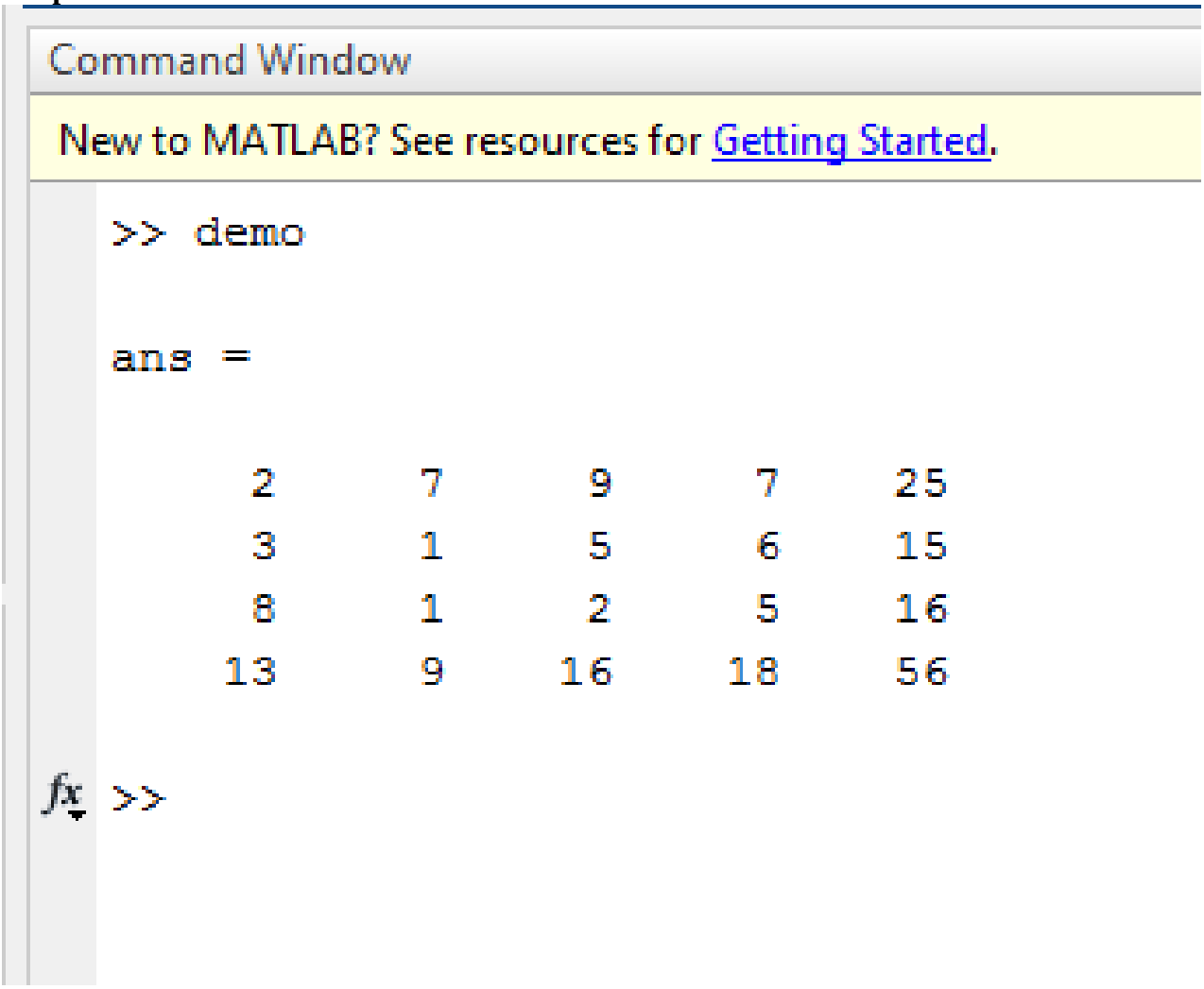


**k. [ [ S ; sum(S) ] [ sum(S,2) ; sum(S(:)) ] ]**

**Code:**



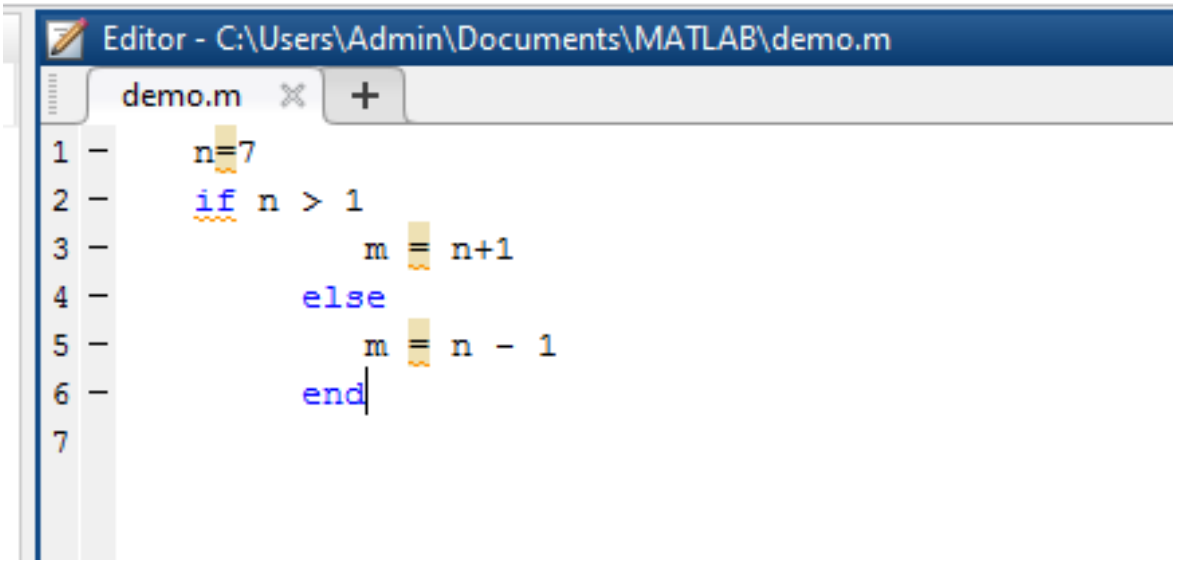

**Output**



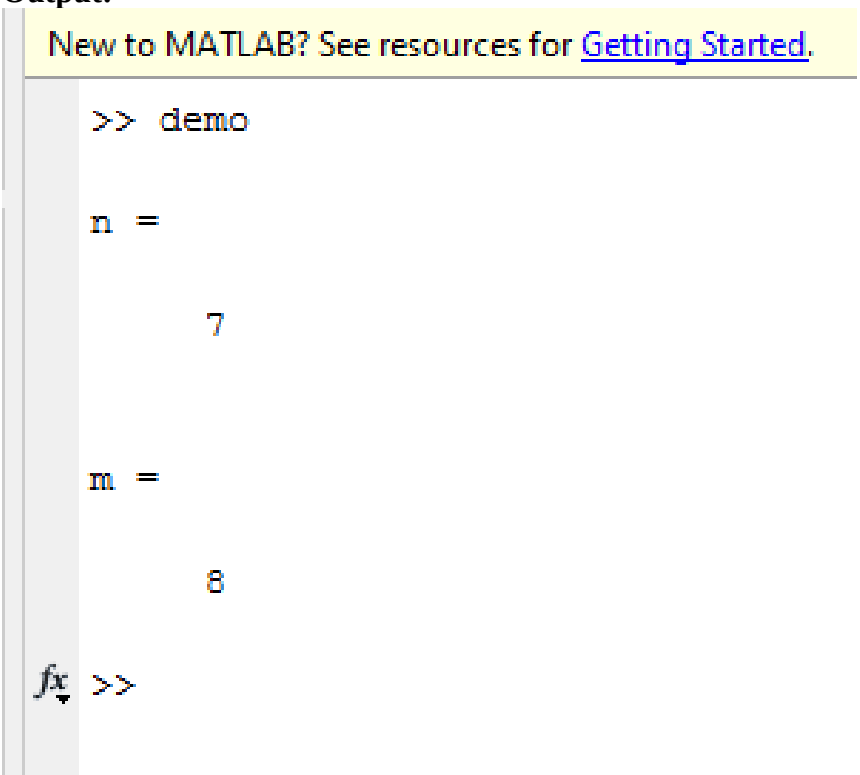
1. **if n > 1**
   **m = n+1**
   **else**
   **m = n - 1**
   **end**

**a. n = 7 m = ?**
**b. n = 0 m = ?**
**c. n = -10 m = ?**

**a. n = 7 m = ?**
**Code:**




**Output:**



**b. n = 0  m = ?**
**Code:**

Editor - C:\Users\Admin\Documents\MATLAB\demo.m

demo.m

```
1 -     n=0
2 -     if n > 1
3 -             m = n+1
4 -         else
5 -             m = n - 1
6 -         end
7
```

**Output:**

Command Window

New to MATLAB? See resources for Getting Started.

```
        8

>> demo

n =

     0


m =

    -1

>>
```

**c.n = -10 m = ?**
**Code:**

Editor - C:\Users\Admin\Documents\MATLAB\demo.m

demo.m

```
1 -     n=-10
2 -     if n > 1
3 -             m = n+1
4 -         else
5 -             m = n - 1
6 -         end
7
```

**Output:**

Command Window

New to MATLAB? See resources for Getting Started.

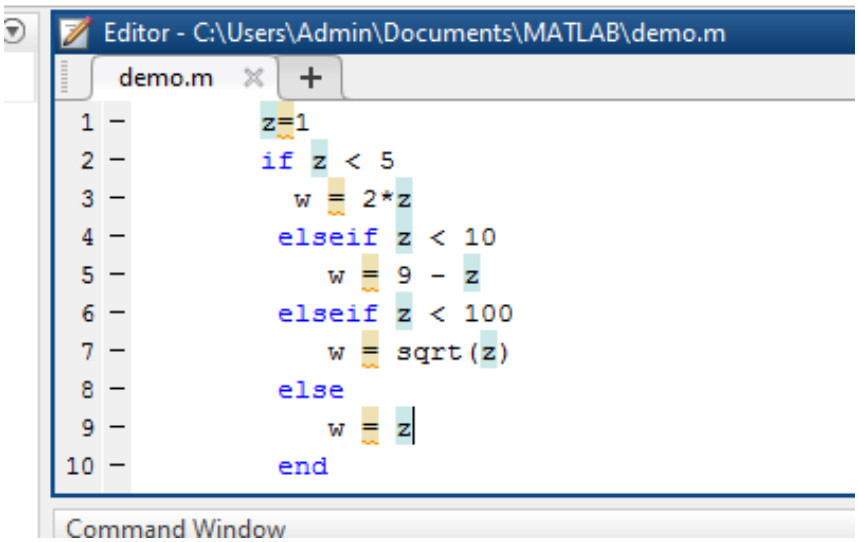
```
    -11

>> demo

n =

    -10


m =

    -11

>>
```

2. **if z < 5**
   **w = 2*z**
   **elseif z < 10**
   **w = 9 - z**
   **elseif z < 100**
   **w = sqrt(z)**
   **else**
   **w = z**
   **end**

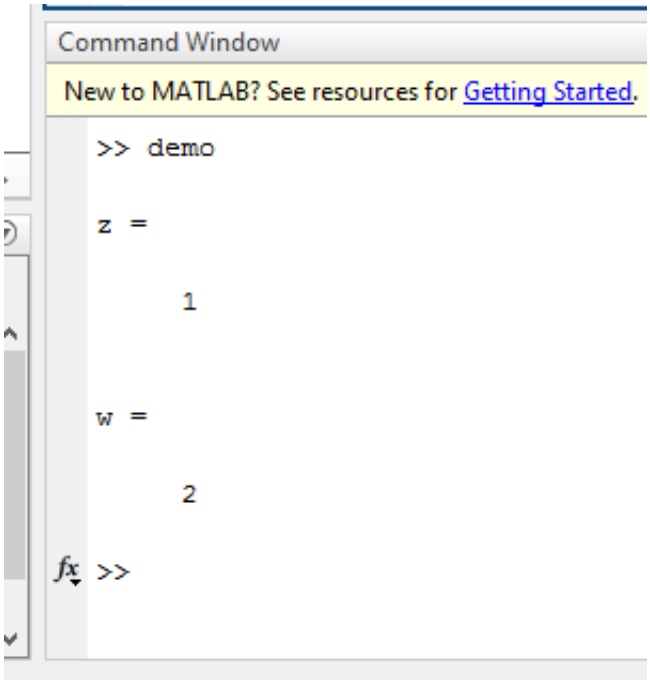
**a. z = 1 w = ?**
**b. z = 9 w = ?**
**c. z = 60 w = ?**
**d. z = 200 w = ?**

**a. z = 1 w = ?**

**Code:**

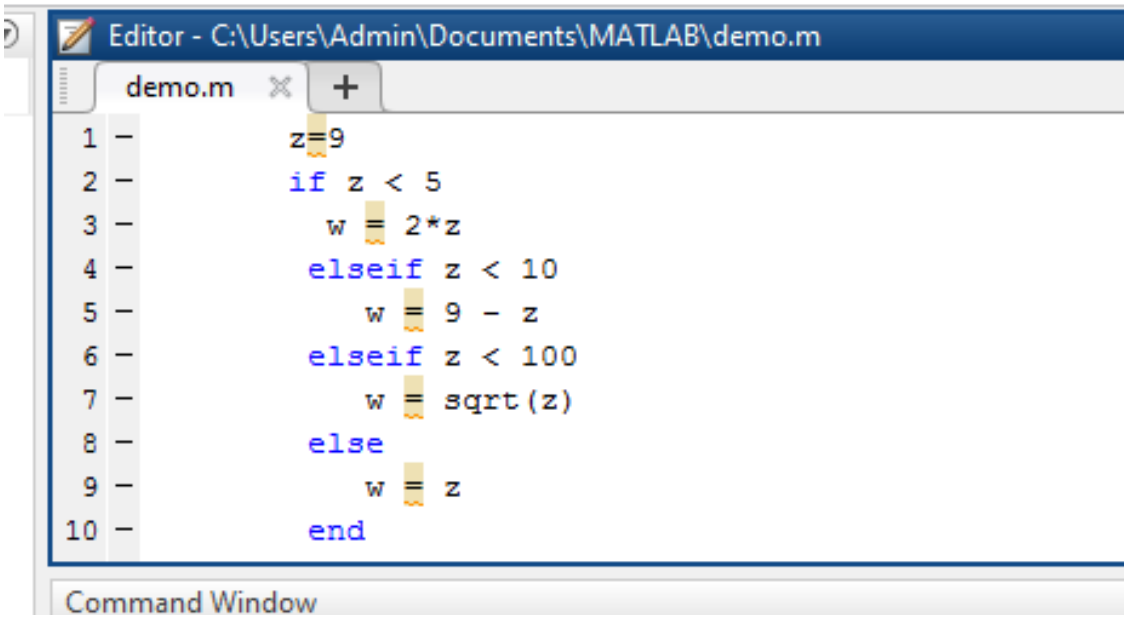



**Output:**

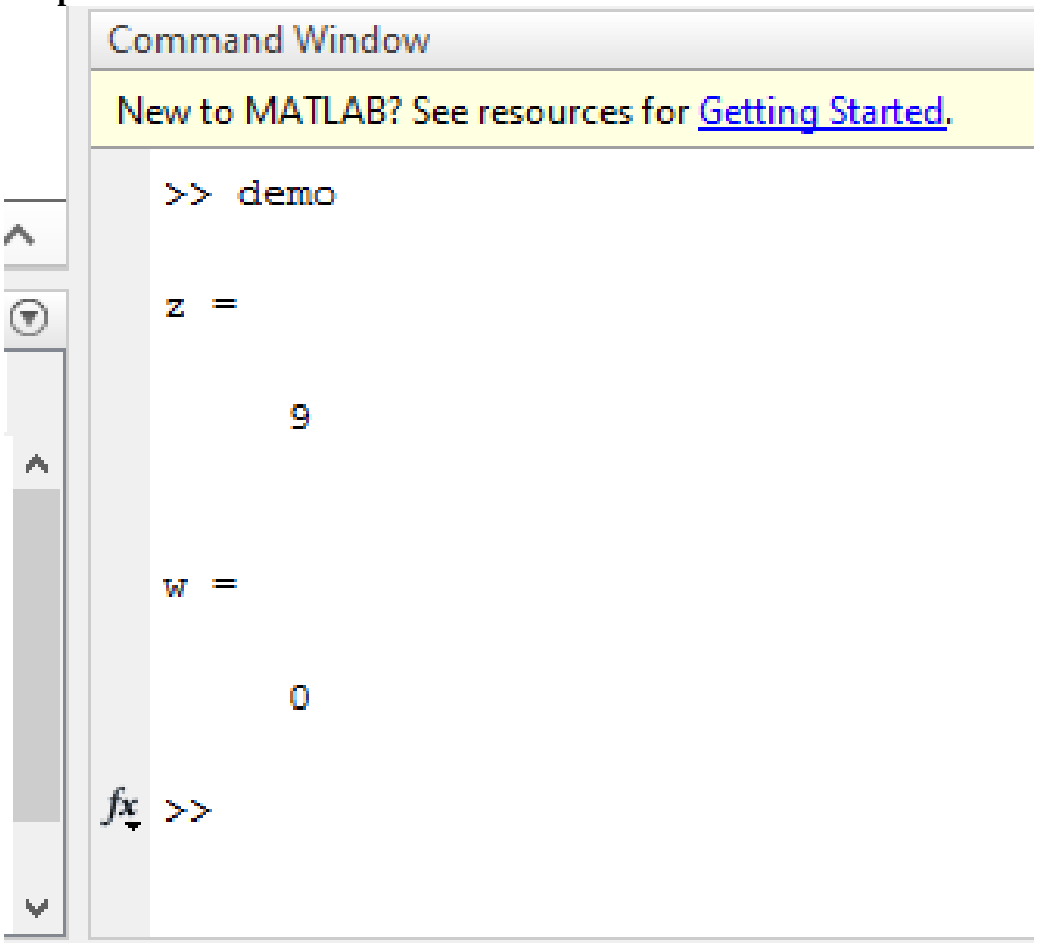


**b. . z = 9  w = ?**

**Code:**

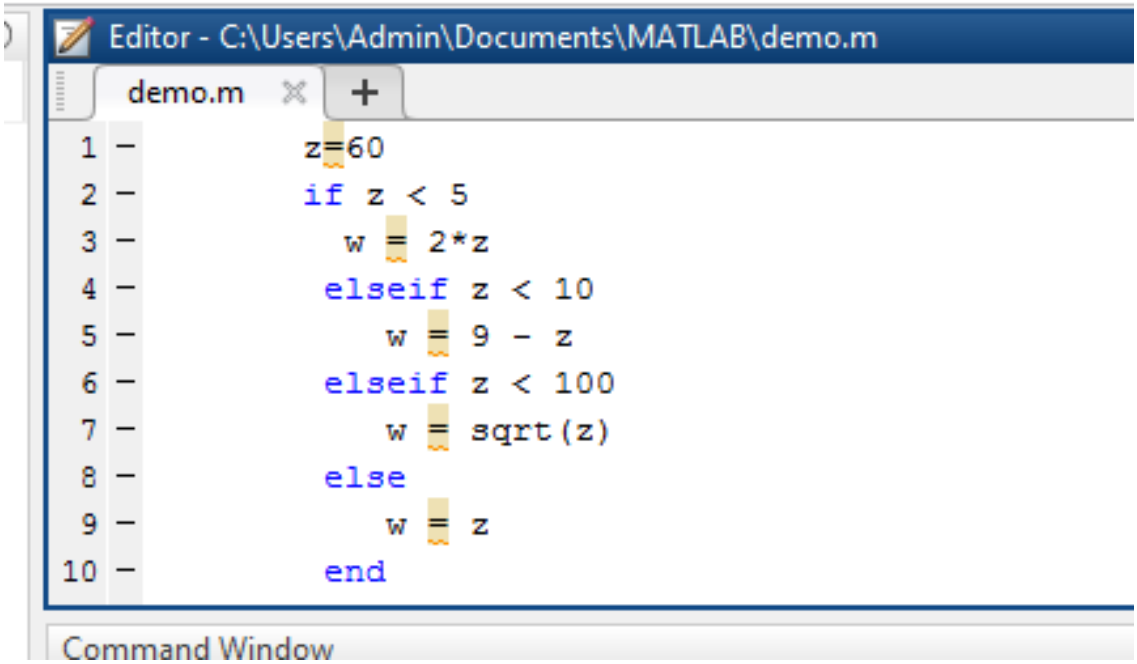



**Output:**

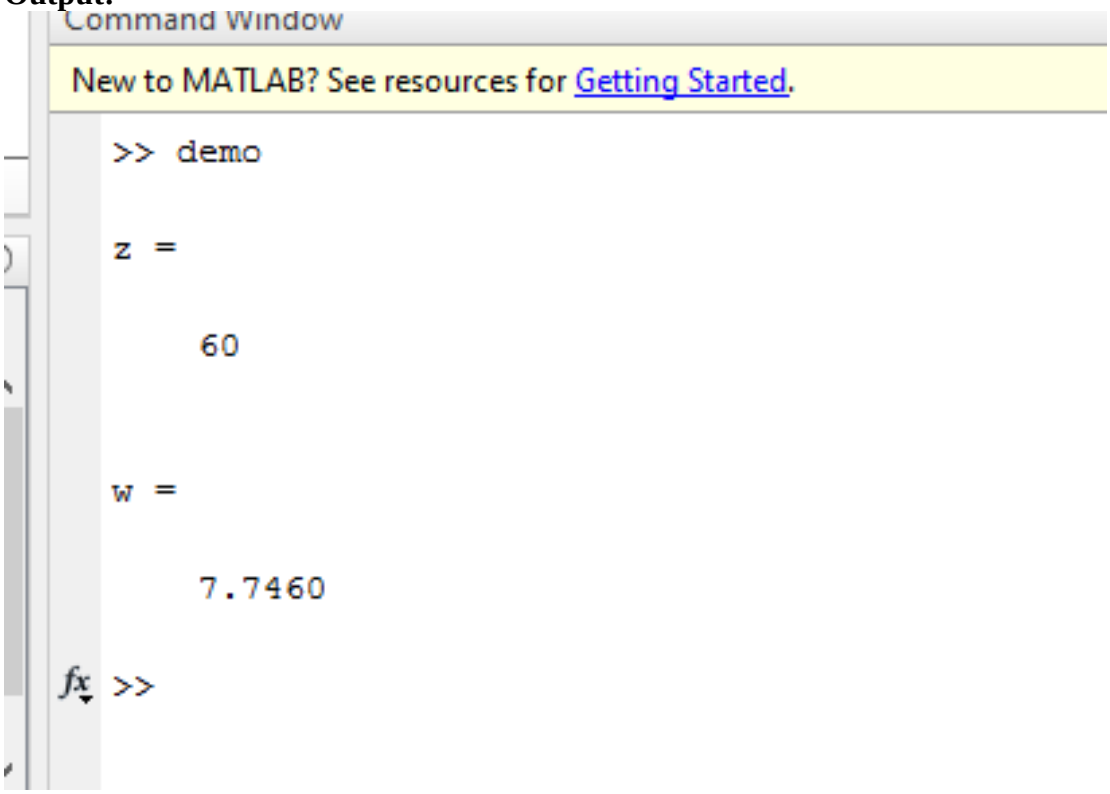


**c. z = 60  w = ?**

**Code:**

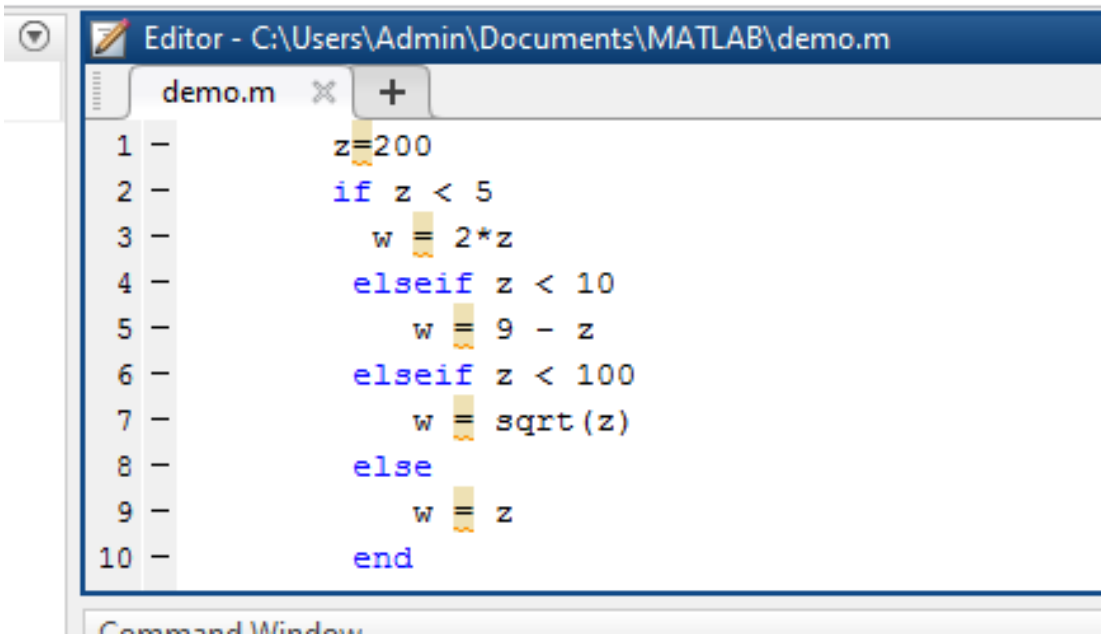



**Output:**

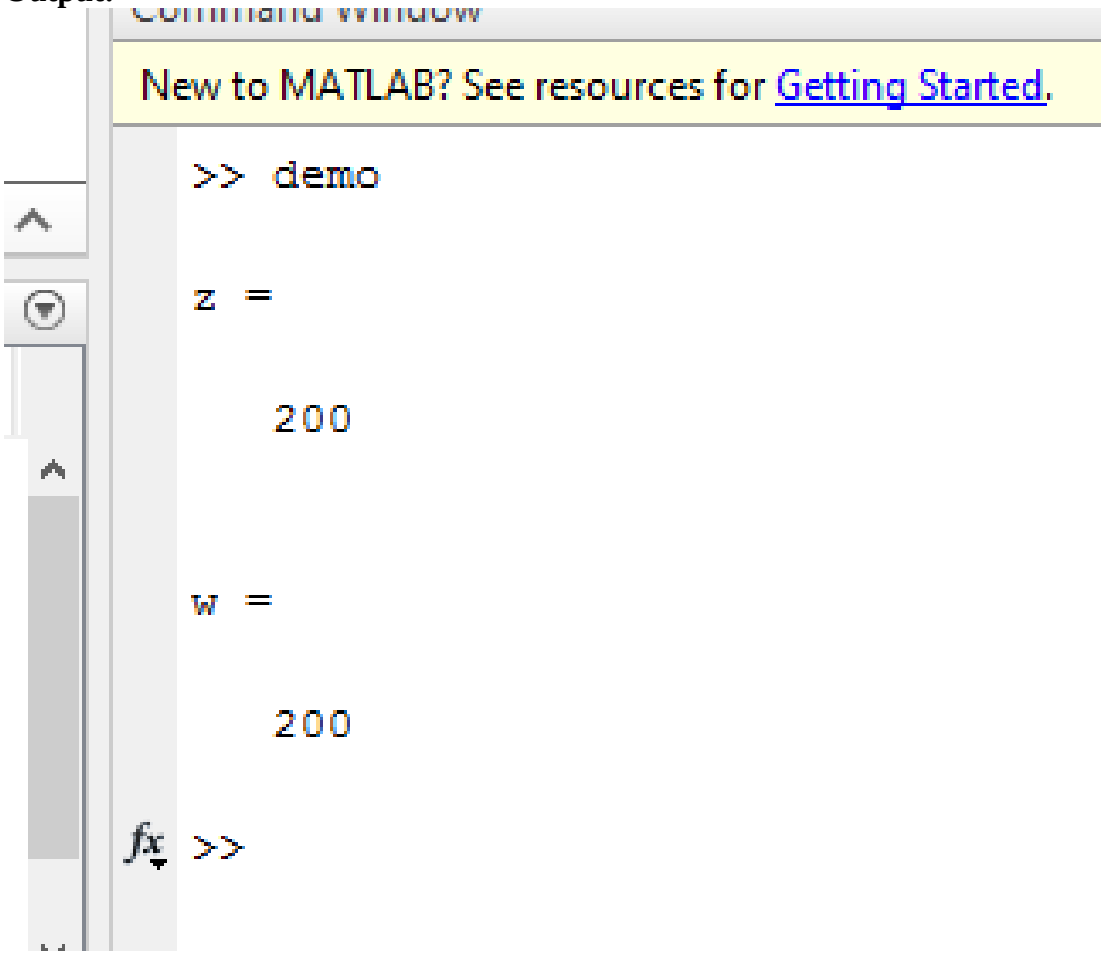


**d. z = 200  w = ?**

**Code:**



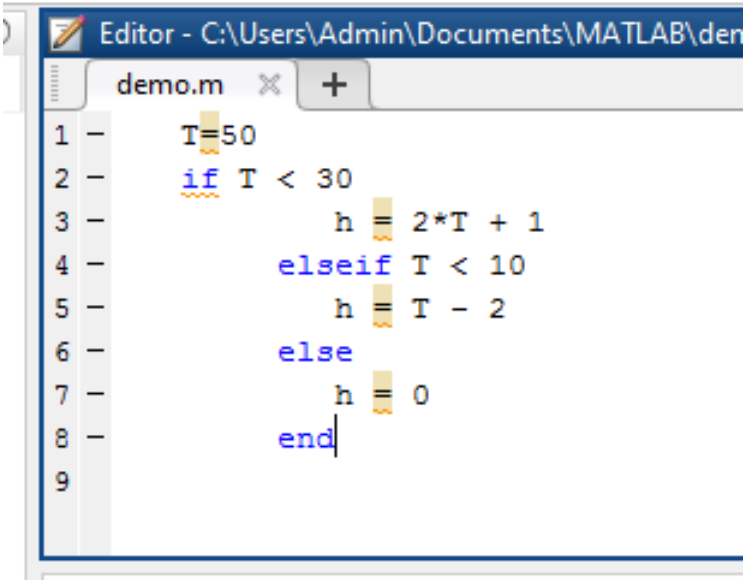

**Output:**



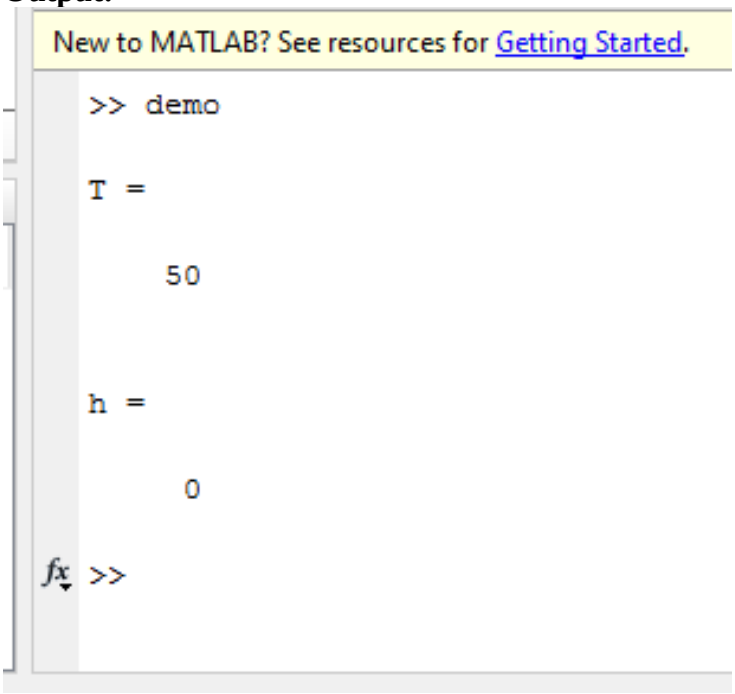
3. **if T < 30**
   **h = 2*T + 1**
   **elseif T < 10**
   **h = T - 2**
   **else**
   **h = 0**
   **end**

**a. T = 50 h = ?**
**b. T = 15 h = ?**
**c. T = 0 h = ?**

**a. . T = 50 h = ?**
**Code:**

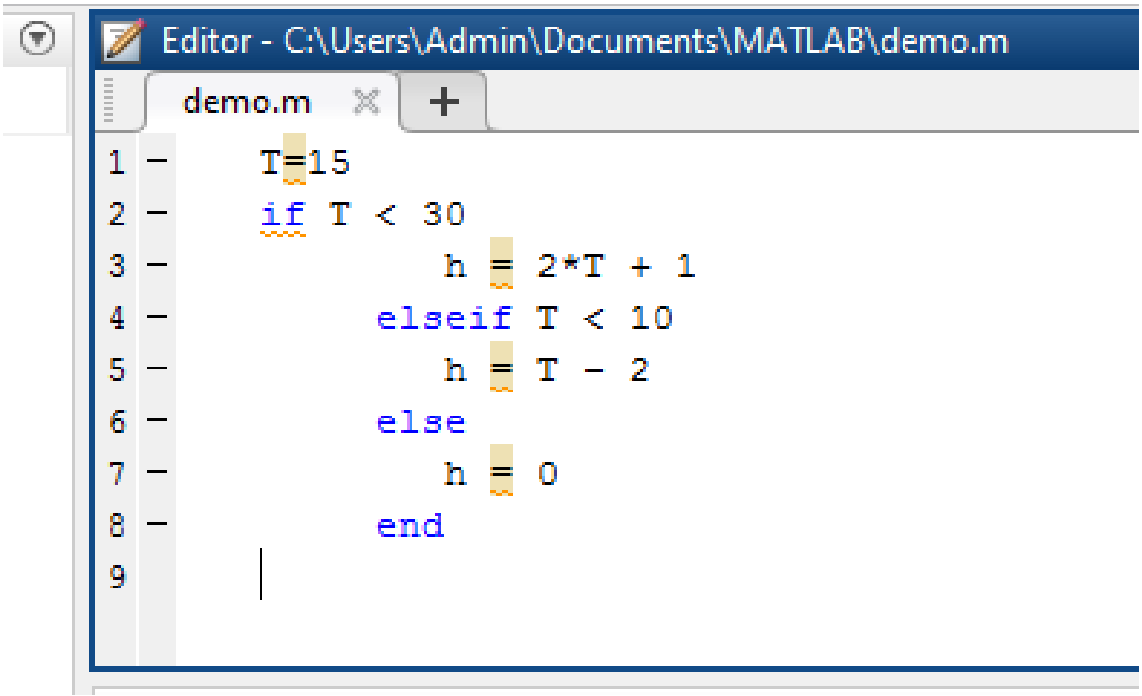



**Output:**

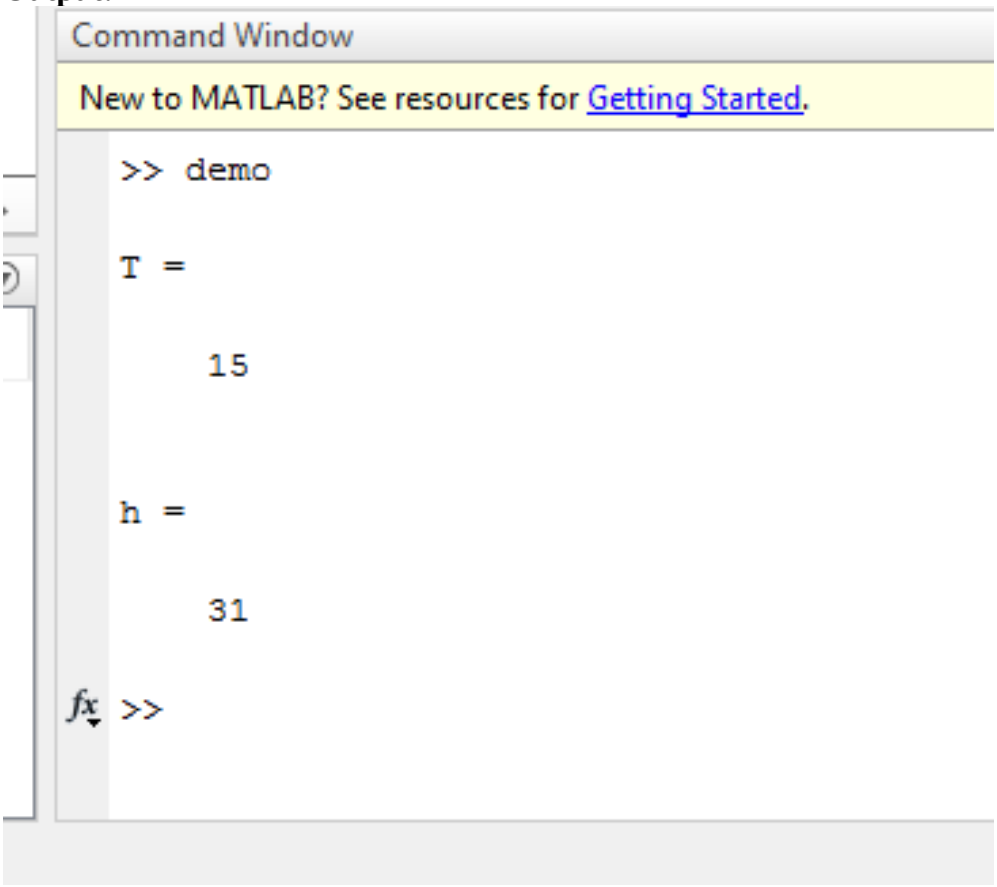


**b. T = 15  h = ?**

**Code:**

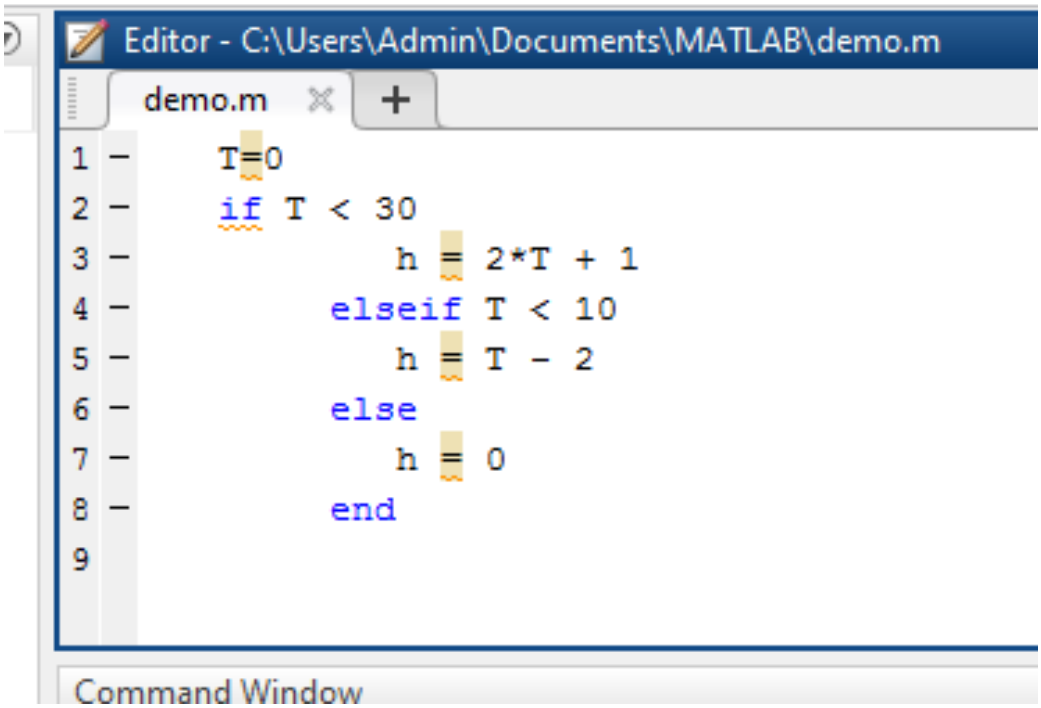



**Output:**

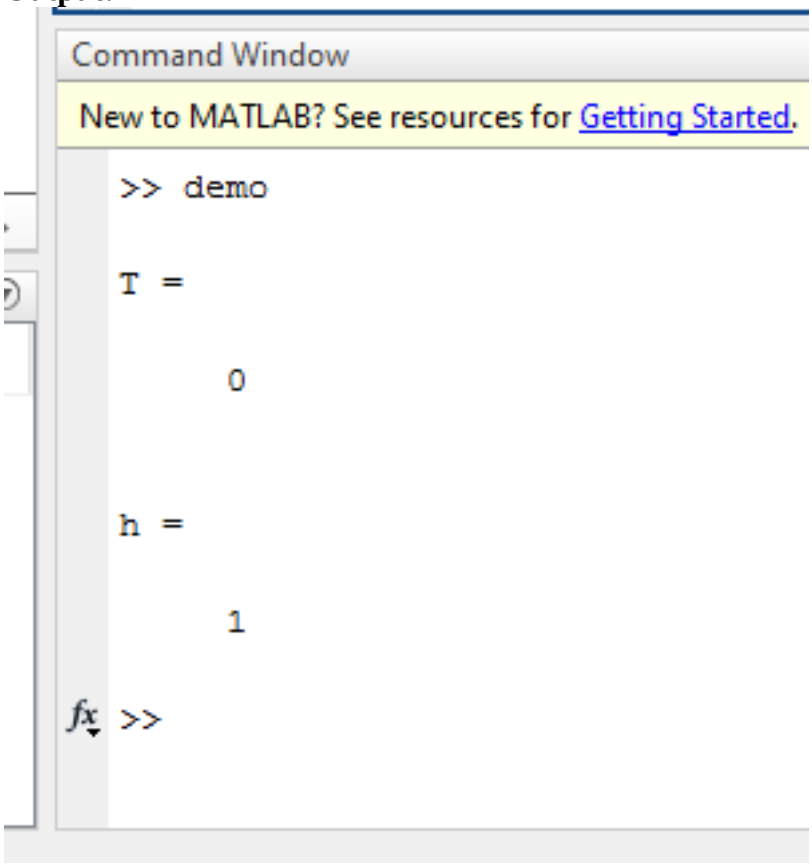


**c. T = 0  h = ?**

**Code:**




**Output:**

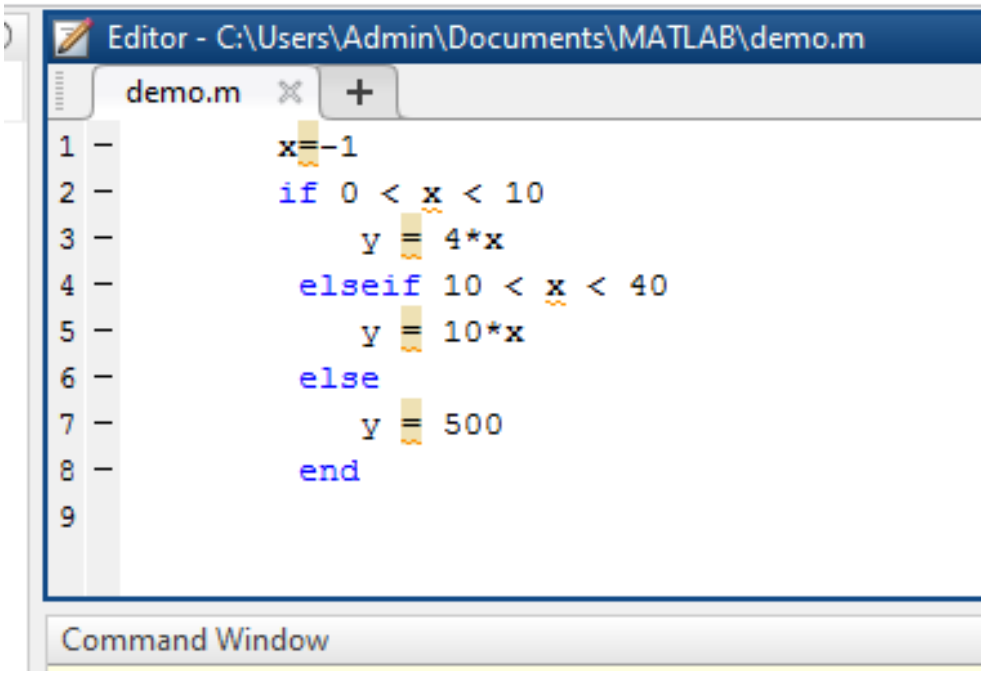


4. **if 0 < x < 10**
   **y = 4*x**
   **elseif 10 < x < 40**
   **y = 10*x**
   **else**
   **y = 500**
   **end**

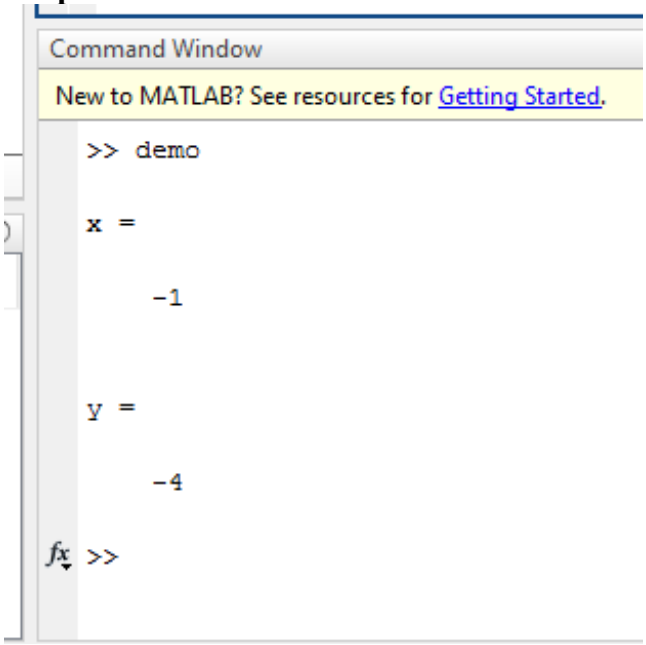
**a. x = -1 y = ?**
**b. x = 5 y = ?**
**c. x = 30 y = ?**
**d. x = 100 y = ?**

**a. . x = -1 y = ?**

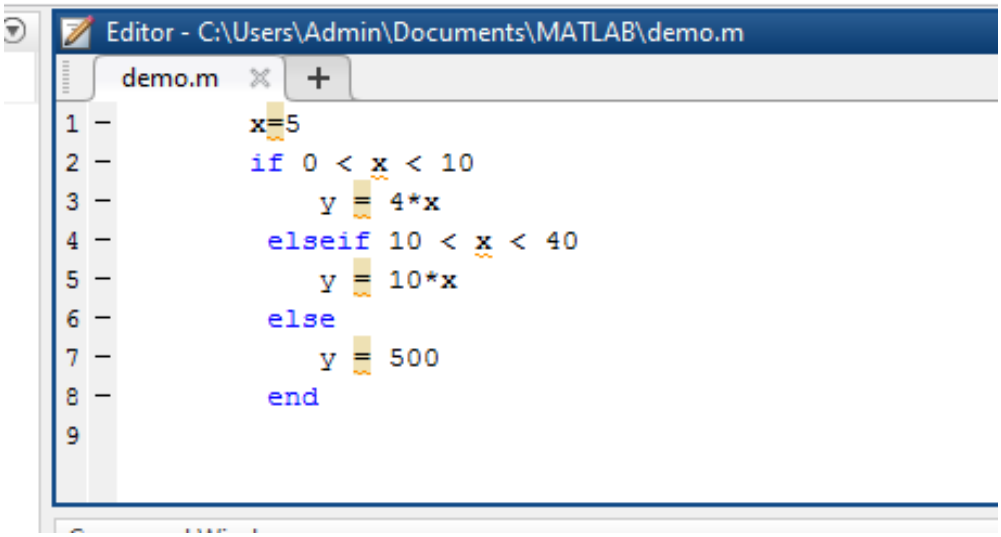
**Code:**



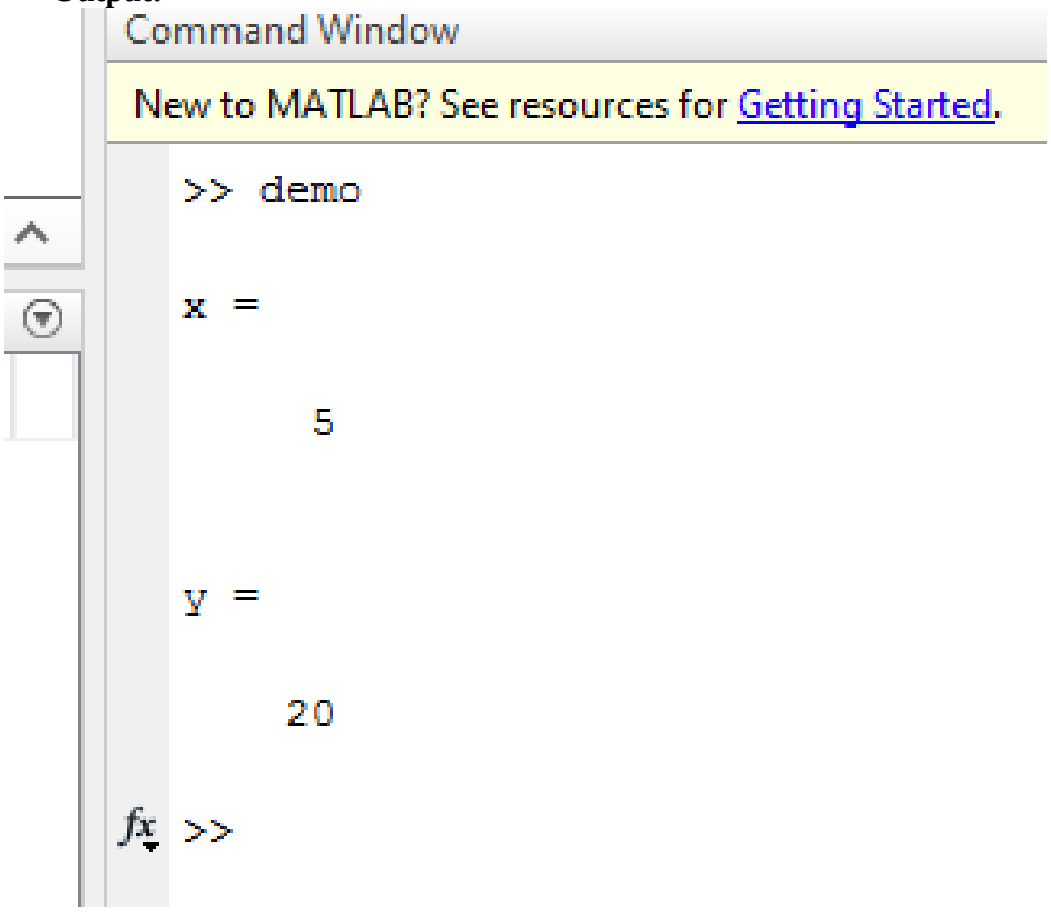

**Output:**



**b. x = 5  y = ?**
**Code:**

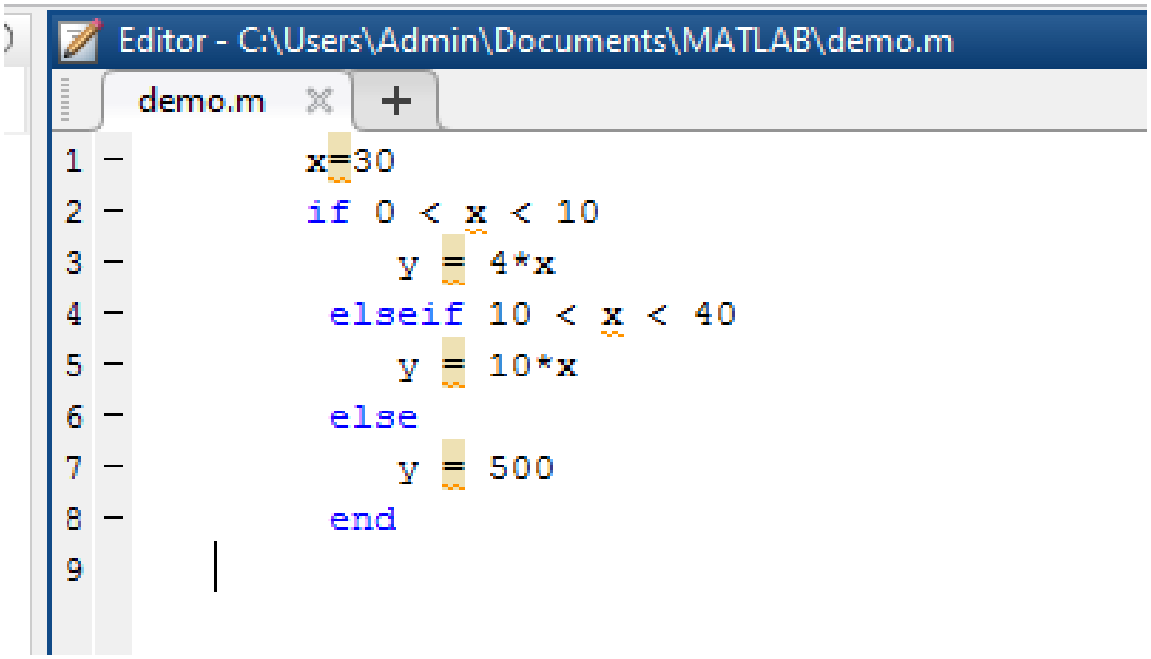



**Output:**

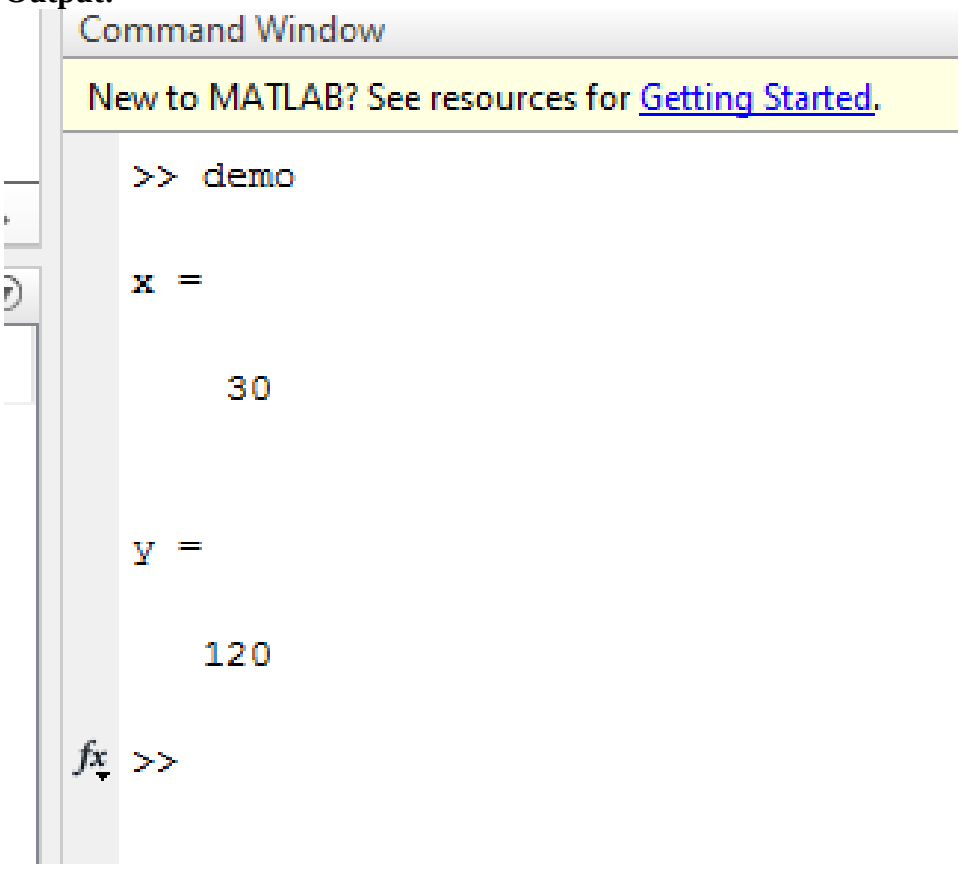


**c. . x = 30  y = ?**

**Code:**

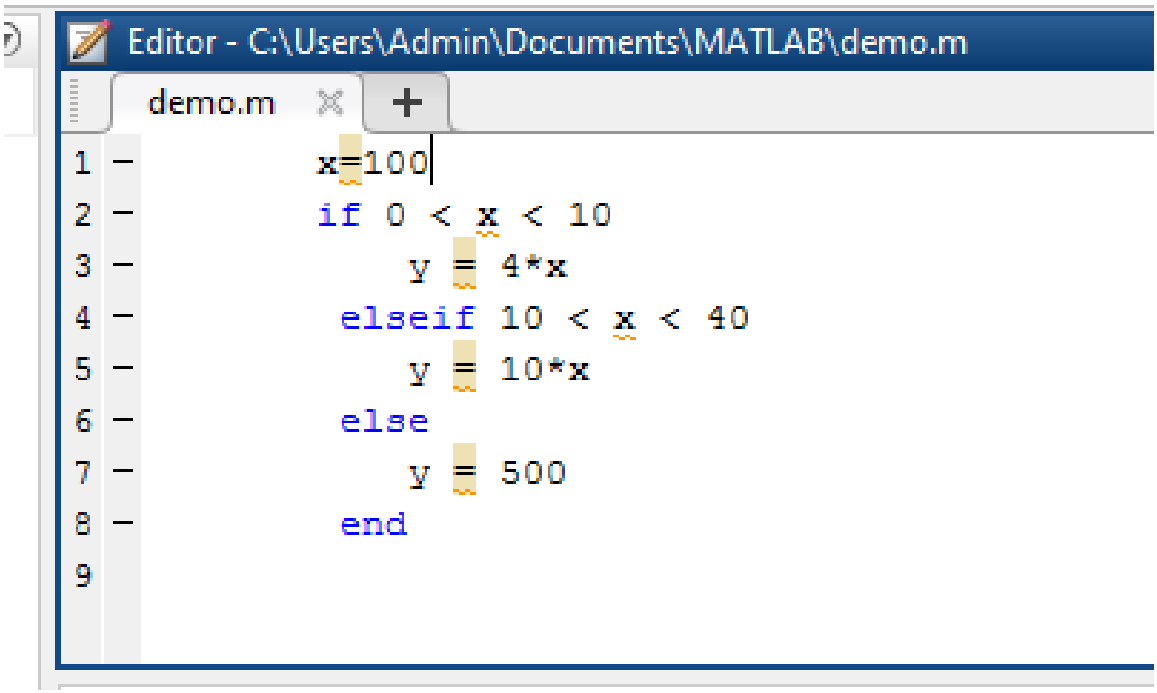



**Output:**

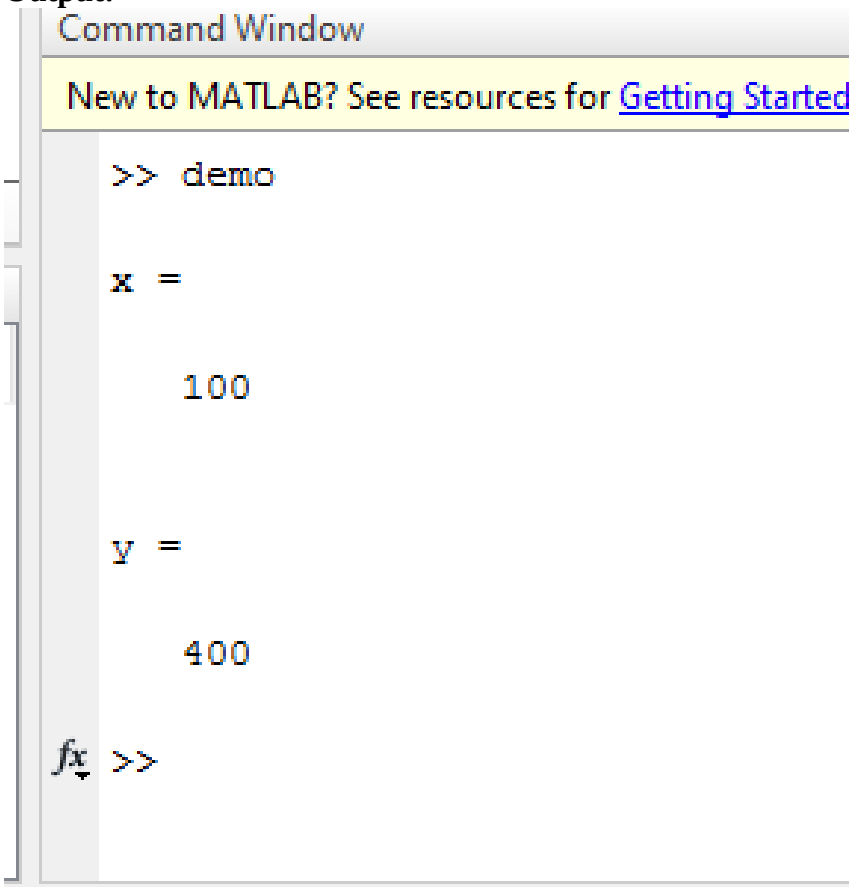


**d. . x = 100  y = ?**

**Code:**



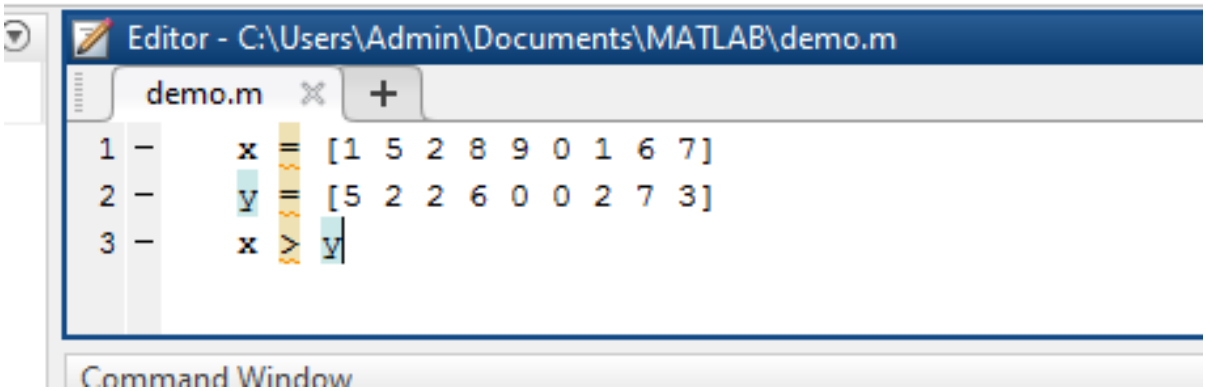

**Output:**



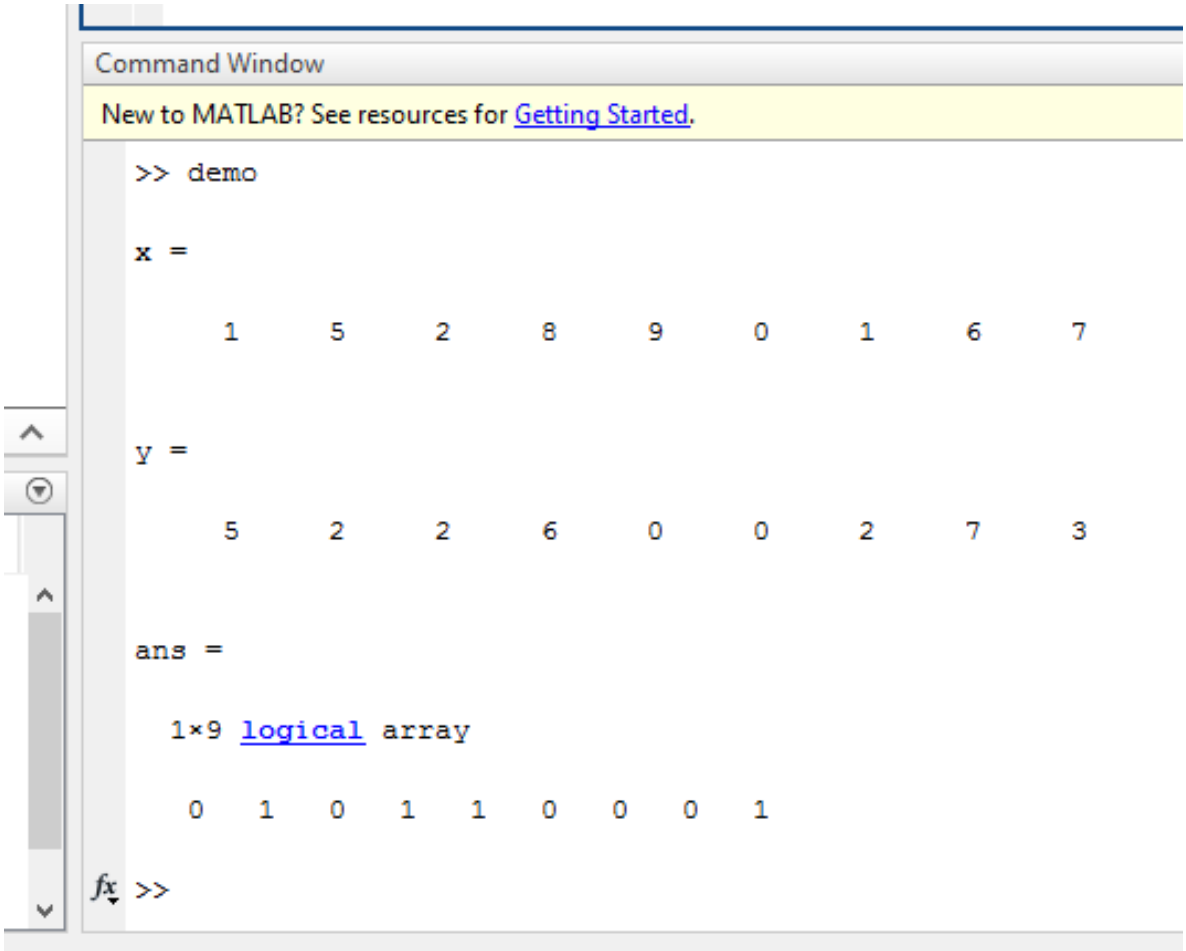
5. **Given that x = [1 5 2 8 9 0 1 6 7] and y = [5 2 2 6 0 0 2 7 3], execute and explain the results of the following commands:**

   a. **x > y**

**Code:**

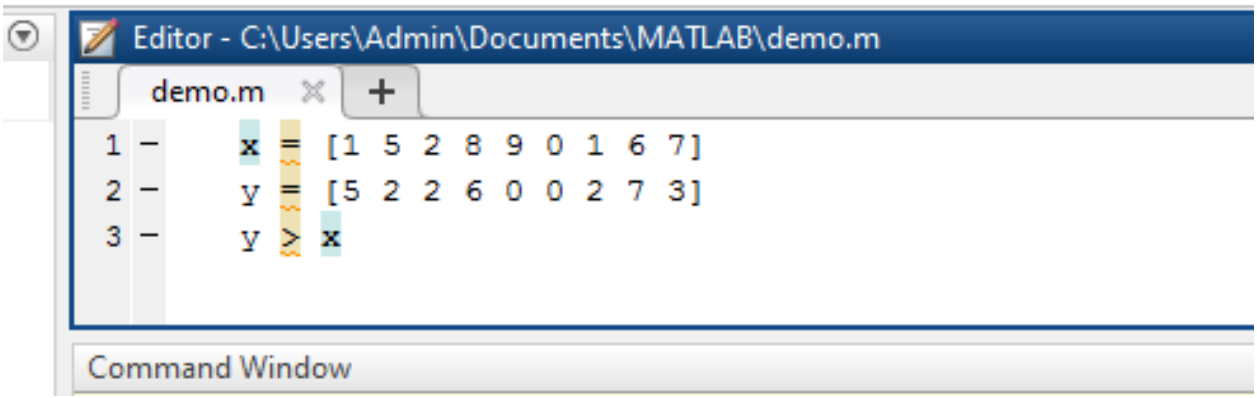



**Output:**

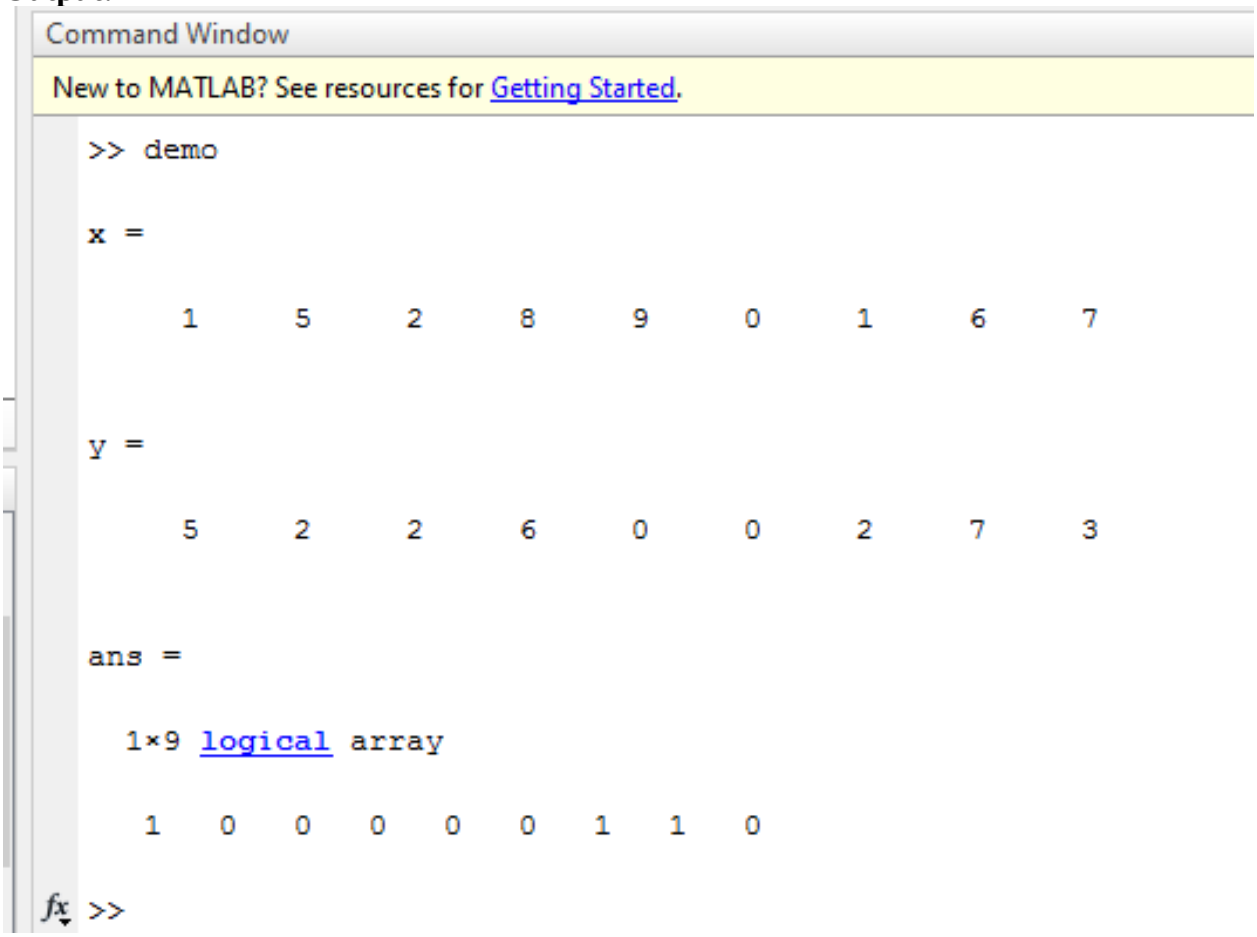


**b. y > x**

**Code:**




**Output:**



c. x == y

**Code:**

```
x = [1 5 2 8 9 0 1 6 7]
y = [5 2 2 6 0 0 2 7 3]
y == x
```

**Output:**

```
>> demo

x =

     1     5     2     8     9     0     1     6     7


y =

     5     2     2     6     0     0     2     7     3


ans =

  1×9 logical array

   0   0   1   0   0   1   0   0   0

>>
```

**d. x <= y**

**Code:**

MATLAB

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
1 -     x = [1 5 2 8 9 0 1 6 7]
2 -     y = [5 2 2 6 0 0 2 7 3]
3 -     x <= y
4
```

**Output:**

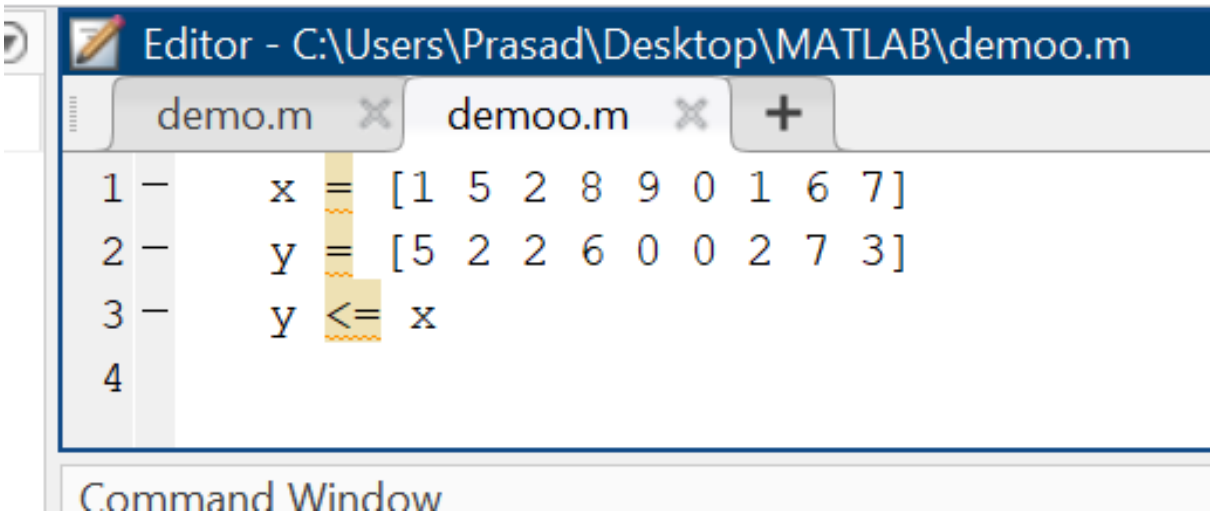
Command Window

New to MATLAB? See resources for Getting Started.

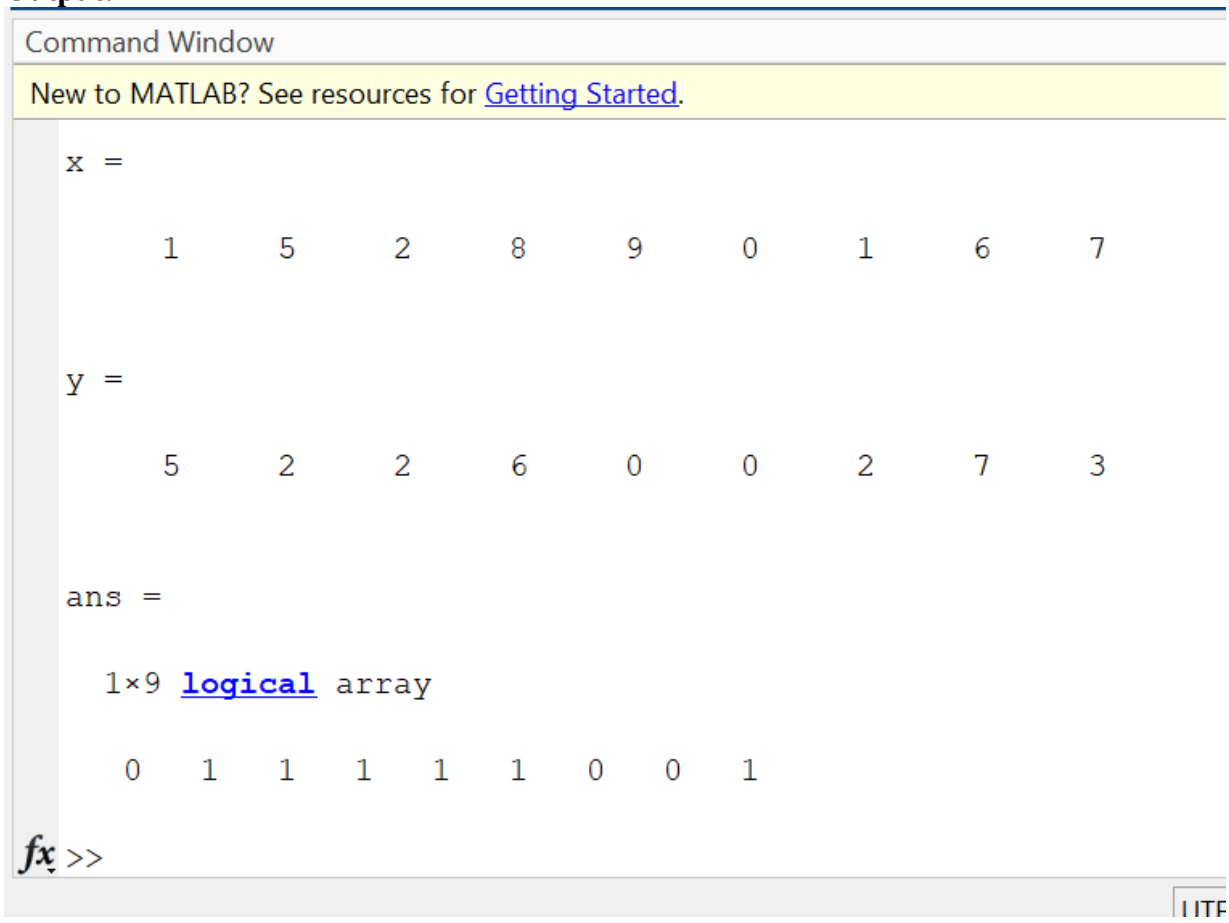
```
x =

     1     5     2     8     9     0     1     6     7


y =

     5     2     2     6     0     0     2     7     3


ans =

  1×9 logical array

   1   0   1   0   0   1   1   1   0

fx >>
```

**e. y <= x**

**Code:**

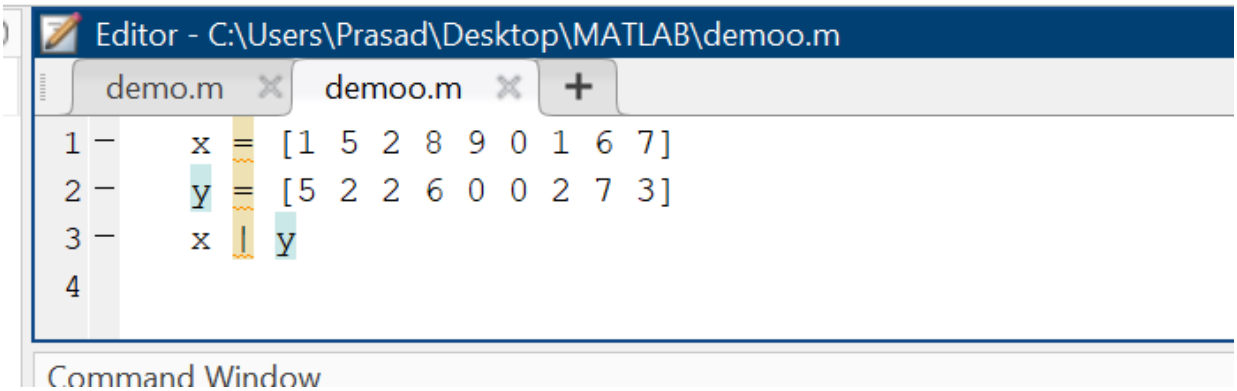



**Output:**



**f. x | y**

**Code:**




**Output:**

**g. x & y**

**Code:**

MATLAB

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
1 -    x = [1 5 2 8 9 0 1 6 7]
2 -    y = [5 2 2 6 0 0 2 7 3]
3 -    x & y
4
```

**Output:**

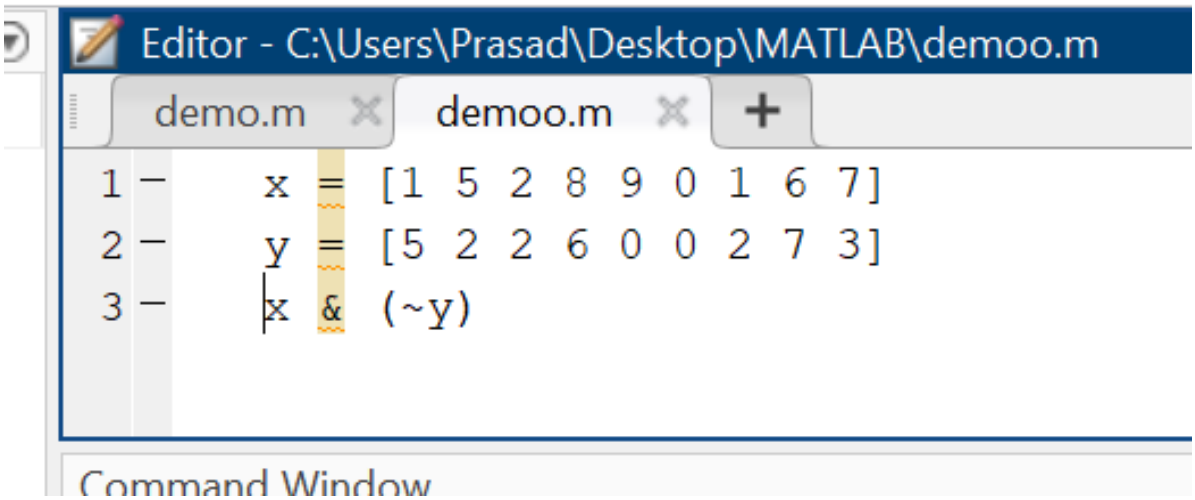
Command Window

New to MATLAB? See resources for Getting Started.

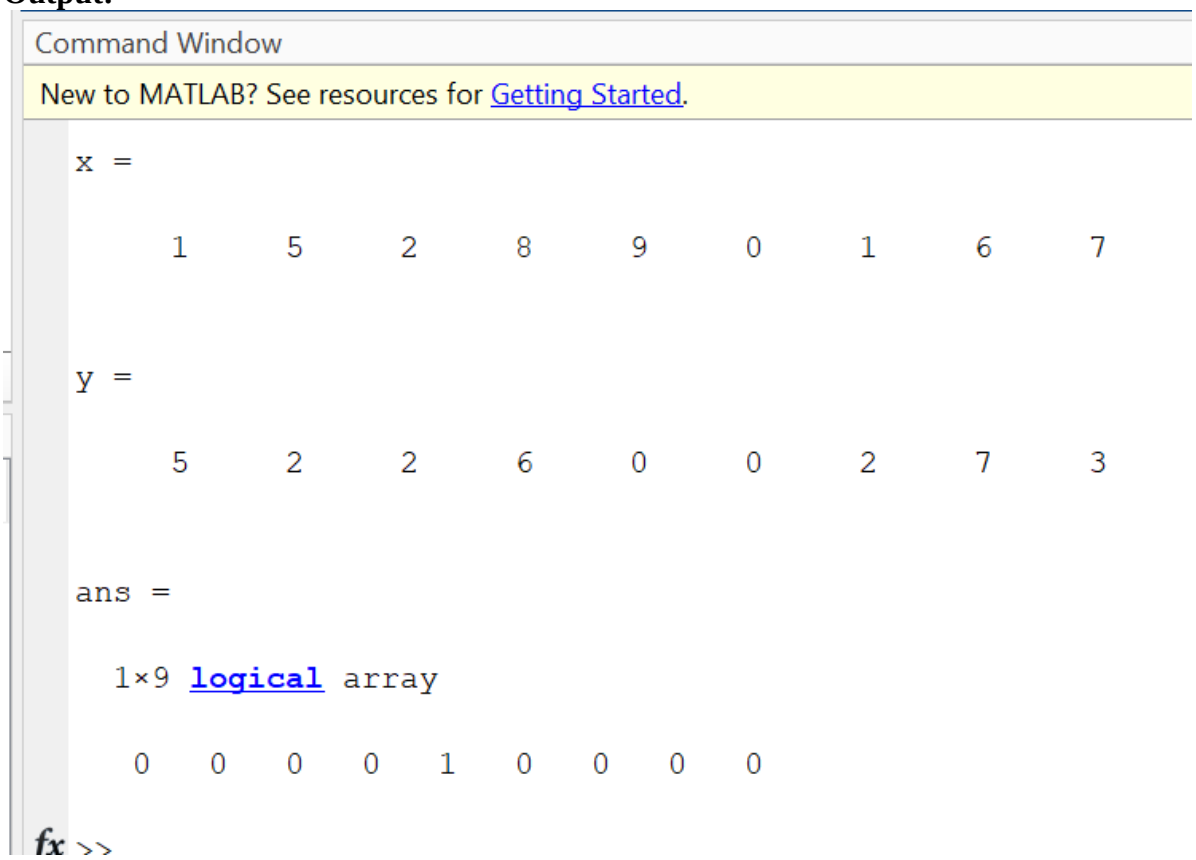
```
x =

     1     5     2     8     9     0     1     6     7


y =

     5     2     2     6     0     0     2     7     3


ans =

  1×9 logical array

   1   1   1   1   0   0   1   1   1

fx >>
```

**h. x & (~y)**

**Code:**

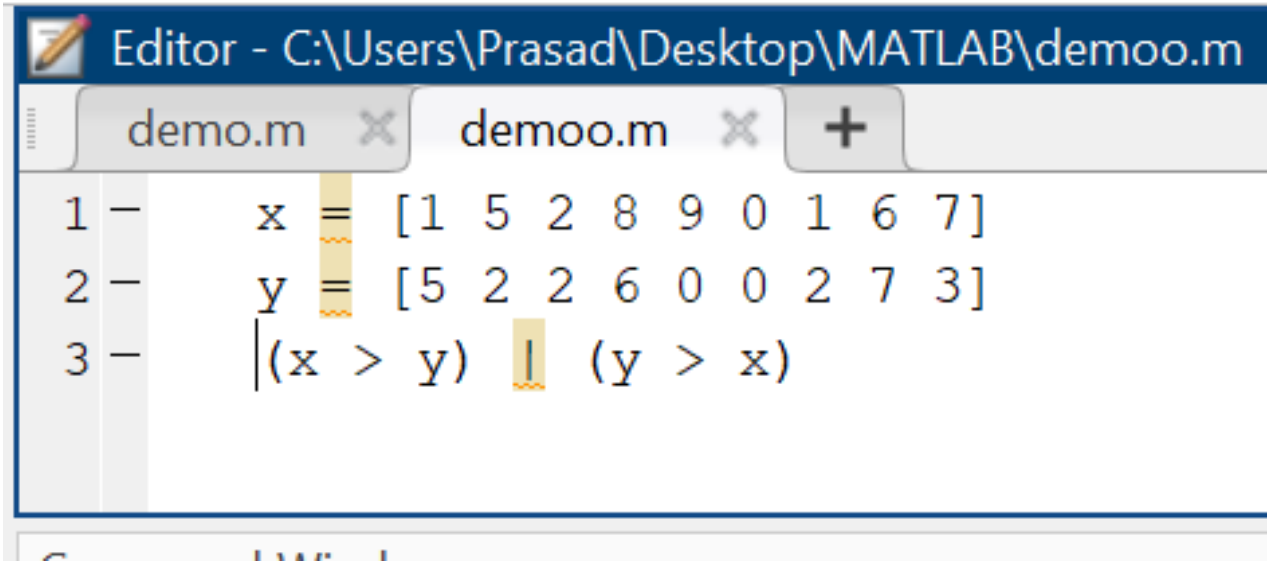



**Output:**



i. **(x > y) | (y > x)**

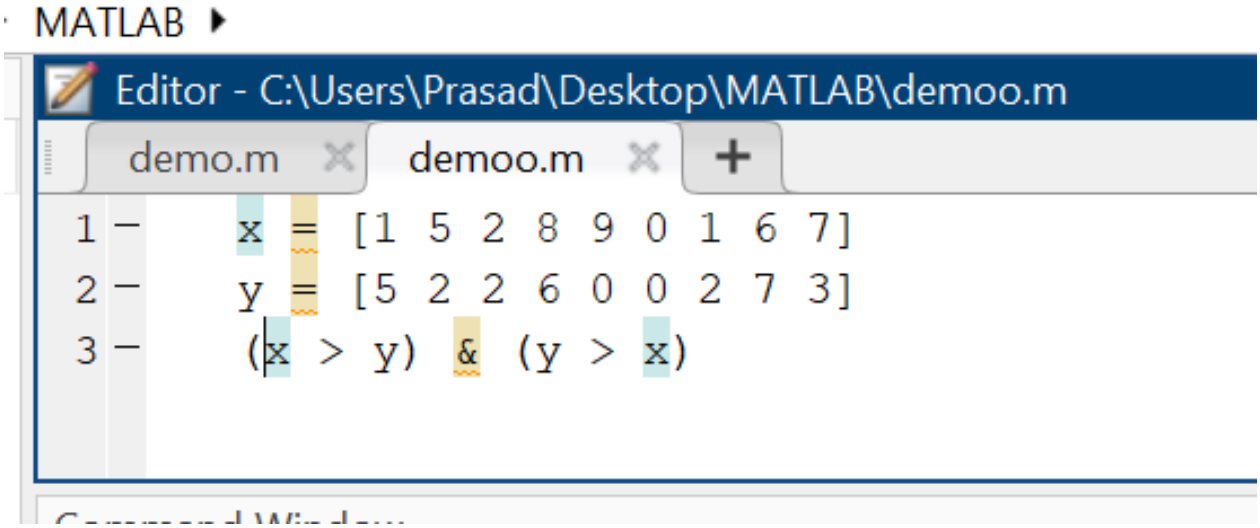
**Code:**



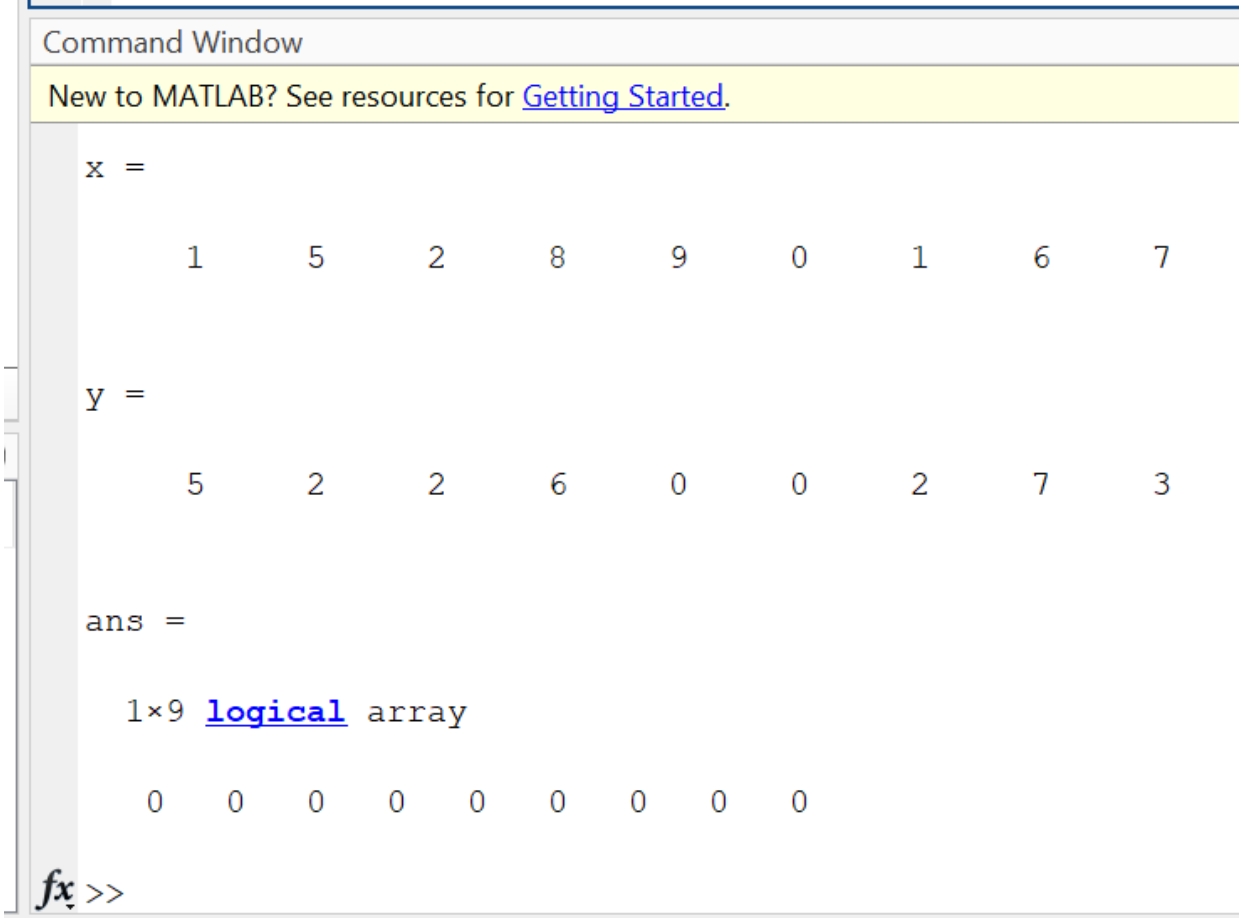

**Output:**

**j. (x > y) & (y > x)**

**Code:**




**Output:**



6. **The exercises here show the techniques of logical-indexing (indexing with 0-1 vectors). Given x = 1:10 and y = [3 1 5 6 8 2 9 4 7 0], execute and interpret the results of the following commands:**

**a. (x > 3) & (x < 8)**

**Code:**

MATLAB

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
1 -    x = 1:10
2 -    y = [3 1 5 6 8 2 9 4 7 0]
3 -    (x > 3) & (x < 8)
```

Command Window

**Output:**

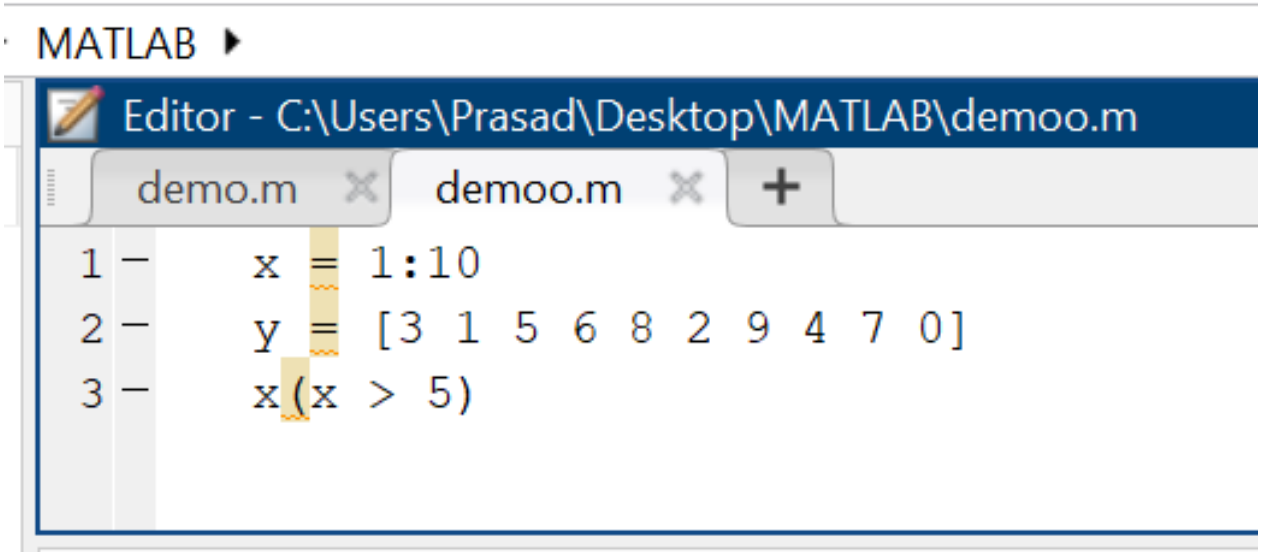
Command Window

New to MATLAB? See resources for Getting Started.

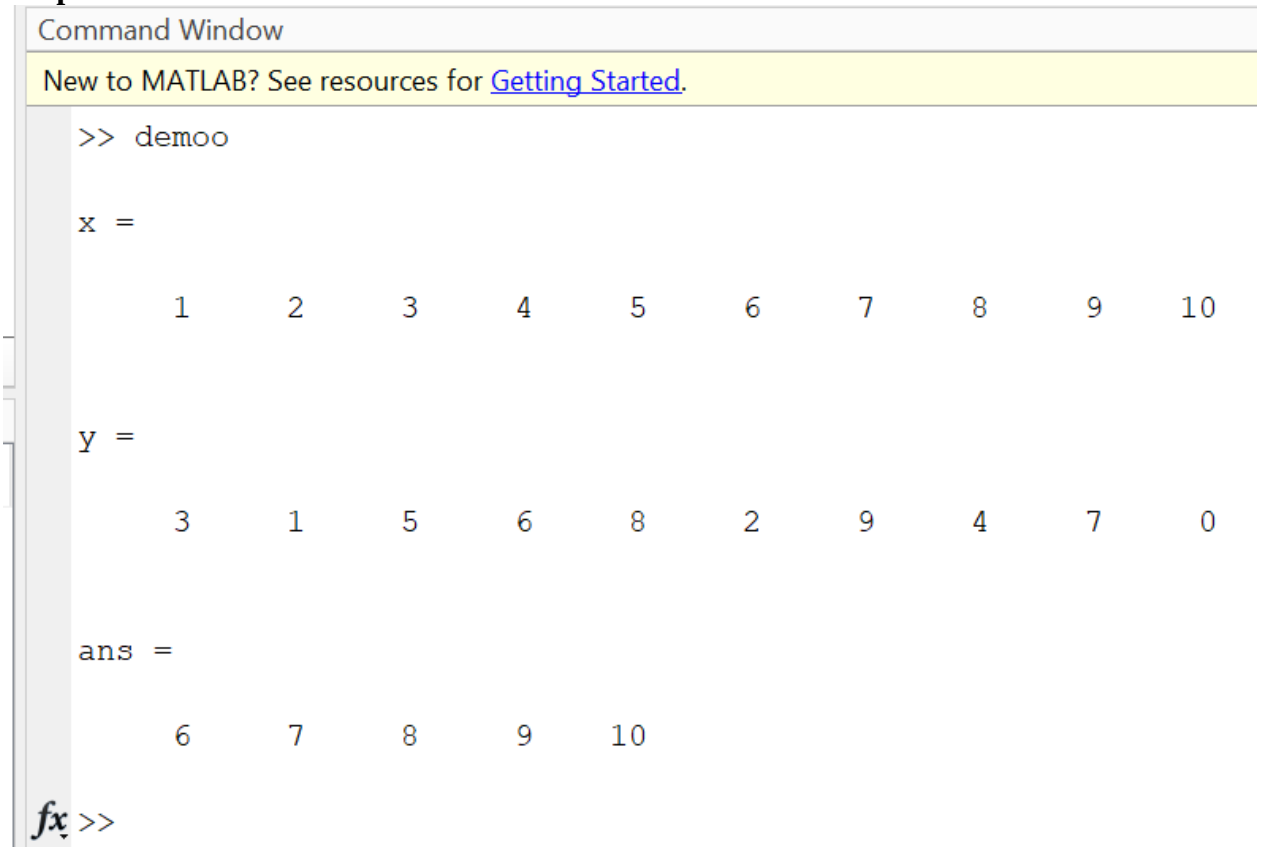
```
x =

     1     2     3     4     5     6     7     8     9    10


y =

     3     1     5     6     8     2     9     4     7     0


ans =

  1×10 logical array

   0   0   0   1   1   1   1   0   0   0

fx >>
```

**b.x(x > 5)**

**Code:**




**Output:**



**c. y(x <= 4)**

**Code:**

MATLAB

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
1 -     x = 1:10
2 -     y = [3 1 5 6 8 2 9 4 7 0]
3 -     y(x <= 4)
```

**Output:**

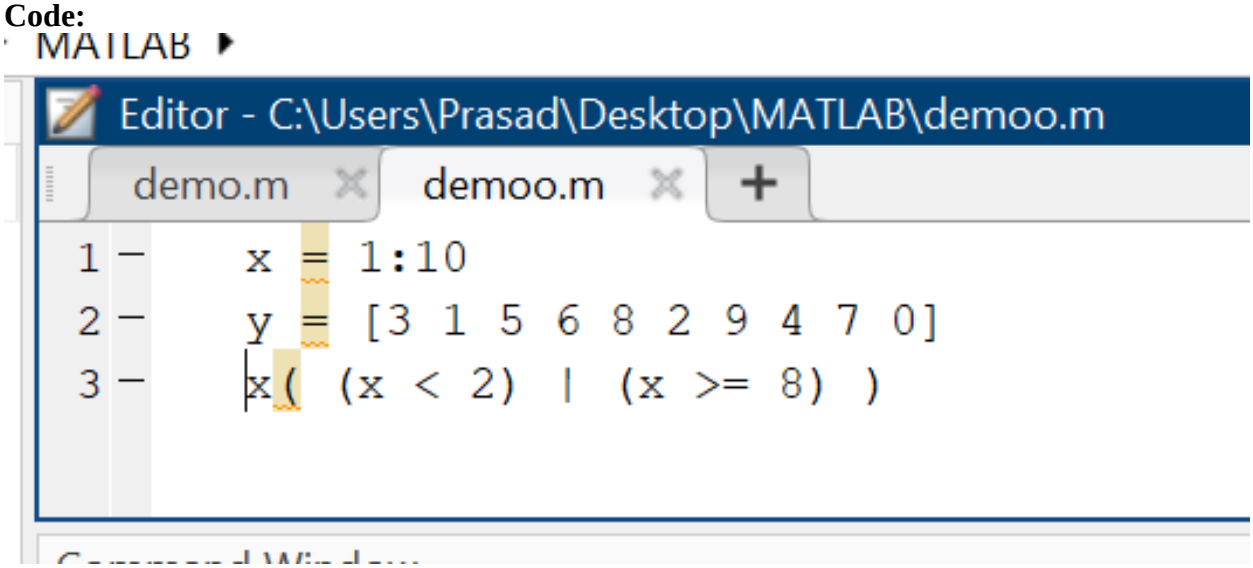
Command Window

New to MATLAB? See resources for Getting Started.

```
>> demoo

x =

     1     2     3     4     5     6     7     8     9    10


y =

     3     1     5     6     8     2     9     4     7     0


ans =

     3     1     5     6

fx >>
```

**d.x( (x < 2) | (x >= 8) )**

**Code:**




**Output:**

**e. y( (x < 2) | (x >= 8) )**

**Code:**

MATLAB

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
1 -     x = 1:10
2 -     y = [3 1 5 6 8 2 9 4 7 0]
3 -     y( (x < 2) | (x >= 8) )
```

**Output:**

Command Window

New to MATLAB? See resources for Getting Started.

```
>> demoo

x =

     1     2     3     4     5     6     7     8     9    10


y =

     3     1     5     6     8     2     9     4     7     0


ans =

     3     4     7     0

fx >>
```

**f. x(y < 0)**

**Code:**


**Output:**

New to MATLAB? See resources for Getting Started.

```
>> demoo

x =

     1     2     3     4     5     6     7     8     9    10


y =

     3     1     5     6     8     2     9     4     7     0


ans =

  1×0 empty double row vector

fx >>
```

**Assignment 4** **Date of Submission: 12/08/2022**

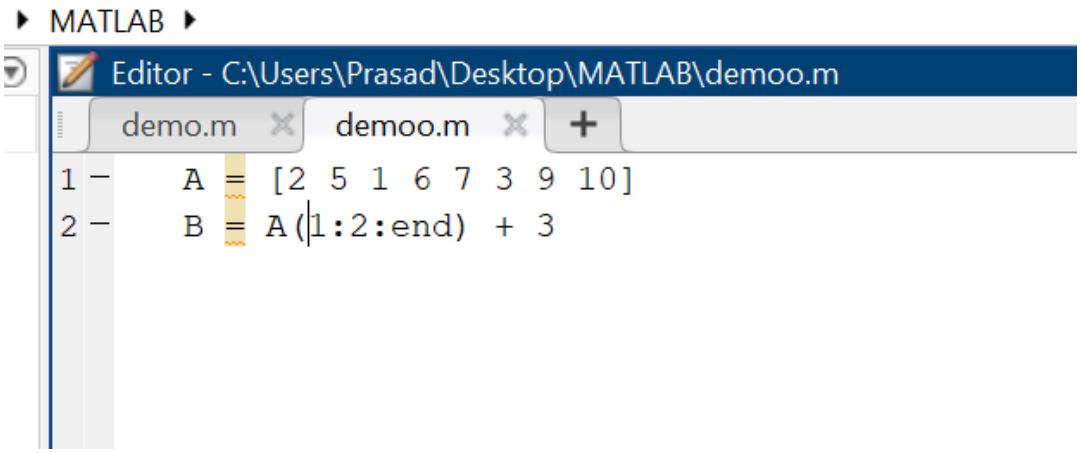
**In-built Functions of MATLAB**

1. **A = [2 5 1 6 7 3 9 10], add 3 to the odd indexed numbers.**

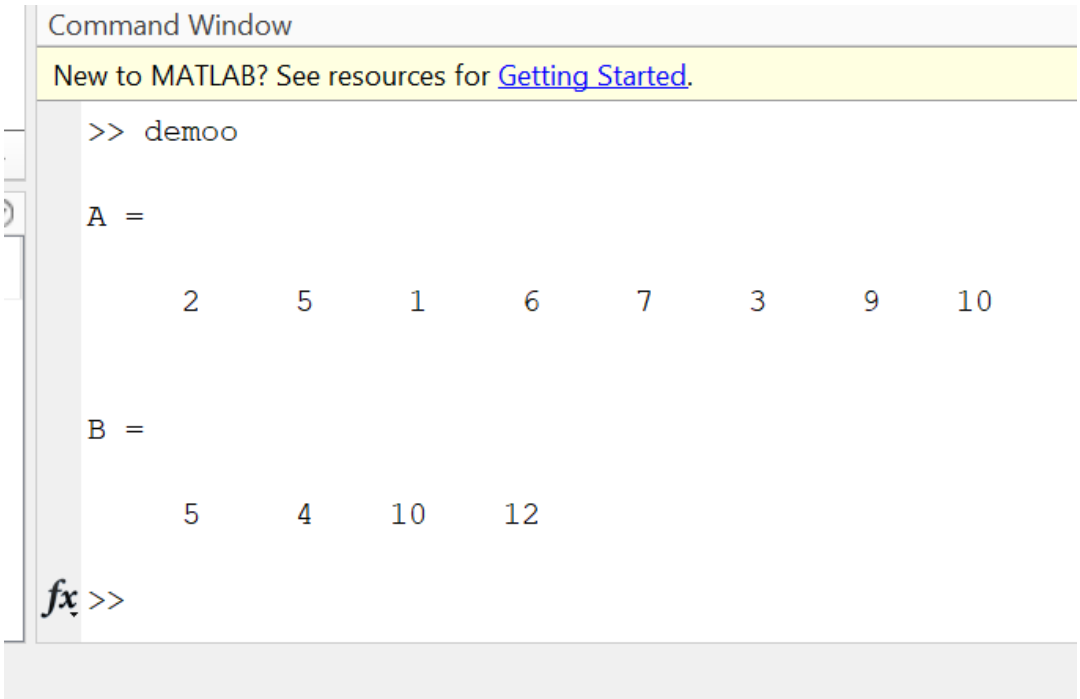
**Explanation:**

By using parenthesis and colon operator access the index required that and add 3,
B=A(1:2:end)+3

**Code:**




**Output:**

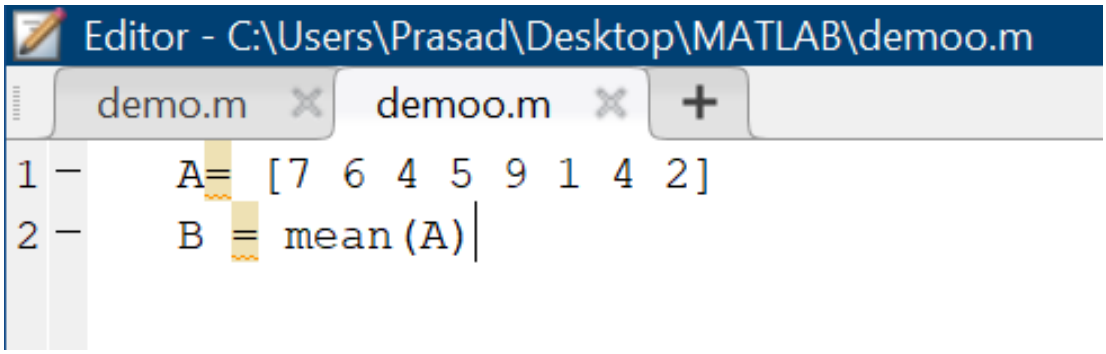


2. Create a row vector [7 6 4 5 9 1 4 2] and calculate the mean of it.

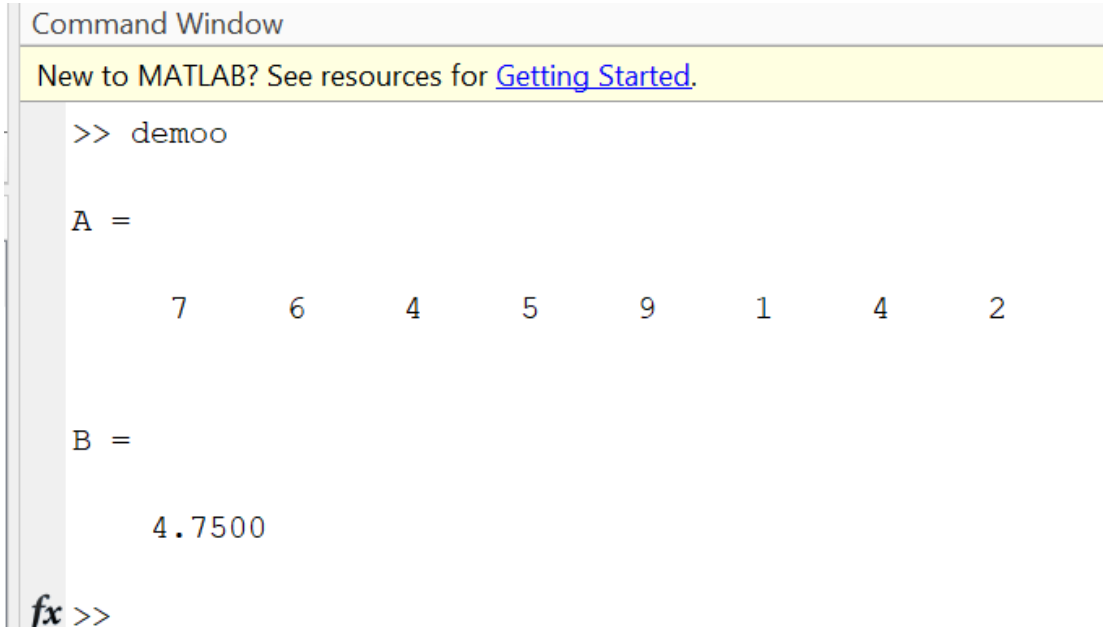
**Explanation:**

Use the function mean() to calculate the mean of the row vector.

**Code:**




**Output:**



3. Evaluate the following MATLAB expressions by hand and use MATLAB to check the answers. Explain **round(), floor()** and **ceil().**

i. 2 / 2 * 3

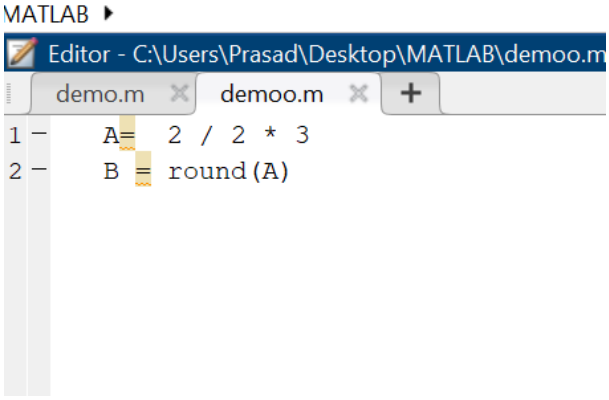
**Explanation:**

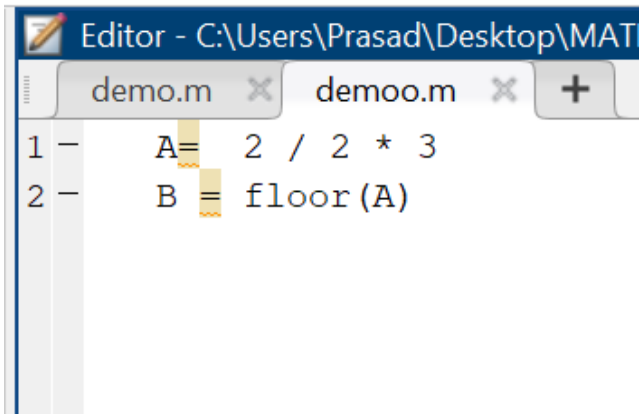
round( ) rounds each element of X to the nearest integer

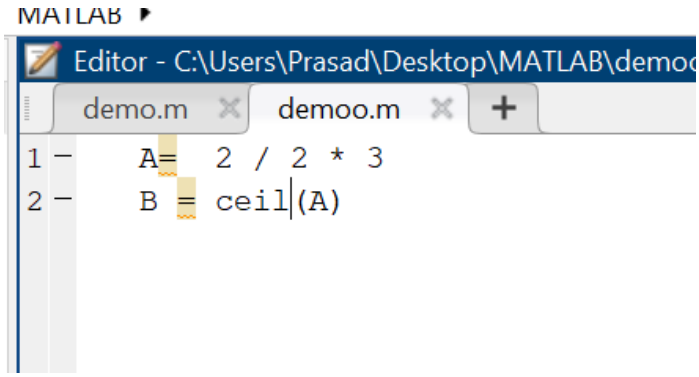
ceil() rounds to the next higher integer, in the +infinity direction.

floor() rounds to the next lower integer, in the -infinity direction.

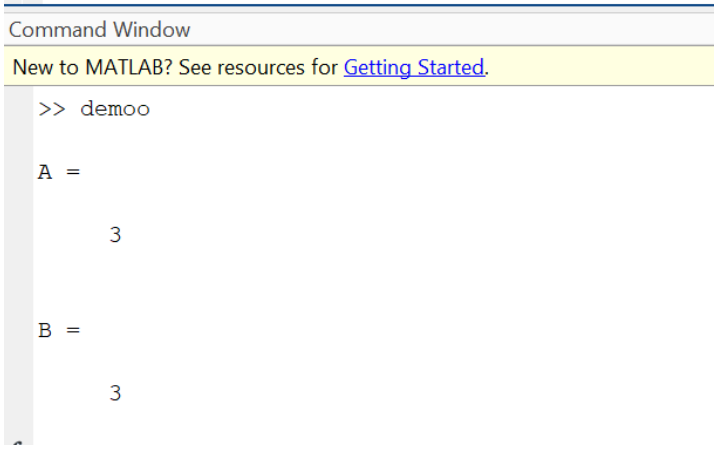
**Code:**

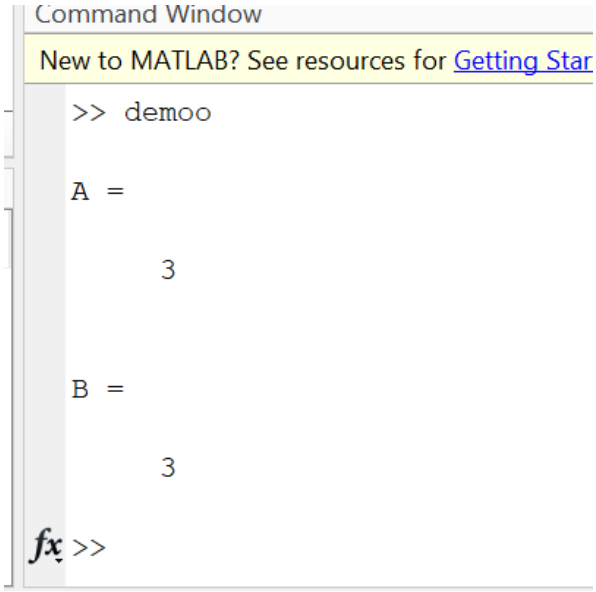


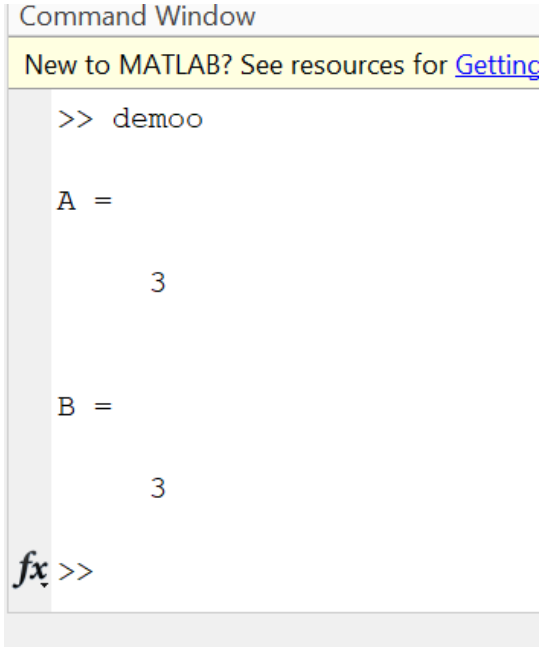



**Output:**

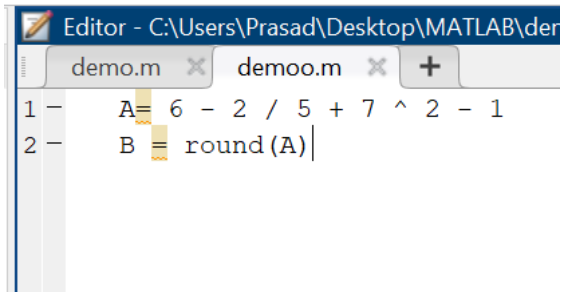

Command Window
New to MATLAB? See resources for Getting Started.
>> demoo
A =
3
B =
3

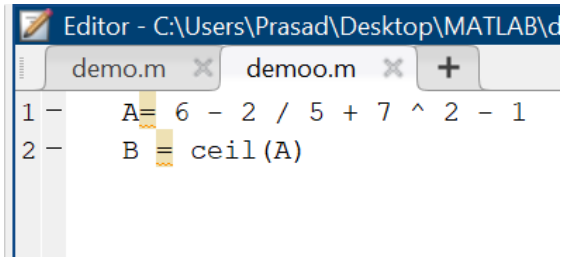

Command Window
New to MATLAB? See resources for Getting Star
>> demoo
A =
3
B =
3
>>

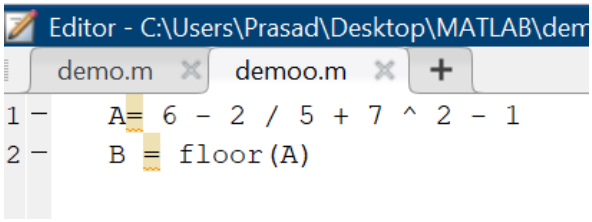

Command Window
New to MATLAB? See resources for Getting
>> demoo
A =
3
B =
3
>>

ii. 6 - 2 / 5 + 7 ^ 2 – 1

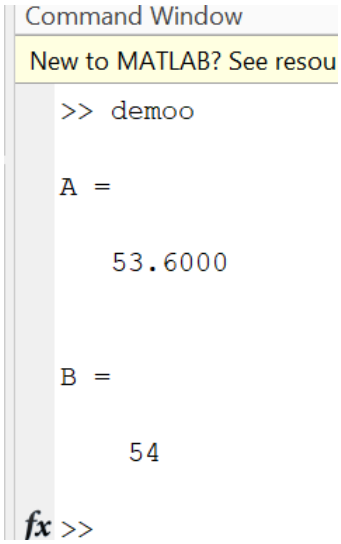
Code:

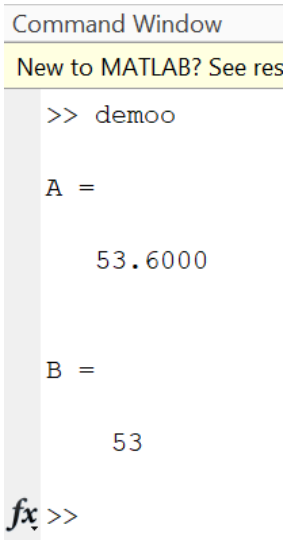


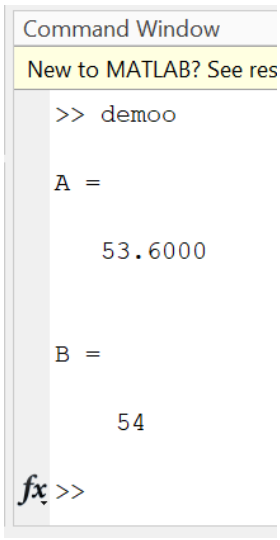




Output:

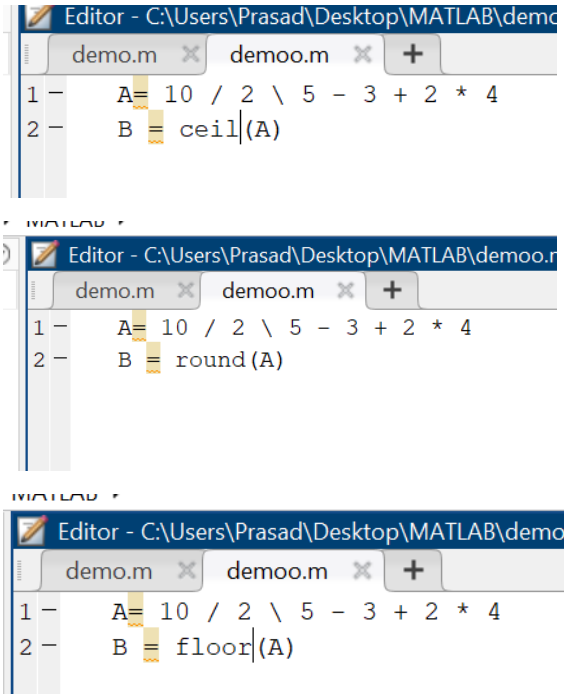



Command Window
New to MATLAB? See res
>> demoo
A =
53.6000
B =
53
>>

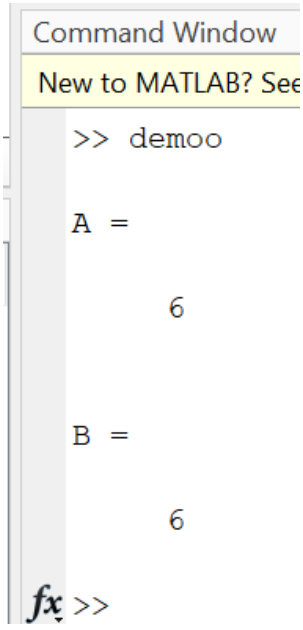

Command Window
New to MATLAB? See res
>> demoo
A =
53.6000
B =
54
>>

iii. 10 / 2 \ 5 - 3 + 2 * 4

Code:




Output:

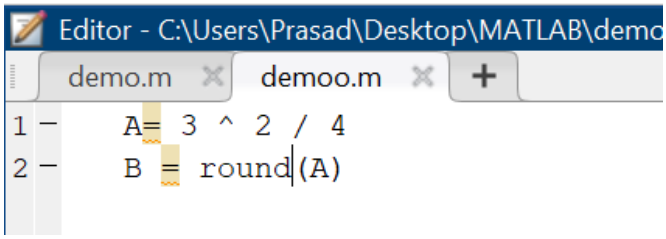


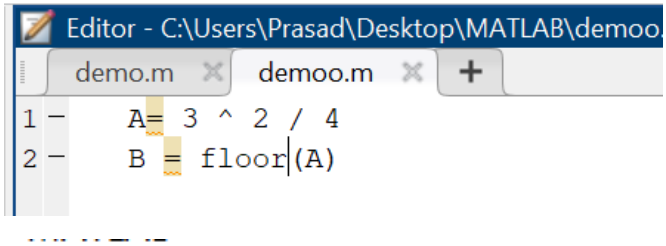
```
>> demoo
A =
     6

B =
     6

>>
```

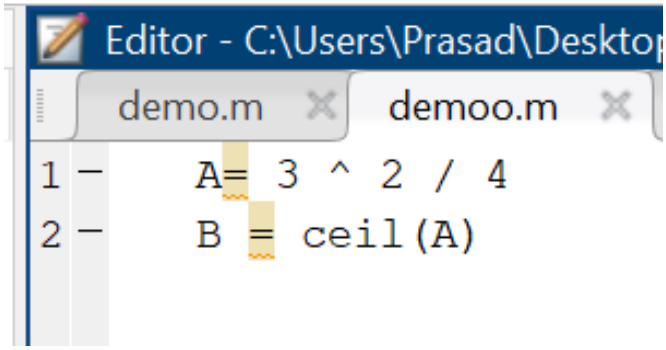
```
>> demoo
A =
     6

B =
     6

>>
```

iv. 3 ^ 2 / 4

Code:



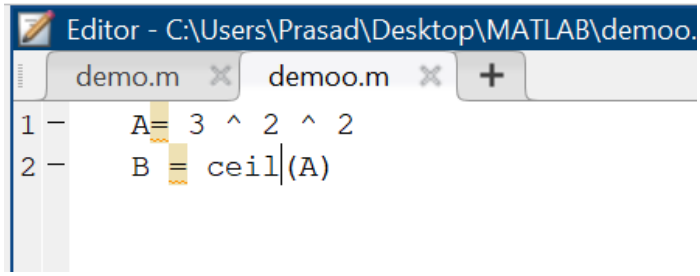

Output:

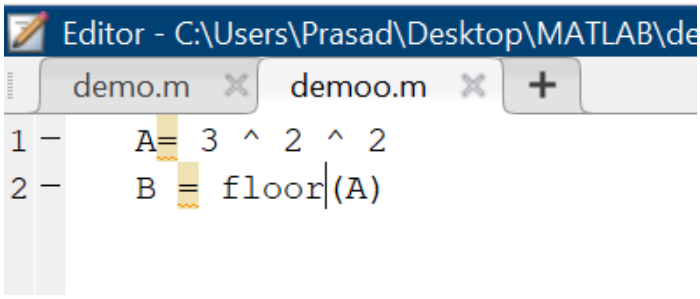
```
A =

    2.2500


B =

     2

fx >>
```

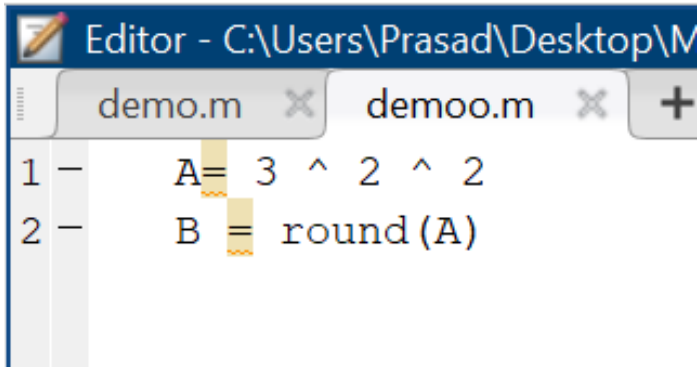
```
>> demoo
A =
    2.2500
B =
     2
fx >>
A =
    2.2500
B =
     3
fx >>
```

v. 3 ^ 2 ^ 2

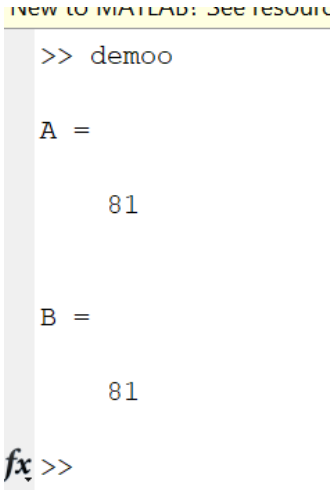
Code:




Output:

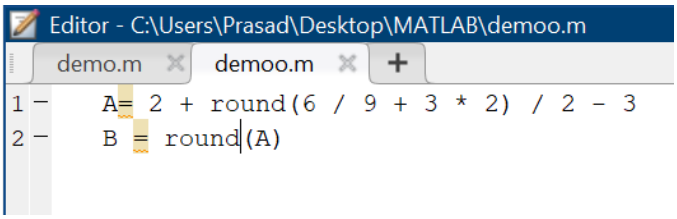


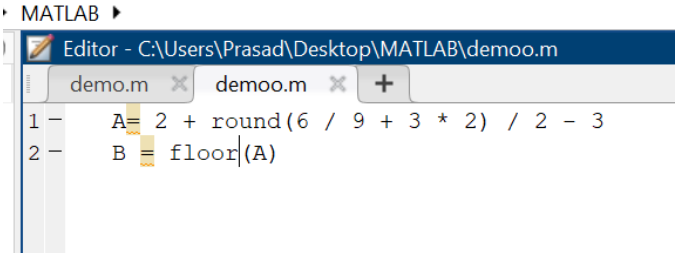
```
>> demoo

A =

    81


B =

    81

fx >>
```

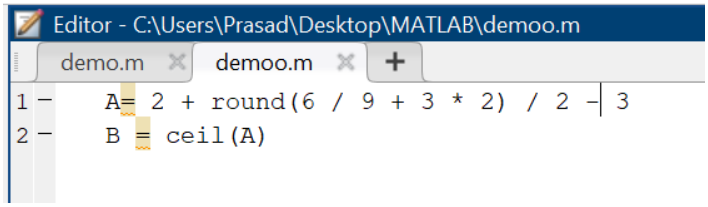
```
>> demoo
A =
    81

B =
    81
fx >>
```

vi. 2 + round(6 / 9 + 3 * 2) / 2 – 3

Code:




Output:

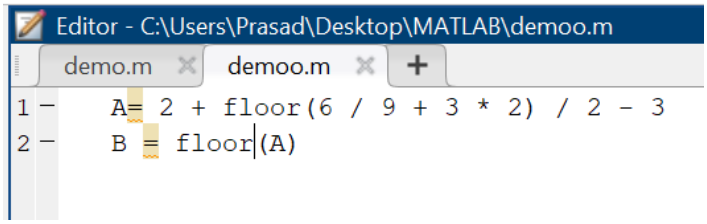
```
>> demoo

A =

    2.5000


B =

     3

fx >>
```

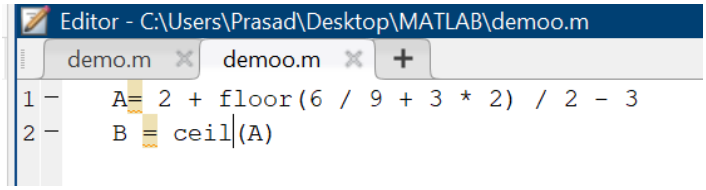
```
>> demoo

A =

    2.5000


B =

     2

fx >>
```

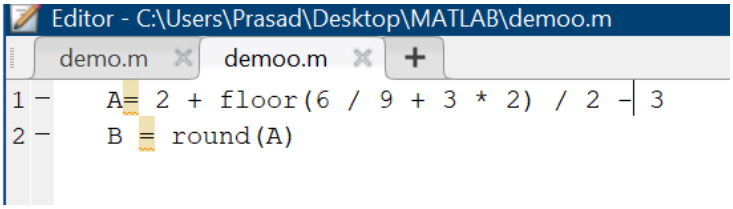
```
>> demoo

A =

    2.5000


B =

     3

fx >>
```

vii. 2 + floor(6 / 9 + 3 * 2) / 2 – 3

Code:




Output:

```
>> demoo

A =

     2


B =

     2

fx >>
```

```
>> demoo

A =

     2


B =

     2

fx >>
```

```
>> demoo
A =
     2

B =
     2

fx >>
```

viii. 2 + ceil(6 / 9 + 3 * 2) / 2 – 3

Code:

demo.m | demoo.m

```
A= 2 + ceil(6 / 9 + 3 * 2) / 2 - 3
B = round(A)
```

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
A= 2 + ceil(6 / 9 + 3 * 2) / 2 - 3
B = ceil(A)
```

Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m

demo.m | demoo.m

```
A= 2 + ceil(6 / 9 + 3 * 2) / 2 - 3
B = floor(A)
```

Output:

```
>> demoo

A =

    2.5000


B =

     3

fx >>
```

```
>> demoo

A =

    2.5000


B =

     3

fx >>
```

```
>> demoo
A =

    2.5000

B =

     2

fx >>
```

7. Write a program to demonstrate the use of functions length (), ndims (), numel (), and size (). How they are different for 1-D and 2-D array? How many functions are having same values w.r.t 1-D and 2-D array.

**Explanation:**

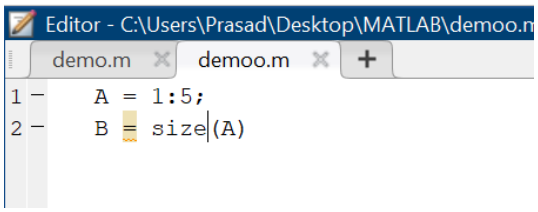
ndims() returns the number of dimensions in the array A. The number of dimensions is always greater than or equal to *2*. The function ignores trailing singleton dimensions, for which size(A,dim) = 1.

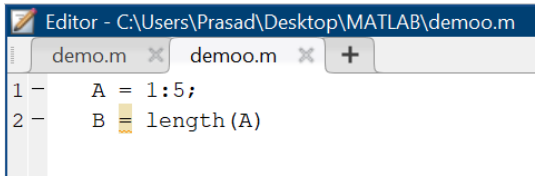
length() returns the length of the largest array dimension in X

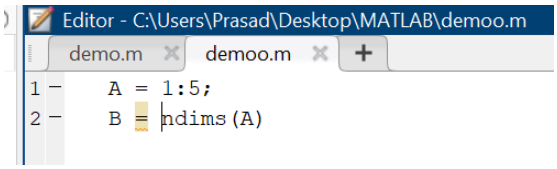
size () returns a row vector whose elements are the lengths of the corresponding dimensions of A

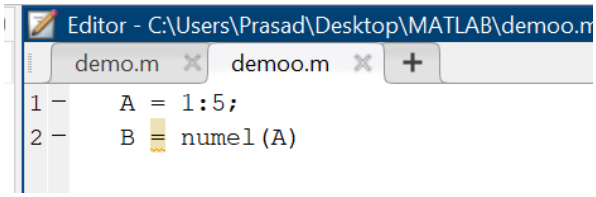
numel() returns the number of elements, n, in array A, equivalent to prod(size())

**Code:**

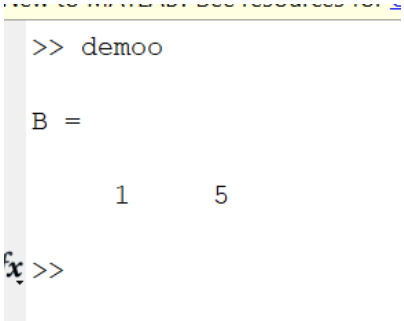



Output:



```
>> demoo

B =

     5

fx >>
```

```
>> demoo

B =

     2

fx >>
```

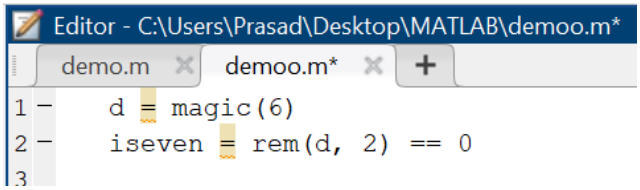
```
>> demoo

B =

     5

fx >>
```

4. Write a program to create a magic array of size 6 and check whether each element is odd or even and display the output accordingly.

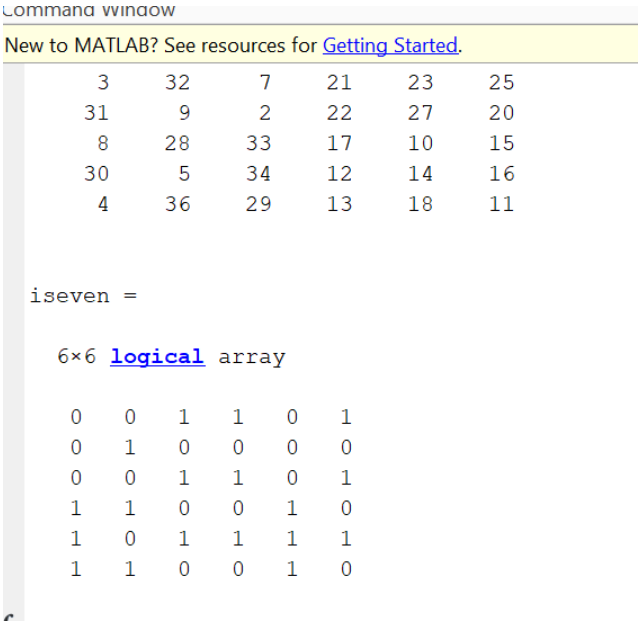
Explanation:

First create a magic array of size 6 and than use iseven function to point out even numbers as 1 and odd as 0.

Code:




Output:



5. What is the difference between transpose (), reshape () and rot90 ()? Explain with program.

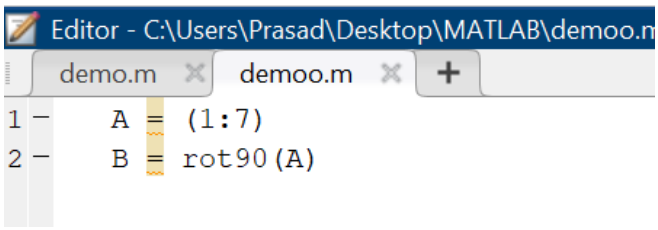
Explanation:

rot90() rotates array A counterclockwise by 90 degrees.

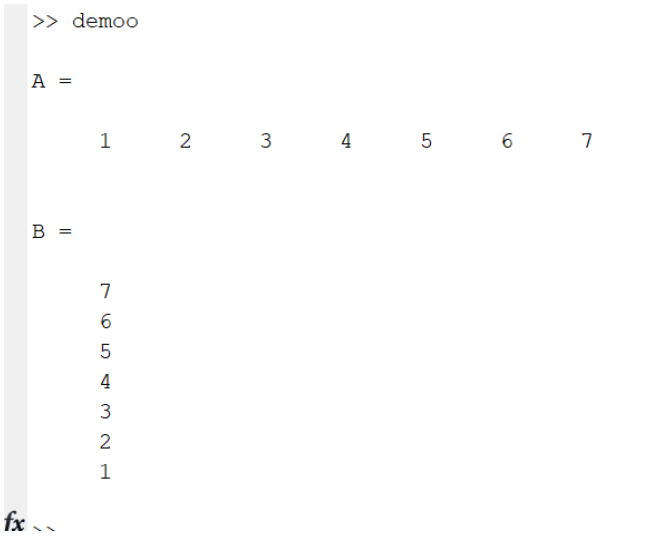
Transpose () = B = A.' returns the nonconjugate transpose of A, that is, interchanges the row and column index for each element. If A contains complex elements, then A.' does not affect the sign of the imaginary parts.

B = reshape(A,sz) reshapes A using the size vector, sz, to define size(B). For example, reshape(A,[2,3]) reshapes A into a 2-by-3 matrix. sz must contain at least 2 elements, and prod(sz) must be the same as numel(A).

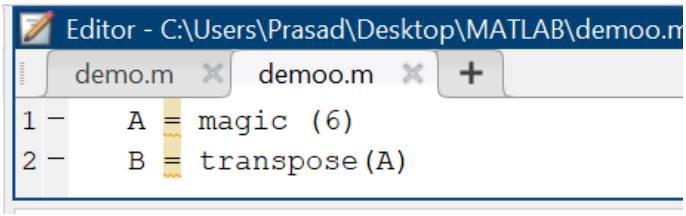
Code:



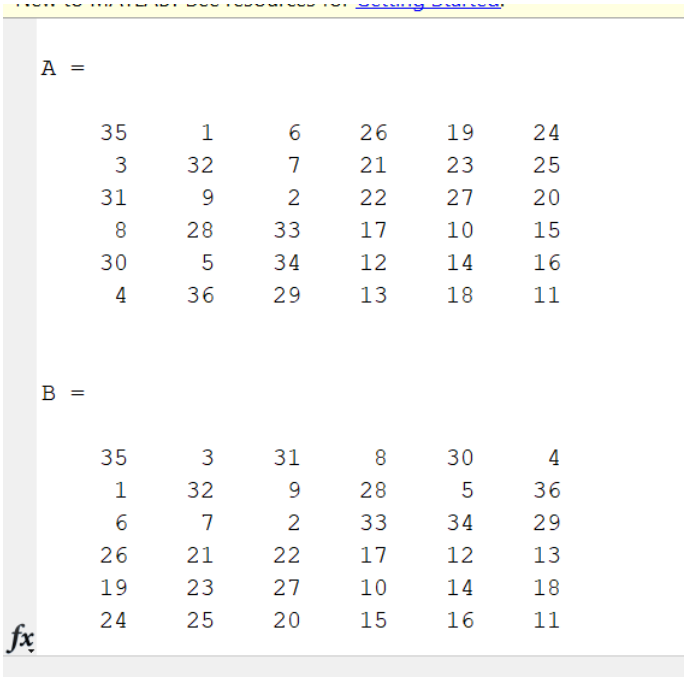

Output:



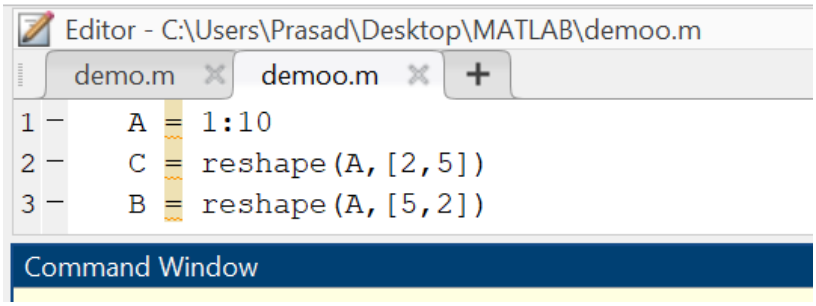
Code:


Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m
demoo.m
1 A = magic (6)
2 B = transpose(A)

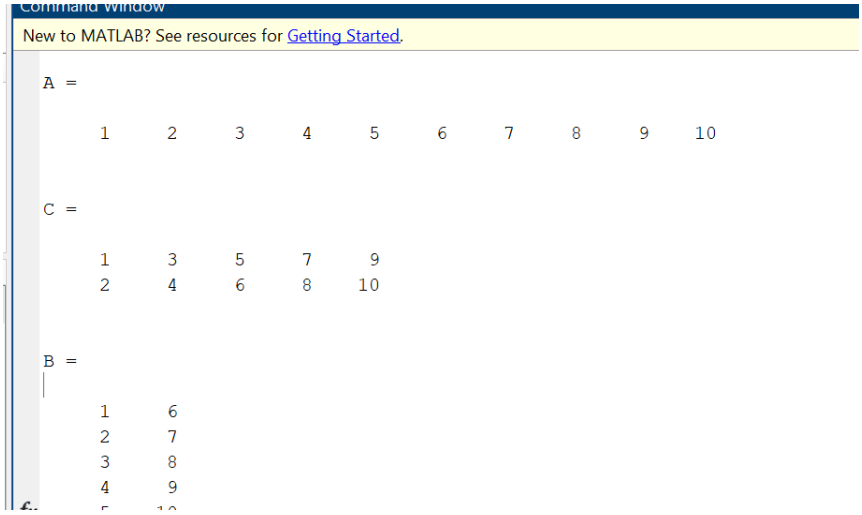
Output:


A =
35 1 6 26 19 24
3 32 7 21 23 25
31 9 2 22 27 20
8 28 33 17 10 15
30 5 34 12 14 16
4 36 29 13 18 11
B =
35 3 31 8 30 4
1 32 9 28 5 36
6 7 2 33 34 29
26 21 22 17 12 13
19 23 27 10 14 18
24 25 20 15 16 11

Code:


Editor - C:\Users\Prasad\Desktop\MATLAB\demoo.m
demo.m
demoo.m
1 A = 1:10
2 C = reshape(A,[2,5])
3 B = reshape(A,[5,2])
Command Window

Output:


Command Window
New to MATLAB? See resources for Getting Started.
A =
1 2 3 4 5 6 7 8 9 10
C =
1 3 5 7 9
2 4 6 8 10
B =
1 6
2 7
3 8
4 9

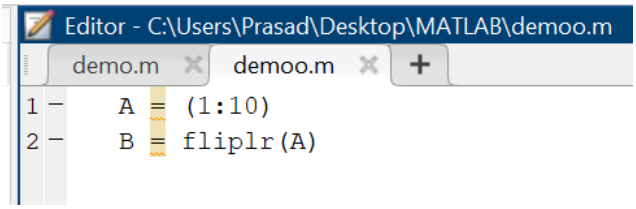
6. Write a program to demonstrate fliplr () and flipud () in both 1-D and 2-D array.

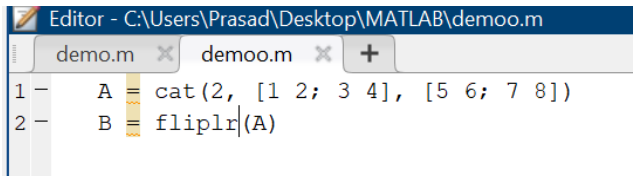
fliplr() returns A with its columns flipped in the left-right direction (that is, about a vertical axis).

flipud () function to flip arrays in the vertical direction

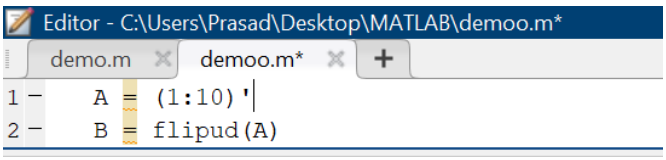
**Code:**

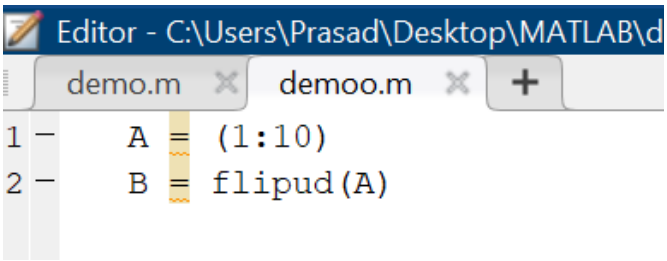
fliplr() :

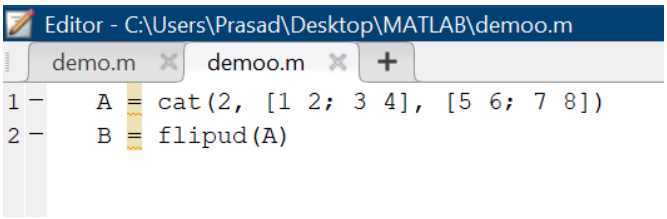



flipud ():



**Output:**

```
>> demoo
A =
     1     2     3     4     5     6     7     8     9    10
B =
    10     9     8     7     6     5     4     3     2     1
fx >>
```

```
      7
      8
      9
     10
B =
     10
      9
      8
      7
      6
      5
      4
      3
      2
      1
```

```
>> demoo
A =
     1     2     3     4     5     6     7     8     9    10
B =
     1     2     3     4     5     6     7     8     9    10
fx >>
```

```
A =
     1     2     5     6
     3     4     7     8
B =
     3     4     7     8
     1     2     5     6
fx >>
```

```
A =

     1     2     5     6
     3     4     7     8


B =

     6     5     2     1
     8     7     4     3

fx >>
```

7. Write a program to demonstrate cumprod (). Display row-wise and column-wise.

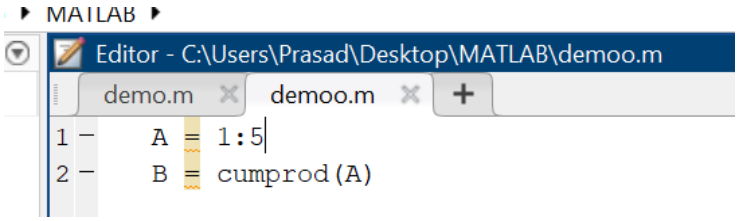
Explanation:

B = cumprod(A) returns the cumulative product of A starting at the beginning of the first array dimension in A whose size does not equal 1.

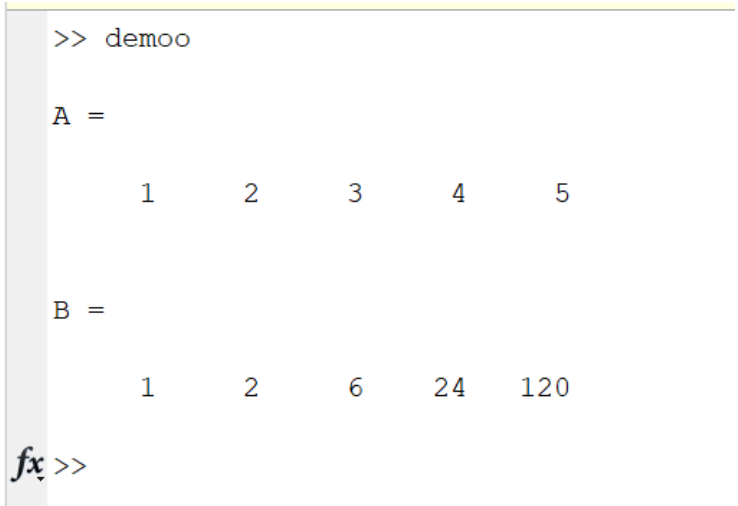
- If A is a vector, then cumprod(A) returns a vector containing the cumulative product of the elements of A.
- If A is a matrix, then cumprod(A) returns a matrix containing the cumulative products for each column of A.
- If A is a multidimensional array, then cumprod(A) acts along the first nonsingleton dimension.

i) 1-D

Code:

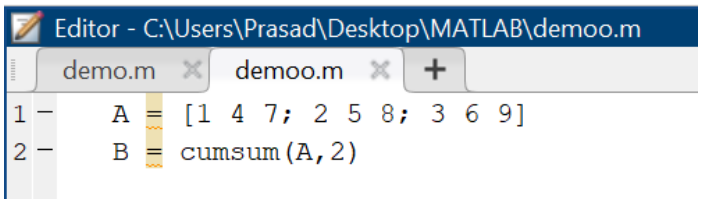



Output:



ii) 2-D

Code:

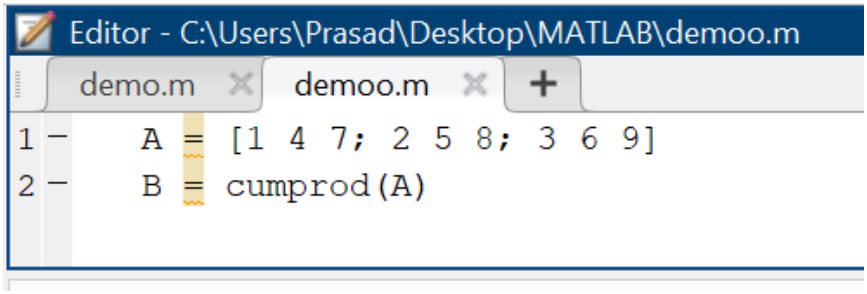



Output:

```
>> demoo

A =

     1     4     7
     2     5     8
     3     6     9


B =

     1     5    12
     2     7    15
     3     9    18

fx >>
```

iii) 3-D

Code:



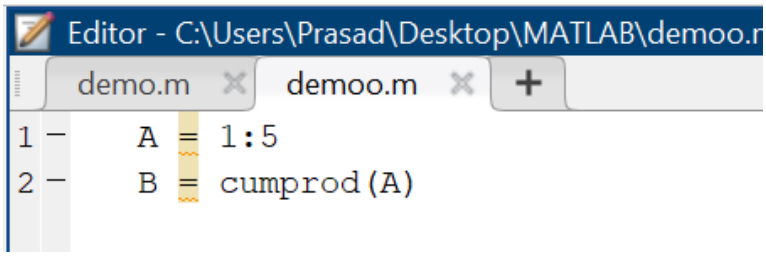

Output:

```
>> demoo

A =

     1     4     7
     2     5     8
     3     6     9


B =

     1     4     7
     2    20    56
     6   120   504

fx >>
```

8. Write a program to demonstrate reverse cumulative product. Display row-wise and column-wise.

    i. 1-D

Code:

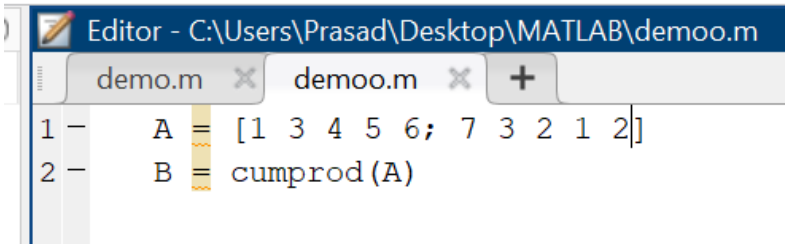



Output:

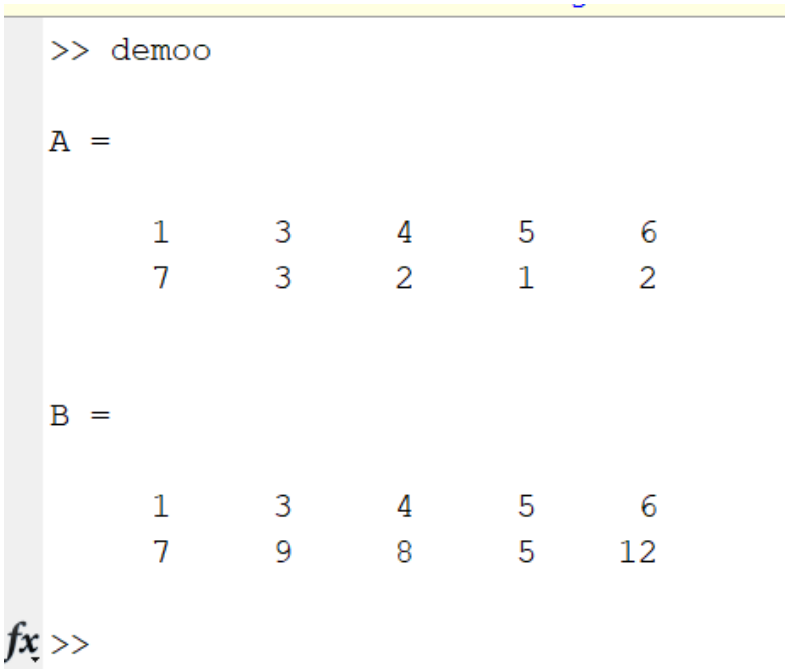
```
>> demoo

A =

     1     2     3     4     5


B =

     1     2     6    24   120

fx >>
```

ii. 2-D

Code:




Output:



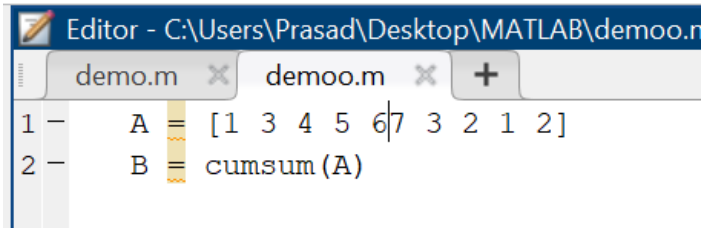
9. Write a program to demonstrate cumsum (). Display row-wise and column-wise.

B = cumsum(A) returns the cumulative sum of A starting at the beginning of the first array dimension in A whose size does not equal 1. The output B has the same size as A.

- If A is a vector, then cumsum(A) returns a vector containing the cumulative sum of the elements of A.
- If A is a matrix, then cumsum(A) returns a matrix containing the cumulative sums of each column of A.
- If A is a multidimensional array, then cumsum(A) acts along the first nonsingleton dimension.

i. 1-D

Code:

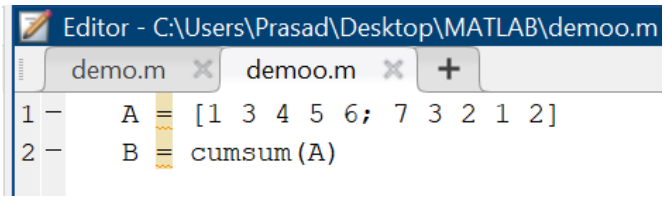



Output:

```
>> demoo

A =

     1     3     4     5    67     3     2     1     2


B =

     1     4     8    13    80    83    85    86    88

fx >>
```

ii. 2-D

Code:

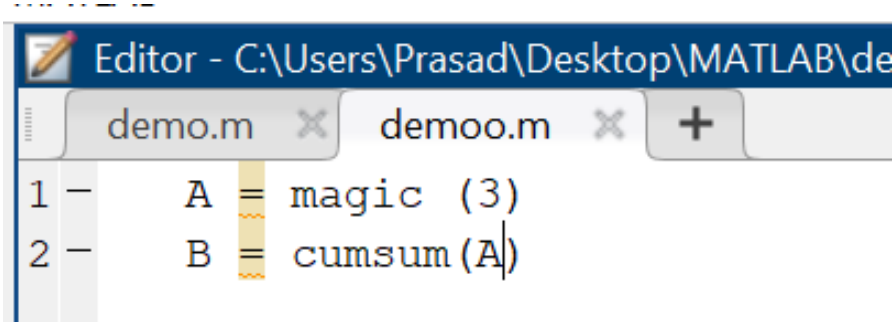



Output:

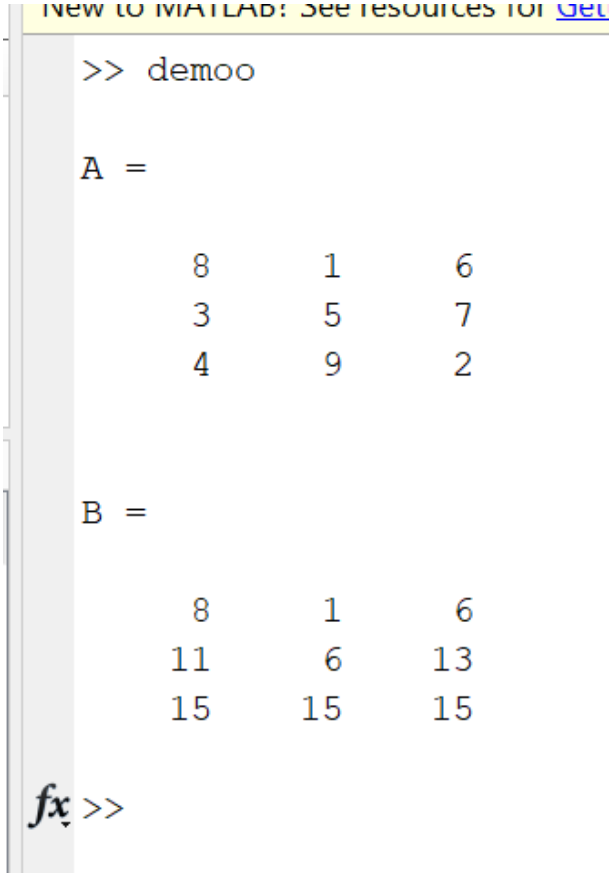
```
>> demoo

A =

     1     3     4     5     6
     7     3     2     1     2


B =

     1     3     4     5     6
     8     6     6     6     8

fx >>
```

iii. 3-D

Code:




Output:



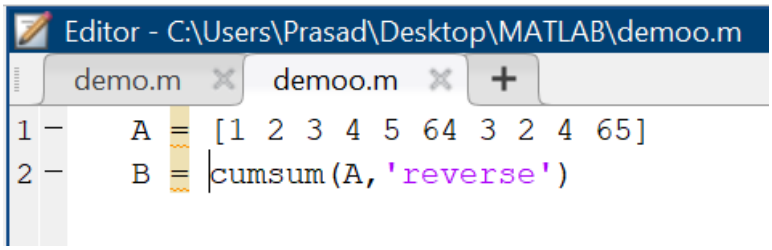
10. Write a program to demonstrate reverse cumulative sum. Display row-wise and column-wise.

Explanation:

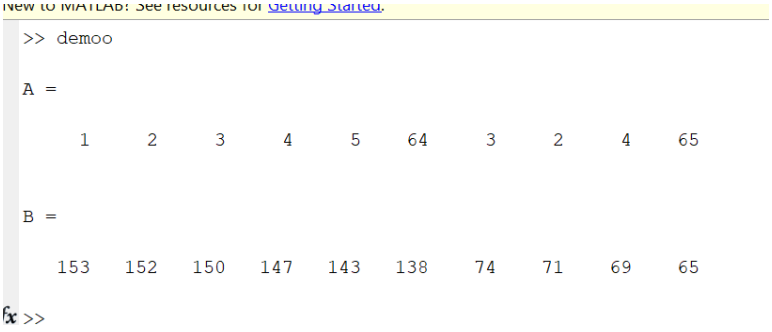
B = cumsum(___,direction) specifies the direction using any of the previous syntaxes. For instance, cumsum(A,2,'reverse') returns the cumulative sum within the rows of A by working from end to beginning of the second dimension.

i. 1-D

Code:

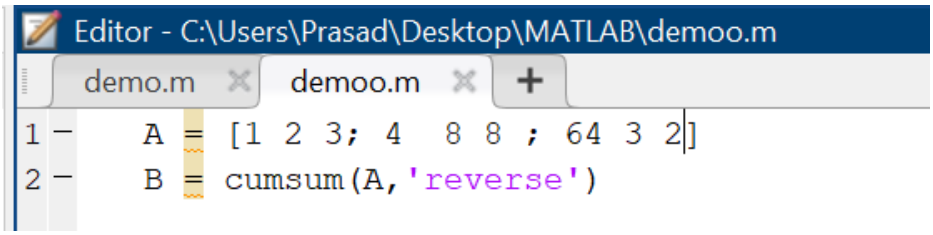



Output:



ii. 2-D

Code:




Output:

```
>> demoo

A =

     1     2     3
     4     8     8
    64     3     2


B =

    69    13    13
    68    11    10
    64     3     2

>>
```

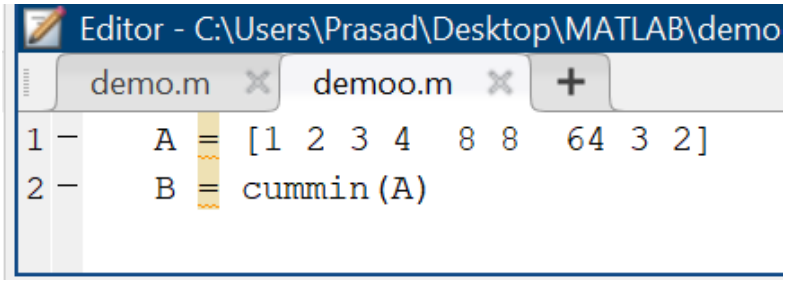
11. Write a program to demonstrate cummin (). Display row-wise and column-wise.

M = cummin(A) returns the cumulative minimum elements of A. By default, cummin(A) operates along the first array dimension whose size does not equal 1.

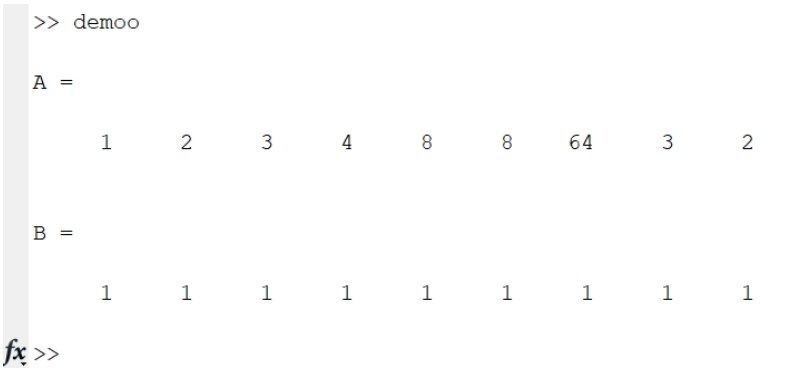
- If A is a vector, then cummin(A) returns a vector of the same size containing the cumulative minima of A.
- If A is a matrix, then cummin(A) returns a matrix of the same size containing the cumulative minima in each column of A.
- If A is a multidimensional array, then cummin(A) returns an array of the same size containing the cumulative minima along the first array dimension of A whose size does not equal 1.

i. 1-D

Code:

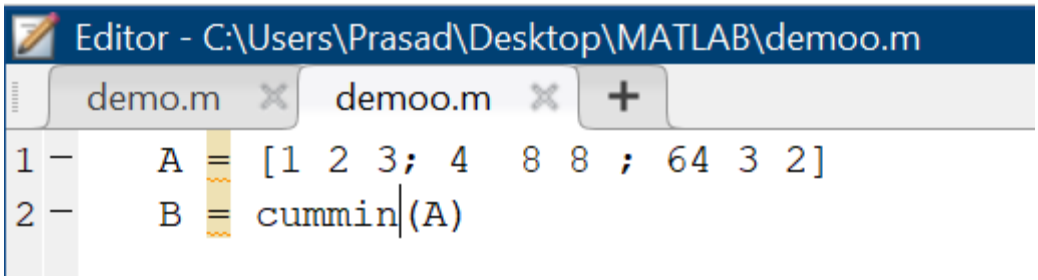



Output:



ii. 2-D

Code:




Output:

```
>> demoo

A =

     1     2     3
     4     8     8
    64     3     2


B =

     1     2     3
     1     2     3
     1     2     2

>>
```

12. Write a program to demonstrate reverse cumulative minimum. Display row-wise and column-wise.

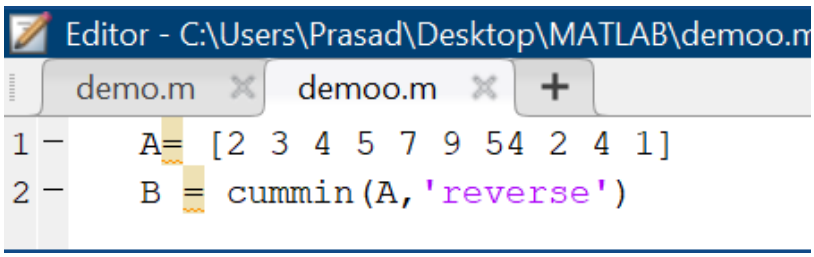
Explanation:

reverse' works from end to 1 of the active dimension.

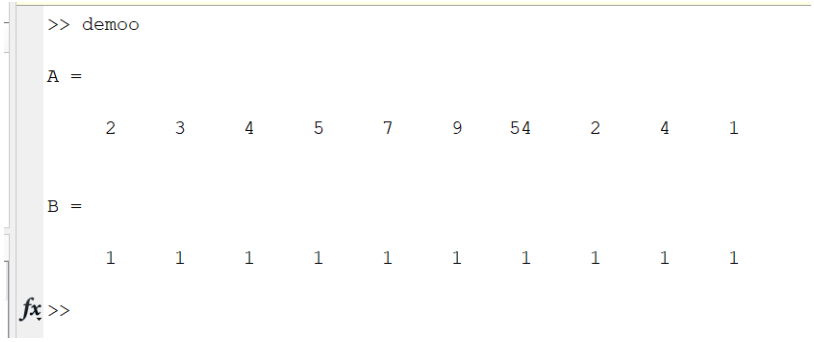
M = cummin(___,direction) optionally specifies the direction using any of the previous syntaxes. You must specify A and, optionally, can specify dim. For instance, cummin(A,2,'reverse') returns the cumulative minima of A by working from end to beginning of the second dimension of A.

i. 1-D

Code:

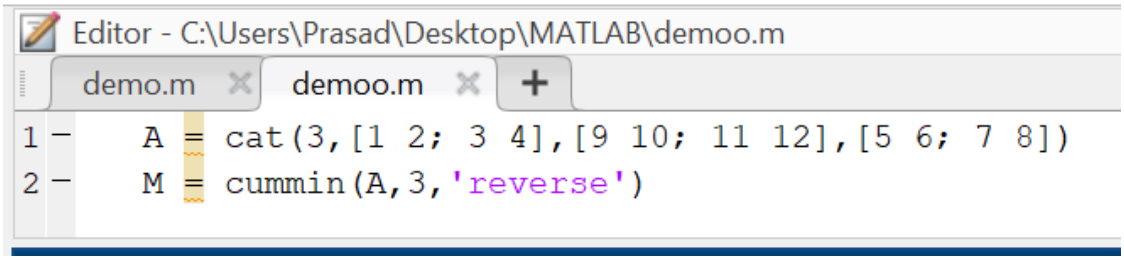



Output:



ii. 2-D

Code:




Output:

```
>> demoo

A(:,:,1) =

     1     2
     3     4


A(:,:,2) =

     9    10
    11    12


A(:,:,3) =

     5     6
     7     8
```

```
M(:,:,1) =

     1     2
     3     4


M(:,:,2) =

     5     6
     7     8


M(:,:,3) =

     5     6
     7     8
```

13. Write a program to demonstrate cummax (). Display row-wise and column-wise.

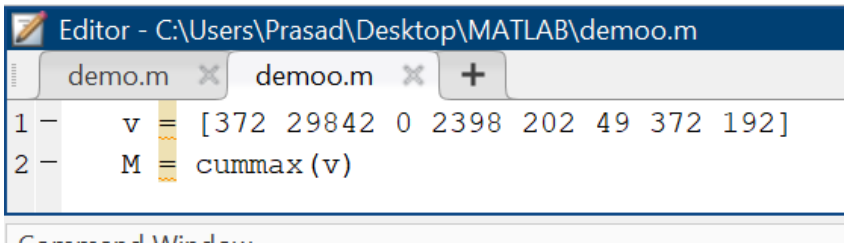
Explanation:

M = cummax(A) returns the cumulative maximum elements of A. By default, cummax(A) operates along the first array dimension whose size does not equal 1.

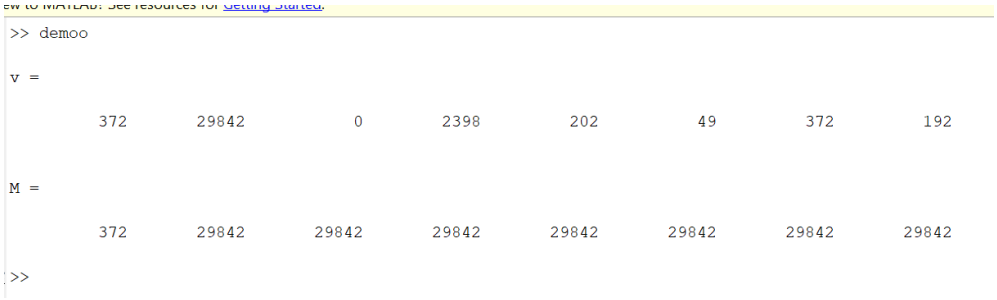
- If A is a vector, then cummax(A) returns a vector of the same size containing the cumulative maxima of A.
- If A is a matrix, then cummax(A) returns a matrix of the same size containing the cumulative maxima in each column of A.
- If A is a multidimensional array, then cummax(A) returns an array of the same size containing the cumulative maxima along the first array dimension of A whose size does not equal 1.

i. 1-D

Code:

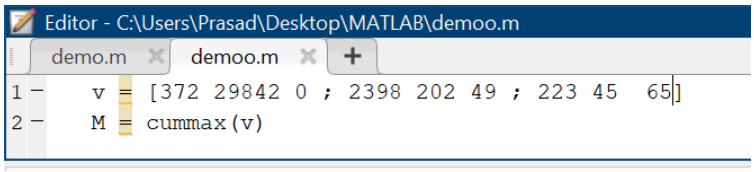



Output:



ii. 2-D

Code:

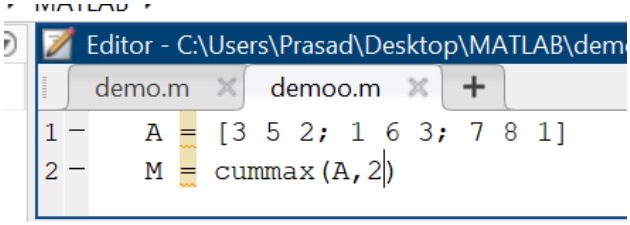



Output:

```
>> demoo

v =

         372       29842           0
        2398         202          49
         223          45          65


M =

         372       29842           0
        2398       29842          49
        2398       29842          65

>>
```

iii. 3-D

Code:




Output:

```
>> demoo

A =

     3     5     2
     1     6     3
     7     8     1


M =

     3     5     5
     1     6     6
     7     8     8

>>
```

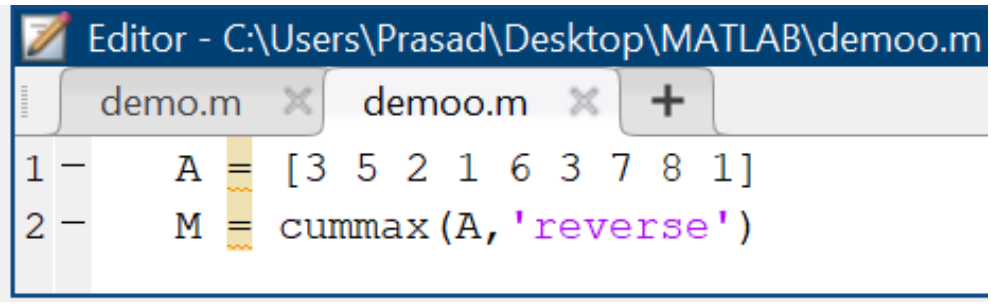
14. Write a program to demonstrate reverse cumulative maximum. Display row-wise and column-wise.

Explanation:

M = cummax(___,direction) optionally specifies the direction using any of the previous syntaxes. You must specify A and, optionally, can specify dim. For instance, cummax(A,2,'reverse') returns the cumulative maxima of A by working from end to beginning of the second dimension of A.

i. 1-D

Code:

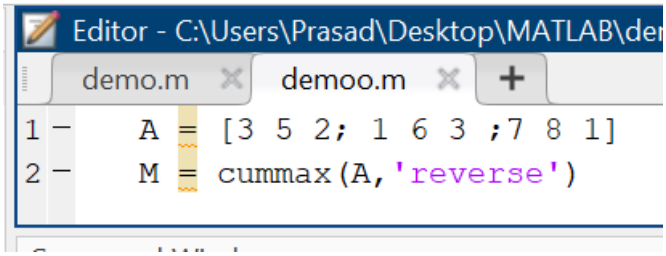



Output:

```
>> demoo

A =

     3     5     2     1     6     3     7     8     1


M =

     8     8     8     8     8     8     8     8     1

fx >>
```

ii. 2-D

Code:




Output:

```
>> demoo

A =

     3     5     2
     1     6     3
     7     8     1


M =

     7     8     3
     7     8     3
     7     8     1

fx >>
```

15. What is the difference between sort () and sortrows ()? Explain with a program.

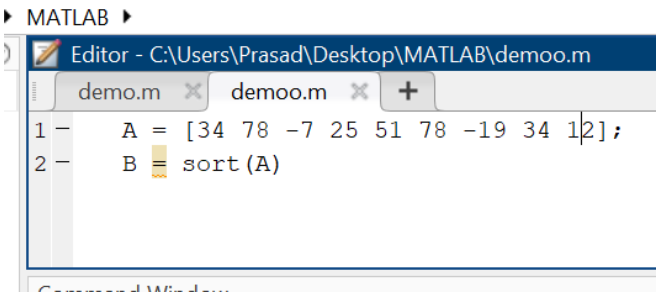
**Explanation:**

The sortrows() function provides additional flexibility for subsorting over multiple columns of matrix or table inputs.

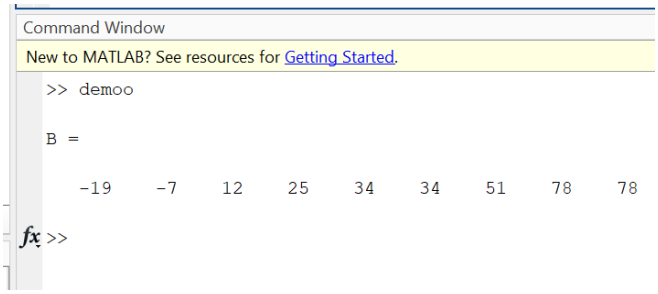
The sort () function and the relational operators use different orderings for complex numbers.

**Sort()**

**Code:**

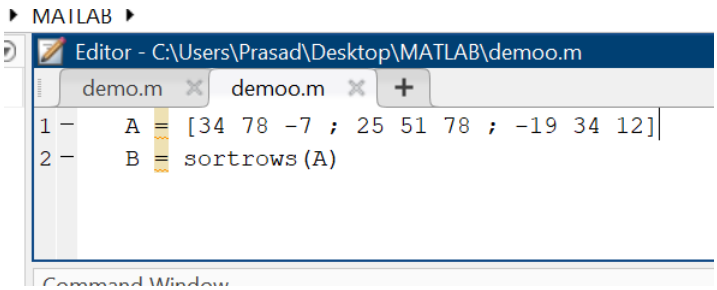



**Output:**

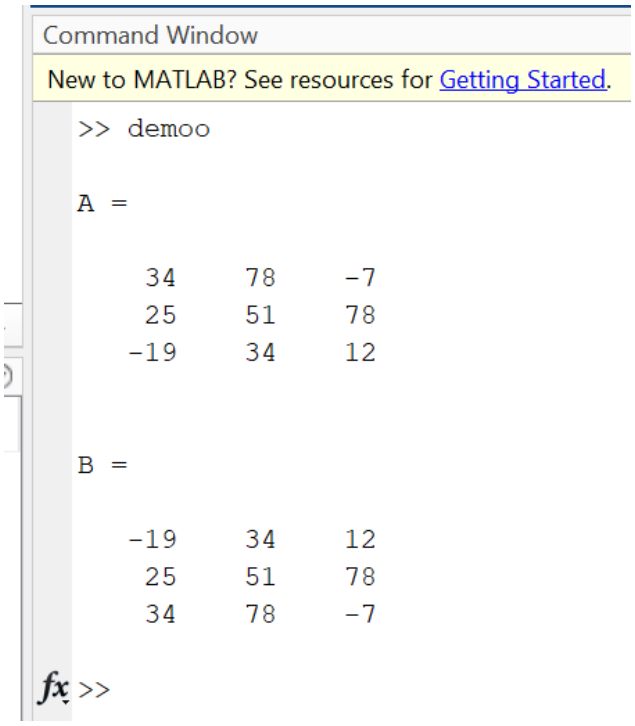


**Sortrows()**

**Code:**




**Output:**

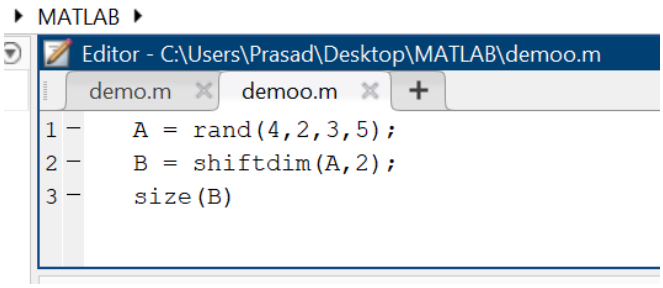


16. Write a program to demonstrate the shiftdim () function.

**Explanation:**

So I have created a 4-by-2-by-3-by-5 array and I am just shifting the dimensions 2 positions to the left, wrapping the first 2 lengths to the last 2 dimensions.

**Code:**




**Output:**