MATLAB/Octave matrices vs. Python NumPy arrays

Note: in order to use the “np" shortcut notation for the numpy package, make sure you import the NumPy package as follows

```
>>> import numpy as np
```

	MATLAB/Octave matrices & vectors	NumPy arrays

Matrices (here: 3x3 matrix)	`octave:1> A = [ 1 2 3; 4 5 6; 7 8 9]`
	`A =`
	`1 2 3`
	`4 5 6`
	`7 8 9`
		`>>> A = np.array([ [1,2,3], [4,5,6], [7,8,9] ])`
		`>>> A`
		`array([[1, 2, 3],`
		`[4, 5, 6],`
		`[7, 8, 9]])`

Access rows (here: first row)	`octave:10> A(1,:)`
	`ans =`
	`1 2 3`
		`>>> A[0,]`
		`array([1, 2, 3])`

Access columns (here: first column)	`octave:11> A(:,1)`
	`ans =`
	`1`
	`4`
	`7`
		`>>> A[:,0]`
		`array([1, 4, 7])`
		`>>> A[:,[0]]`
		`array([[1],`
		`[4],`
		`[7]])`

Access elements (here: first element)	`octave:8> A(1,1)` `ans = 1`	`>>> A[0,0]` `1`

1-D column vector	`octave:3> a = [ 1; 2; 3]`
	`a =`
	`1`
	`2`
	`3`
		`>>> a = np.array([[1],[2],[3]])`
		`>>> a`
		`array([[1],`
		`[2],`
		`[3]])`

1-D row vector	`octave:4> b = [1 2 3]`
	`b =`
	`1 2 3`
		`>>> b = np.array([1,2,3])`
		`>>> b`
		`array([1, 2, 3])`

	MATLAB/Octave matrices & vectors	NumPy arrays

Random m x n matrix	`octave:6> rand(3,2)`
	`ans =`
	
	`0.21977 0.10220`
	`0.38959 0.69911`
	`0.15624 0.65637`
		`>>> np.random.rand(3,2)`
		`array([[ 0.29347865, 0.17920462],`
		`[ 0.51615758, 0.64593471],`
		`[ 0.01067605, 0.09692771]])`

Zero-matrix, m x n	`octave:16> zeros(3,2)`
	`ans =`
	
	`0 0`
	`0 0`
	`0 0`
		`>>> np.zeros((3,2))`
		`array([[ 0., 0.],`
		`[ 0., 0.],`
		`[ 0., 0.]])`
		`>>>`

m x n matrix of ones	`octave:36> ones(3,2)`
	`ans =`
	
	`1 1`
	`1 1`
	`1 1`
		`>>> np.ones([3,2])`
		`array([[ 1., 1.],`
		`[ 1., 1.],`
		`[ 1., 1.]])`

Identity matrix	`octave:39> eye(3)`
	`ans =`
	
	`Diagonal Matrix`
	
	`1 0 0`
	`0 1 0`
	`0 0 1`
		`>>> np.identity(3)`
		`array([[ 1., 0., 0.],`
		`[ 0., 1., 0.],`
		`[ 0., 0., 1.]])`

Matrix diagonal (left-upper corner to right lower)	`octave:40> diag(A)`
	`ans =`
	
	`1`
	`5`
	`9`
		`>>> np.diagonal(A)`
		`array([1, 5, 9])`
		`>>> np.diagonal([A])`
		`array([[1],`
		`[2],`
		`[3]])`

	MATLAB/Octave matrices & vectors	NumPy arrays

Diagonal matrix from a column vector	`octave:42> diag(a)`
	`ans =`
	
	`Diagonal Matrix`
	
	`1 0 0`
	`0 2 0`
	`0 0 3`
		`>>> np.diag(a[:,0])`
		`array([[1, 0, 0],`
		`[0, 2, 0],`
		`[0, 0, 3]])`

Matrix-scalar multiplication (*), subtraction (-), addition (+), division (/)	`octave:18> A * 2`
	`ans =`
	
	`2 4 6`
	`8 10 12`
	`14 16 18`
		`>>> A * 2`
		`array([[ 2, 4, 6],`
		`[ 8, 10, 12],`
		`[14, 16, 18]])`

Matrix element-wise power	`octave:23> A.^2`
	`ans =`
	
	`1 4 9`
	`16 25 36`
	`49 64 81`
		`>>> np.power(A,2)`
		`array([[ 1, 4, 9],`
		`[16, 25, 36],`
		`[49, 64, 81]])`

Element-wise matrix multiplication	`octave:32> A .* A`
	`ans =`
	
	`1 4 9`
	`16 25 36`
	`49 64 81`
		`>>> A * A`
		`array([[ 1, 4, 9],`
		`[16, 25, 36],`
		`[49, 64, 81]])`

Matrix-multiplication	`octave:31> A * A`
	`ans =`
	
	`30 36 42`
	`66 81 96`
	`102 126 150`
		`>>> np.dot(A,A)`
		`array([[ 30, 36, 42],`
		`[ 66, 81, 96],`
		`[102, 126, 150]])`

	MATLAB/Octave matrices & vectors	NumPy arrays

Matrix transpose	`octave:24> A'`
	`ans =`
	
	`1 4 7`
	`2 5 8`
	`3 6 9`
		`>>> A.transpose()`
		`array([[1, 4, 7],`
		`[2, 5, 8],`
		`[3, 6, 9]])`