

## Report

Date:	7July 2020	Name:	Safiya Banu
Course:	MATLAB Onramp	USN:	4AL16EC061
Topic:	<ol style="list-style-type: none"><li>1. Indexing into arrays<ol style="list-style-type: none"><li>i. Indexing into arrays</li><li>ii. Extraxing multiple elements</li></ol></li><li>2. Array calculation<ol style="list-style-type: none"><li>I. Performingarray operations on vectors</li></ol></li></ol>	Semester & Section:	8 <sup>th</sup> sem “B”section
Github Repository:	Safiya-Courses		

← MY COURSES


MATLAB Onramp (35% complete)

SAFIYA BANU

5.1 Indexing into Arrays: (1/2) Indexing into Arrays

← PREVIOUS

NEXT →

 **x**

2

3

1

-9

0

5

-3

Indexing

2	3	6	-9	0
5	-3	7	-1	-8

0:14 / 1:32

← PREVIOUS

NEXT →

MY COURSES
MATLAB Onramp (38% complete)
SAFIYA BANU

5.1 Indexing into Arrays: (2/2) Practice
PREVIOUS
NEXT

Task 1
Task 2
Task 3

Note that you can use arithmetic with the keyword `end`. For example:

```
y = A(end-1,end-2)
```

**TASK**  
 Create a scalar variable `x` that contains the value in the second to last (`end-1`) row and 3<sup>rd</sup> column of `data`.

[Hint](#) | [See Solution](#) | [Reset](#)

**Test Results: Correct!**  
 ✓ Is `x` assigned correctly?

**Further Practice**

HOME
LIVE EDITOR
VIEW

indexingSoln1.mlx x
indexing.mlx \* x

1
2
3
4
5
6
7
8

```

1 load data
2
3
4 x=data(6,3)
5
6 x=data(end,3)
7
8 x = data(end-1,3)

```

Task 1
Task 2
Task 3

```

x=data(6,3)

x=data(end,3)

x = data(end-1,3)

```

data = 7x4
3.0000 0.5300 ...
18.0000 1.7800
19.0000 0.8600
20.0000 1.6000
21.0000 3.0000
23.0000 6.1100
38.0000 2.5400

x = 9.0698
x = 5.3002
x = 9.0698

## Extraxing multiple elements

When used as an index, the colon operator (`:`) specifies all the elements in that dimension. The syntax

```
x = A(2,:)
```

creates a row vector containing all of the elements from the second row of `A`.

### TASK 1

Create a variable named `density` that contains the second column of the matrix named `data`.

```
load datafile
data
density=data(:,2)
```

## Changing values in arrays

### TASK 1

Create a vector named `v2` containing the last column of `data`.

```
v2 = data(:,end)
```

## TASK 2

Change the first element in `v2` from `NaN` to `0.5`.

```
v2(1) = 0.5
```

# ARRAY CALCULATIONS

## Performing array operations on vectors

MATLAB is designed to work naturally with arrays. For example, you can add a scalar value to all the elements of an array.

```
y = x + 2
```

The screenshot shows the MATLAB Onramp interface for a course titled "6.1 Performing Array Operations on Vectors". The user is logged in as "SAFIYA BANU". The interface is divided into several sections:

- Task 1:** A text box explains that MATLAB is designed to work naturally with arrays and provides the example `y = x + 2`. Below this, a task instruction says: "Add 1 to each element of `v1` and store the result in a variable named `r`." There are buttons for "Hint", "See Solution", "Reset", "Submit", and "Next task".
- Test Results:** Shows "Correct!" with two green checkmarks: "Does the variable `r` exist?" and "Is `r` assigned correctly?".
- Task List:** A sidebar on the left lists tasks 1 through 5, with Task 1 currently selected.
- Code Editor:** The main area shows a code editor with the following code:

```
1 load datafile
2 density = data(:,2);
3 v1 = data(:,3);
4 v2 = data(:,4);

5 r = 1 + v1
```
- Workspace:** On the right, the workspace shows a variable `r` of size `7x1` with the following values:

```
5.0753
7.6678
2.5177
4.6375
5.7243
10.0698
6.3002
```
- Command Window:** At the bottom, the command window is visible but empty.

The `*` operator performs matrix multiplication. So, if you use `*` to multiply two equally sized vectors, since the inner dimensions do not agree, you will get an error message.

```
z = [3 4] * [10 20]
```

Error using `*`

Incorrect dimensions for matrix multiplication.

In contrast, the `.*` operator performs element wise multiplication and allows you to multiply the corresponding elements of two equally sized arrays.

```
z = [3 4] .* [10 20]
z = 30 80
```

--