Octave basics part 2

November 29, 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.

You can vertically concatinate existing matrixes in a new matrix if the column lengths match

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

matrix_2x3 = [4 5 6; 7 8 9]

matrix_3x3 = [array_1x3; matrix_2x3]

```

```
array_1x3 =

   1   2   3

matrix_2x3 =

   4   5   6
   7   8   9

matrix_3x3 =

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

You can horizontally concatinate if the row lengths match

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

matrix_1x2 = [4 5]

matrix_1x5 = [array_1x3, matrix_1x2]

```

```
array_1x3 =

   1   2   3

matrix_1x2 =

   4   5

matrix_1x5 =

   1   2   3   4   5
```

In [3]:
```
% Last expression automatically stored in variable 'ans'

[array_1x3; matrix_2x3]

```

```
ans =

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

In [4]:
```
ans

```

```
ans =

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

In [5]:
```
% Semicolon at the end of the line will supress output

ans;

```

Some basic MATLAB functions:

In [6]:
```
abs(-10)

```

```
ans =  10
```

In [7]:
```
sqrt(9)

```

```
ans =  3
```

In [8]:
```
disp('Hello CISC 106')

```

```
Hello CISC 106
```

In [9]:
```
%ones(rows,cols) -- Matrix of all ones

ones(3, 3)

```

```
ans =

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

In [10]:
```
%zeros(rows,cols) -- Matrix of all zeros

zeros(3, 3)

```

```
ans =

   0   0   0
   0   0   0
   0   0   0
```

In [11]:
```
% rand(rows,cols) -- Matrix of uniformly distributed random elements

rand(3, 3)

```

```
ans =

   0.54123   0.98552   0.16629
   0.21249   0.14925   0.13327
   0.24804   0.40516   0.69135
```

In [12]:
```
% randi([min_val,max_val],rows,cols) -- Matrix of random integers between min_
randi([1, 6], 3, 3)

```

```
ans =

   1   2   2
   4   2   6
   4   6   2
```

Variable naming:

- Must start with a letter, and can have numbers, letters, underscores
- Case sensitive
- Don't need to declare
- You can make the variables any length -- but MATLAB only uses the first N characters to identify the variable – make first N characters unique
- N = number returned by - namelengthmax function
- Try: length_var_name = namelengthmax

In [13]:
```
namelengthmax()

```

```
ans =  63
```

In [14]:
```
length_var_name = namelengthmax

```

```
length_var_name =  63
```

In [15]:
```
length_var_name2 = namelengthmax

```

```
length_var_name2 =  63
```

In [16]:
```
length_var_name
```

```
length_var_name =  63
```

In [17]:
```
% Every value assigned to a variable is an array

v = 15
size(v)

v(1)

v = [15]
size(v)

v = [15 16 17 18]
size(v)
```

```
v =  15
ans =

   1   1

ans =  15
v =  15
ans =

   1   1

v =

   15   16   17   18

ans =

   1   4
```

Let's solve [this Hackerrank problem (https://www.hackerrank.com/challenges/diagonal-difference/problem)](https://www.hackerrank.com/challenges/diagonal-difference/problem) using Octave: Complete the diagonalDifference function in the editor below. It must return an integer representing the absolute diagonal difference.

Here is an example function from Professor Wassil's slides (MATLAB Ch. 2 & 5) -- You can download from Canvas, I linked to them on the main page, and they are also listed under Files.

```
function [hzn_concat, vert_concat] = concatMat(A,B)
    % Use percent sign for comments.
    % Sorry, no block comments
    % Just use lots of percent signs
    hzn_concat = [A,B];
    vert_concat = [A;B];
end
```

In [18]:
```
function diag_diff = diagonalDifference(a_matrix)
    regular_diag = sum(diag(a_matrix));
    anti_diag = sum(diag(flipud(a_matrix)));

    diag_diff = abs(regular_diag - anti_diag);

end
```

In [19]:
```
matrix_3x3 = [1 2 3; 7 8 9; 6 2 6]

d_diff = diagonalDifference(matrix_3x3)
```

```
matrix_3x3 =

   1   2   3
   7   8   9
   6   2   6

d_diff =  2
```

In [20]:
```
% Function definition from Wassil's slides
function [hzn_concat, vert_concat] = concatMat(A,B)
    hzn_concat = [A,B];
    vert_concat = [A;B];
end
```

In [21]:
```
matrix_1x3_a = [1 5 9]
matrix_1x3_b = [10 15 19]

[h_cat, v_cat] = concatMat(matrix_1x3_a, matrix_1x3_b)
```

```
matrix_1x3_a =

   1   5   9

matrix_1x3_b =

   10   15   19

h_cat =

    1    5    9   10   15   19

v_cat =

    1    5    9
   10   15   19
```

In [22]:
```
% From Wassil's PDF chap 2 & 5, slide 6

A = 10:-2:-10

A(1,5)

A(1,6)

if ( A(1,5) == A(1,6) )
    disp("equal")
elseif ( A(1,5) > A(1,6) )
    disp("greater")
else
    disp("less than")
end
```

```
A =

   10    8    6    4    2    0   -2   -4   -6   -8  -10

ans =  2
ans = 0
greater
```

In [23]:
```

i
```

```
ans =  0 + 1i
```

In [24]:
```
j
```

```
ans =  0 + 1i
```

In [25]:
```
i == j
```

```
ans = 1
```

In [26]:
```
% Boolean examples

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

```
matrix_3x3 =

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

In [27]:
```
% Not of every element (0 if not 0, 1 if 0)

~matrix_3x3

~[1 0 3]

```

```
ans =

  0  0  0
  0  0  0
  0  0  0

ans =

  0  1  0
```

In [28]:
```
matrix_3x3 & 1

```

```
ans =

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

In [29]:
```
matrix_3x3 | 1

```

```
ans =

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

In [30]:
```
% How to find index with single number?

matrix_3x3(8)

% Start at (1, 1) and go down through the first column, then continue
% counting at (1,2) and continue down the second column, etc., until you
% find the 8th element

```

```
ans =  6
```