Octave basics

November 27, 2018

No Binder links because Binder does not work with Octave. You are recommended to only use this as a reference while programming in the Octave or Matlab application.

We are starting from the beginning. Pre-Thanksgiving Octave guide is at octave_basics_original.ipynb (octave_basics_original.ipynb).

Assign a value to a variable with the '=' operator:

In [1]:

```
a = 1
```

```
a =  1
```

Matlab is primarily used with matrixes and arrays (single-row matrix).

Create an array with brackets and numbers seperated by spaces:

In [2]:

```
array_1x3 = [1 2 3]
```

```
array_1x3 =

   1   2   3
```

Check the dimentions with the `size()` function:

In [3]:

```
size(array_1x3)
```

```
ans =

   1   3
```

Note than when executing an expression, the answer is prefaced with `ans =`.

To create a matrix, seperate rows with a semicolon:

In [4]:

```
array_3x1 = [1; 2; 3]
```

```
array_3x1 =

   1
   2
   3
```

Let's create a 3x3 matrix and explore some functions.

In [6]:

```
% Comments use a percent sign in Octave/Matlab

matrix_3x3 = [1 2 3; 4 5 6; 7 8 9]

```

```
matrix_3x3 =

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

In [7]:

```
% For a matrix, sum() will add up the columns and return a single-row array

sum(matrix_3x3)

```

```
ans =

   12   15   18
```

In [8]:

```
% For an array, sum() will add up the elements and return a single number

sum(sum(matrix_3x3))

```

```
ans =  45
```

In [9]:

```
% diag() returns the diagonal from top left to bottom right

diag(matrix_3x3)

```

```
ans =

   1
   5
   9
```

In [10]:

```
% Flip left-right with fliplr()

fliplr(matrix_3x3)

```

```
ans =

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

In [11]:

```
% Flip up-down with flipud()

flipud(matrix_3x3)

```

```
ans =

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

In [14]:

```
% Let's put some things together to get the anti-diagonal (bottom-left to uppe
% Looking for [7; 5; 3]

matrix_3x3

diag(flipud(matrix_3x3))
```

```
matrix_3x3 =

   1   2   3
   4   5   6
   7   8   9

ans =

   7
   5
   3
```

In [15]:

```
% putting a single quote after a variable name will transpose the variable

matrix_3x3'
```

```
ans =

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

In [16]:

```
% Note the diagonal of a matrix is the same as diagonal of that matrix transpo
diag(matrix_3x3)

diag(matrix_3x3')
```

```
ans =

   1
   5
   9

ans =

   1
   5
   9
```

In [17]:

```
% Access elements of a matrix with parenthesis, seperate indexes with a comma
% Remember that indexing starts at 1

matrix_3x3

matrix_3x3(2,3)

```

```
matrix_3x3 =

   1   2   3
   4   5   6
   7   8   9

ans =  6
```

In [18]:

```
% You can slice an array or matrix in Octave/Matlab just like in Python

matrix_3x3(1:2, 1:2)

```

```
ans =

   1   2
   4   5
```

In [19]:

```
% The colon is basically a wildcard. Replacing a number with a colon will retu

% First row:
matrix_3x3(1,:)

```

```
ans =

   1   2   3
```

In [20]:

```
% Second column
matrix_3x3(:,2)

```

```
ans =

   2
   5
   8
```

In [23]:

```
% The whole matrix
matrix_3x3(:,:)

```

```
ans =

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

In [24]:

```
% You can use an entire matrix in a conditional expression. Note this creates

matrix_3x3 <= 5
```

```
ans =

  1  1  1
  1  1  0
  0  0  0
```

In [26]:

```
% You can also compare to a list or matrix *IF* they are either equal size or
% divisible into the larger:

matrix_3x3

% Array of length 3

matrix_3x3 <= [4, 5, 6]
```

```
matrix_3x3 =

   1   2   3
   4   5   6
   7   8   9

ans =

  1  1  1
  1  1  1
  0  0  0
```

In [27]:

```
% Array of length 3

% Note that each row of the matrix is compared by index to the element in the

matrix_3x3 <= [1, 2, 10]
```

```
ans =

  1  1  1
  0  0  1
  0  0  1
```

In [31]:

```
% Marix of size 3x3

matrix_3x3 == [1 2 3; 4 5 6; 7 8 9]
```

```
ans =

  1  1  1
  1  1  1
  1  1  1
```

In [51]:

```
% Or Call the eq() function

eq(matrix_3x3, [1 2 3; 4 5 6; 7 8 9])

```

```
ans =

  1  1  1
  1  1  1
  1  1  1
```

In [32]:

```
matrix_4x4 = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]

```

```
matrix_4x4 =

    1    2    3    4
    5    6    7    8
    9   10   11   12
   13   14   15   16
```

In [33]:

```
matrix_2x2 = [1 2; 3 4]

```

```
matrix_2x2 =

   1   2
   3   4
```

In [34]:

```
% Think this will work?

matrix_4x4 == matrix_2x2

```

```
error: mx_el_eq: nonconformant arguments (op1 is 4x4, op2 is 2x2)
```

In [38]:

```
% How about this?

matrix_4x4 == [5 6]

```

```
error: mx_el_eq: nonconformant arguments (op1 is 4x4, op2 is 1x2)
```

In [40]:

```
% The any() function is basically a series of `if` statements, will be 1 if an

% Note the placement of the closing parenthesis

matrix_4x4(1,:)

any(matrix_4x4(1,:) < 5)

```

```
ans =

   1   2   3   4

ans = 1
```

In [41]:

```
% The all() function is basically a series of `and` statements, will be 1 only

matrix_4x4(1,:)

all(matrix_4x4(1,:) < 5)

```

```
ans =

   1   2   3   4

ans = 1
```

In [43]:

```
% any() and all() again

% Note the placement of the closing parenthesis

matrix_4x4(2,:)

any(matrix_4x4(2,:) < 7)

all(matrix_4x4(2,:) < 7)

```

```
ans =

   5   6   7   8

ans = 1
ans = 0
```

In [44]:

```
% Let's go back to our 3x3
matrix_3x3

```

```
matrix_3x3 =

   1   2   3
   4   5   6
   7   8   9

```

In [45]:

```
% The dot . operator runs a component-wise application of the operator that fo

% Multiply by 2:

matrix_3x3 .* 2

```

```
ans =

    2    4    6
    8   10   12
   14   16   18
```

In [46]:

```
% Elements squared

matrix_3x3 .^ 2
```

```
ans =

    1    4    9
   16   25   36
   49   64   81
```

In [47]:

```
% Recall these variables from before:

array_1x3

array_3x1
```

```
array_1x3 =

   1   2   3

array_3x1 =

   1
   2
   3
```

In [48]:

```
% Anyone know how matrix mutiplication works?

array_1x3 * array_3x1
```

```
ans =  14
```

In [49]:

```
% Swap the order before multiplying:

array_3x1 * array_1x3
```

```
ans =

   1   2   3
   2   4   6
   3   6   9
```

The lab will have you play around and learn more about mutliplicatoin, as well as inverses and identities. We will cover those on Thursday in the notebook octave_basics_2 .