

DAILY ASSESSMENT FORMAT

Date:	6 th July 2020	Name:	Sushmitha R Naik
Course:	Mat lab Onramp and Cisco IoT	USN:	4AL17EC090
Topic:	1. Course Overview 2. Commands 3. MATLAB Desktop and Editor 4. Vectors and Matrices 5.CISCO IOT	Semester & Section:	6 th B
GitHub Repository:	Sushmitha_naik		

FORENOON SESSION DETAILS

The screenshot displays the MATLAB Onramp course interface. The browser address bar shows the URL: matlabacademy.mathworks.com/R2020a/portal.html?course=gettingstarted#chapter=2&lesson=1§ion=1. The course title is "MATLAB Onramp" with a progress indicator of "5% complete". The user's name, "Sushmitha R Naik", is visible in the top right corner.

The left sidebar lists the tasks for the current section, "2.1 Entering Commands":

- Task 1
- Task 2
- Task 3
- Task 4
- Task 5
- Task 6
- Task 7
- Further Practice

The main command window shows the following commands and their outputs:

```

>> k=8-2;
Task 5 ✓
>> m=3*k
Task 6 ✓
m =
    18

Task 7 ✓
>> y
y =
     8

>> y=m/2
y =
     9
  
```

The workspace panel on the right shows the following variables:

Name	Size
ans	1x1
k	1x1
m	1x1
y	1x1

A blue callout box provides instructions on how to recalculate variables in MATLAB:

The value of `y` was unchanged because MATLAB does not rerun previous commands in the Command Window.

If you want to recalculate `y` after `m` is modified, you need to repeat the command `y = m/2`.

Try this now! Use the Up arrow to recall the command `y = m/2`, then press Enter. To see the new value of `y`, remember not to use a semicolon at the end of the command.

The bottom of the screenshot shows the Windows taskbar with the date and time: 06 July 2020, Monday, 15:36, 06-07-2020.

Verify Email Address | Solve Algorithms | Dashboard | Home | SAP Scholar | C Tutorial | Sol | 19_ds_oop | MATLAB Onramp | +

matlabacademy.mathworks.com/R2020a/portal.html?course=gettingstarted#chapter=3&lesson=1§ion=1

MY COURSES MATLAB Onramp (17% complete) Sushmitha R Naik

3.1 MATLAB Desktop and Editor

← PREVIOUS NEXT →

MATLAB

HOME PLOTS APPS LIVE EDITOR ASSESS VIEW

File Edit View Tools Window Help

Current Folder: work

Workspace

Command Window

```

>> m = 3;
v = 1.8;
KE = 1/2*m*v^2;

Calculate potential energy

g = 9.8;
h = 50;
PE = m*g*h;

Calculate the total mechanical energy
  
```

0:52 / 2:27

← PREVIOUS NEXT →

Type here to search

Verify Email Address | Solve Algorithms | Dashboard | Home | SAP Scholar | C Tutorial | Sol | 19_ds_oop | MATLAB Onramp | +

matlabacademy.mathworks.com/R2020a/portal.html?course=gettingstarted#chapter=4&lesson=3§ion=1

MY COURSES MATLAB Onramp (29% complete) Sushmitha R Naik

4.3 Array Creation Functions

← PREVIOUS NEXT →

numbers to create nonsquare matrices.

```

x = rand(2)
x =
    0.8147    0.1270
    0.9058    0.9134

x = rand(2,3)
x =
    0.6324    0.2785    0.9575
    0.0975    0.5469    0.9649
  
```

TASK
Use `rand` to create an array that contains 5 rows and 1 column. Assign the result to a variable named `x`.

[Hint](#) | [See Solution](#) | [Reset](#) | [Submit](#) | [Next task](#)

Test Results: Correct!

- ✓ Does `x` have the correct size?
- ✓ Does `x` have the correct values?

Task 3

HOME LIVE EDITOR VIEW

Text Code Control Refactor Task Run Section Run and Advance Run Step Stop

createarrays.mlx

Array Creation Functions

Instructions are in the task pane to the left. Complete and submit each task one at a time.

Task 1

```
1 x=rand(5)
```

Task 2

```
3 x=rand(5,1)
```

Task 3

Workspace

```

x = 5x5
    0.8147    0.0975    0.1576 ...
    0.9058    0.2785    0.9706
    0.1270    0.5469    0.9572
    0.9134    0.9575    0.4854
    0.6324    0.9649    0.8003

x = 5x1
    0.7577
    0.7431
    0.3922
    0.6555
    0.1712
  
```

COMMAND WINDOW

Type here to search

Task 1
Background
A single number, called a *scalar*, is actually an array meaning it contains 1 row and 1 column.

TASK
Create a variable named `x` with a value of 2.

Hint | See Solution | Reset

Task 2
Task 3
Task 4
Task 5
Task 6
Task 7
Further Practice

What's an Array?

All MATLAB variables are *arrays*. This means that each variable can contain multiple elements. You can use arrays to store related data in one variable.

Because you'll use arrays every time you program, it's important to get to know them and the terminology used to describe them.

The diagram illustrates four types of MATLAB arrays within a light blue oval labeled 'ARRAY':

- matrix**: A 2x2 grid containing the values 2, 3, 6, and -9.
- column vector**: A vertical column containing the values 2, 3, 6, and -9.
- row vector**: A horizontal row containing the values 2, 3, 6, and -9.
- scalar**: A single box containing the value 2.

Verify Email Address | Solve Algorithms | Dashboard | Home | SAP Scholar@S | C Tutorial | Solve | 19_ds_oop | MATLAB Onramp | +

matlabacademy.mathworks.com/R2020a/portal.html?course=gettingstarted

← MY COURSES **MATLAB Onramp** (35% complete) Sushmitha R Naik

MATLAB Onramp

- 2. Commands
Enter commands in MATLAB to perform calculations and create variables.
 - ✓ Entering Commands
 - ✓ Naming Variables
 - ✓ Saving and Loading Variables
 - ✓ Using Built-in Functions and Constants
- 3. MATLAB Desktop and Editor
Write and save your own MATLAB programs.
 - ✓ MATLAB Desktop and Editor
 - ✓ The MATLAB Editor
 - ✓ Running Scripts
- 4. Vectors and Matrices
Create MATLAB variables that contain multiple elements.
 - ✓ Manually Entering Arrays
 - ✓ Creating Evenly-Spaced Vectors
 - ✓ Array Creation Functions
- 5. Indexing into and Modifying Arrays
Use indexing to extract and modify rows, columns, and elements of MATLAB arrays.
 - ✓ Indexing into Arrays

Type here to search

16:37 06-07-2020

MAT Lab :

MATLAB is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

As of 2020, MATLAB has more than 4 million users worldwide. MATLAB users come from various backgrounds of engineering, science, and economics.

History

Cleve Moler, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s. He designed it to give his student's access to LINPACK and EISPACK without them having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community. Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded Math Works in 1984 to continue its development. These rewritten libraries were known as JACKPAC. In 2000, MATLAB was rewritten to use a newer set of libraries for matrix manipulation, LAPACK.

MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved in image processing.

Syntax

The MATLAB application is built around the MATLAB programming language. Common usage of the MATLAB application involves using the "Command Window" as an interactive mathematical shell or executing text files containing MATLAB code.

Variables

Variables are defined using the assignment operator, `=`. MATLAB is a weakly typed programming language because types are implicitly converted. It is an inferred typed language because variables can be assigned without declaring their type, except if they are to be treated as symbolic objects, and that their type can change. Values can come from constants, from computation involving values of other variables, or from the output of a function. For example:

```
>> x = 17
```

```
x =  
17
```

```
>> x = 'hat'
```

```
x =  
hat
```

```
>> x = [3*4, pi/2]
```

```
x =  
12.0000 1.5708
```

```
>> y = 3*sin(x)
```

```
y =
```

-1.6097 3.0000

Vectors and matrices

A simple array is defined using the colon syntax: *initial*:*increment*:*terminator*. For instance:

```
>> array = 1:2:9  
array =  
1 3 5 7 9
```

Defines a variable named `array` (or assigns a new value to an existing variable with the name `array`) which is an array consisting of the values 1, 3, 5, 7, and 9. That is, the array starts at 1 (the *initial* value), increments with each step from the previous value by 2 (the *increment* value), and stops once it reaches (or to avoid exceeding) 9 (the *terminator* value).

```
>> array = 1:3:9  
array =  
1 4 7
```

the *increment* value can actually be left out of this syntax (along with one of the colons), to use a default value of 1.

```
>> ari = 1:5  
ari =  
1 2 3 4 5
```

assigns to the variable named `ari` an array with the values 1, 2, 3, 4, and 5, since the default value of 1 is used as the increment.

Indexing is one-based, which is the usual convention for matrices in mathematics, unlike zero-based indexing commonly used in other programming languages such as C, C++, and Java.

Matrices can be defined by separating the elements of a row with blank space or comma and using a semicolon to terminate each row. The list of elements should be surrounded by square brackets `[]`. Parentheses `()` are used to access elements and subarrays (they are also used to denote a function argument list).

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]  
A =  
16 3 2 13  
5 10 11 8  
9 6 7 12  
4 15 14 1
```

```
>> A(2,3)
ans =
11
```

Sets of indices can be specified by expressions such as `2:4`, which evaluates to `[2, 3, 4]`. For example, a submatrix taken from rows 2 through 4 and columns 3 through 4 can be written as:

```
>> A(2:4,3:4)
ans =
11 8
7 12
14 1
```

A square identity matrix of size n can be generated using the function `eye`, and matrices of any size with zeros or ones can be generated with the functions `zeros` and `ones`, respectively.

```
>> eye(3,3)
ans =
1 0 0
0 1 0
0 0 1

>> zeros(2,3)
ans =
0 0 0
0 0 0

>> ones(2,3)
ans =
1 1 1
1 1 1
```

Transposing a vector or a matrix is done either by the function `transpose` or by adding dot-prime after the matrix (without the dot, prime will perform conjugate transpose for complex arrays):

```
>> A = [1 ; 2], B = A.', C = transpose(A)
A =
1
2
B =
1 2
C =
```

1 2

```
>> D = [0 3 ; 1 5], D.'
```

```
D =
```

```
0 3
```

```
1 5
```

```
ans =
```

```
0 1
```

```
3 5
```

Most functions accept arrays as input and operate element-wise on each element. For example, `mod(2*J,n)` will multiply every element in *J* by 2, and then reduce each element modulo *n*. MATLAB does include standard `for` and `while` loops, but (as in other similar applications such as R), using the vectorized notation is encouraged and is often faster to execute. The following code, excerpted from the function *magic.m*, creates a magic square *M* for odd values of *n* (MATLAB function `meshgrid` is used here to generate square matrices *I* and *J* containing `1:n`).

```
[J,I] = meshgrid(1:n);
```

```
A = mod(I + J - (n + 3) / 2, n);
```

```
B = mod(I + 2 * J - 2, n);
```

```
M = n * A + B + 1;
```

Structures

MATLAB supports structure data types. Since all variables in MATLAB are arrays, a more adequate name is "structure array", where each element of the array has the same field names. In addition, MATLAB supports dynamic field names (field look-ups by name, field manipulations, etc.).

Functions

When creating a MATLAB function, the name of the file should match the name of the first function in the file. Valid function names begin with an alphabetic character, and can contain letters, numbers, or underscores. Variables and functions are case sensitive.

Function handles

MATLAB supports elements of lambda calculus by introducing function handles, or function references, which are implemented either in `.m` files or anonymous/nested functions.

Classes and object-oriented programming

MATLAB supports object-oriented programming including classes, inheritance, virtual dispatch, packages, pass-by-value semantics, and pass-by-reference semantics. However, the syntax and calling conventions are significantly different from other languages. MATLAB has value classes and reference classes, depending on whether the class has *handle* as a super-class (for reference classes) or not (for value classes).

Method call behavior is different between value and reference classes. For example, a call to a method

```
object.method();
```

can alter any member of *object* only if *object* is an instance of a reference class, otherwise value class methods must return a new instance if it needs to modify the object.

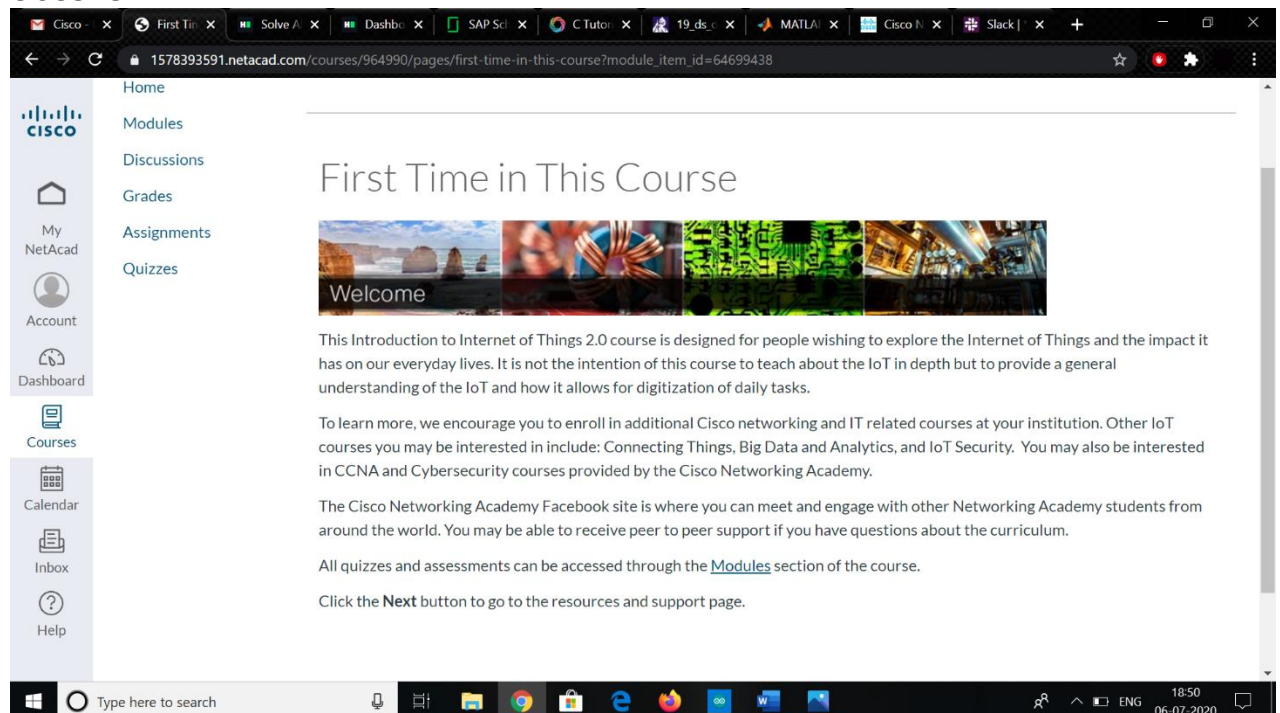
An example of a simple class is provided below.

```
classdef Hello
    methods
        function greet(obj)
            disp('Hello!')
        end
    end
end
```

When put into a file named `hello.m`, this can be executed with the following commands:

```
>> x = Hello();
>> x.greet();
Hello!
```

CISCO IOT:



The screenshot shows a web browser window displaying the Cisco IOT course page on NetAcad. The browser's address bar shows the URL: `1578393591.netacad.com/courses/964990/pages/first-time-in-this-course?module_item_id=64699438`. The page has a sidebar on the left with navigation links: Home, Modules, Discussions, Grades, Assignments, Quizzes, My NetAcad, Account, Dashboard, Courses, Calendar, Inbox, and Help. The main content area is titled "First Time in This Course" and features a "Welcome" banner with four images. Below the banner, the text reads: "This Introduction to Internet of Things 2.0 course is designed for people wishing to explore the Internet of Things and the impact it has on our everyday lives. It is not the intention of this course to teach about the IoT in depth but to provide a general understanding of the IoT and how it allows for digitization of daily tasks." It then encourages enrollment in additional Cisco networking and IT related courses, listing "Connecting Things, Big Data and Analytics, and IoT Security" as examples. It also mentions the Cisco Networking Academy Facebook site for peer support. At the bottom, it states that all quizzes and assessments can be accessed through the "Modules" section and provides a "Next" button to go to the resources and support page. The Windows taskbar at the bottom shows the time as 18:50 on 06-07-2020.

Cisco x Introdu x Solve A x Dashbo x SAP Sci x C Tutor x 19_ds_c x MATLAB x Cisco H x Slack x +

static-course-assets.s3.amazonaws.com/12IoT20/en/index.html#1.1.1.1

Chapter 1
Everything is Connected

1.1
Digital Transformation

1.1.1
Digitization Transforms Business

1.1.1.1
The Evolution of Digital Transformation

Networking Academy

7.4 billion - people on the planet

30 billion - devices connected to the internet by 2020

6.58 - average number of connected devices per consumer in 2020

44% - children under the age of 1 use smart devices

1.4 million - number of pacemakers in use by 2023

15 million - Fitbit exercise monitors sold in 2017

20 billion - Euros to be spent on artificial intelligence by the EU by 2020

The Evolution of Digital Transformation

Tell the truth how many of you could actually make it through a day without your smartphone?

In our world today, there are more smart devices than there are people. A growing number of people are connected to the internet, in one way or another, 24 hours a day. An ever-increasing number of people have, and rely on, three, four, or more smart devices. These might include smartphones, exercise and health monitors, e-readers, and tablets. As shown in Figure 1, by 2020, it is forecast that each consumer will have an average of 6.58 smart devices.

How is it possible for so many devices to be connected?

Modern digital networks make all of this possible. The world is quickly being covered

1 2 Figures

18:54

06-07-2020