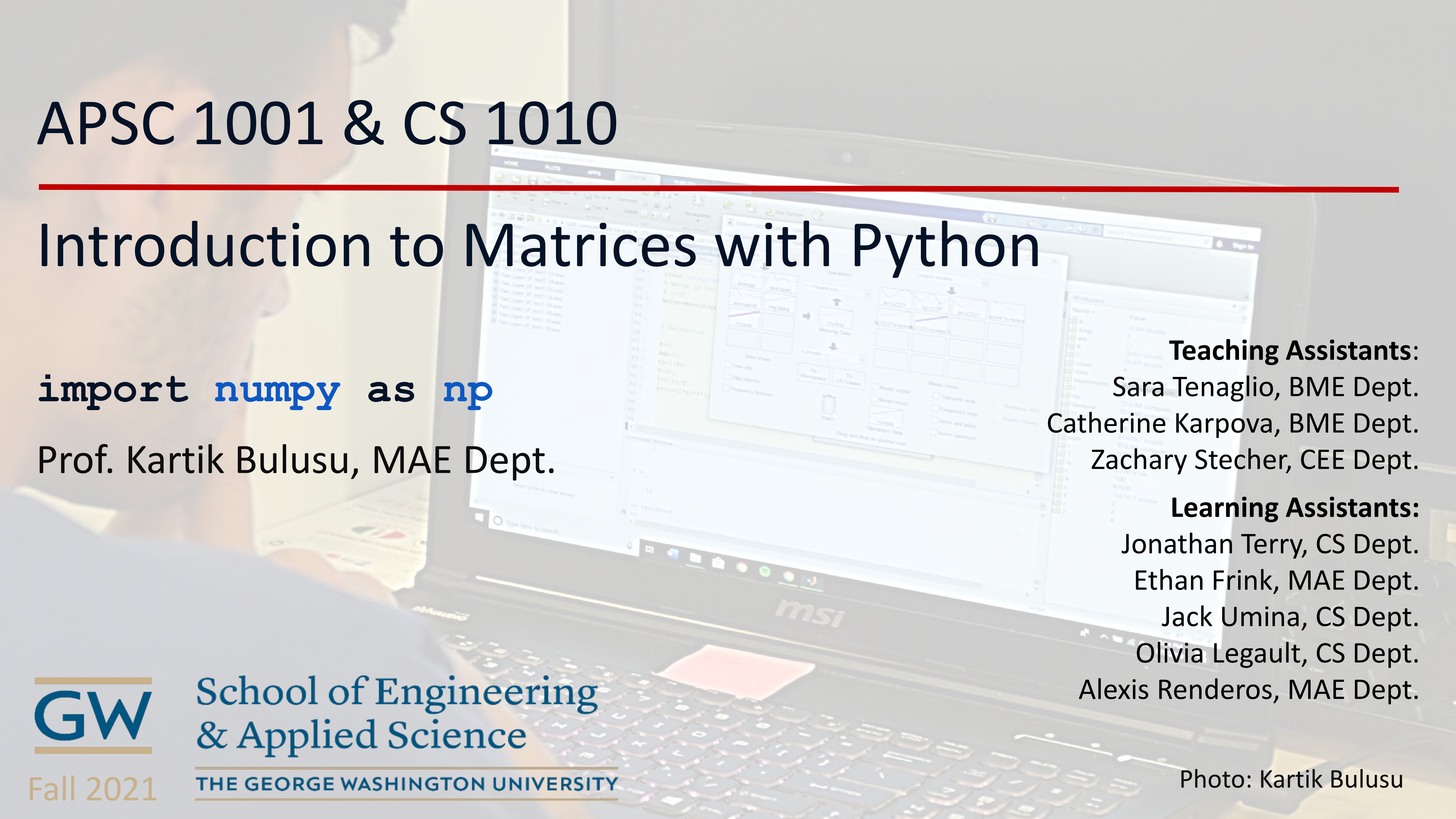


APSC 1001 & CS 1010

Introduction to Matrices with Python

```
import numpy as np
```

Prof. Kartik Bulusu, MAE Dept.

Teaching Assistants:
Sara Tenaglio, BME Dept.
Catherine Karpova, BME Dept.
Zachary Stecher, CEE Dept.

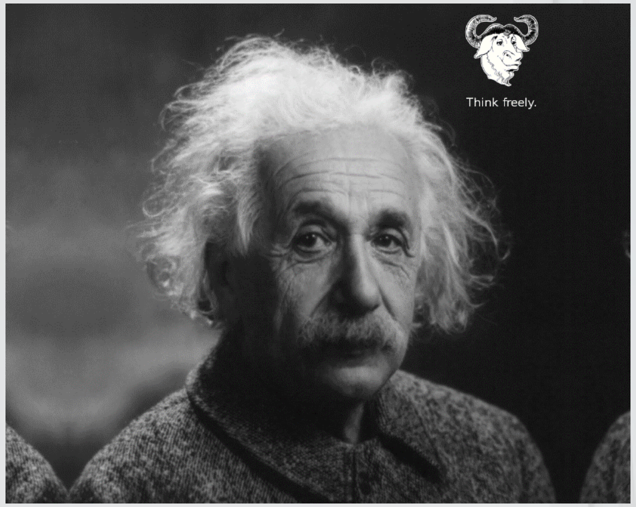
Learning Assistants:
Jonathan Terry, CS Dept.
Ethan Frink, MAE Dept.
Jack Umina, CS Dept.
Olivia Legault, CS Dept.
Alexis Renderos, MAE Dept.

GW School of Engineering & Applied Science
THE GEORGE WASHINGTON UNIVERSITY

Fall 2021

Photo: Kartik Bulusu

Photo: Kartik Bulusu

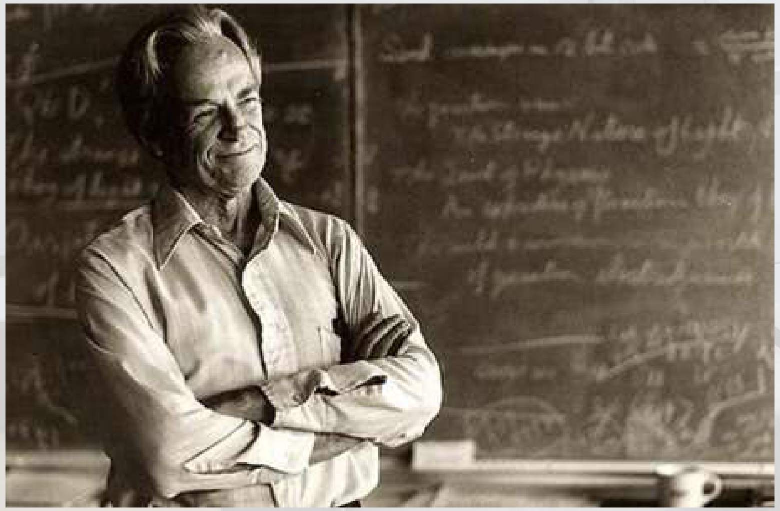


What patterns do you notice ?

Digital image is a matrix

These images contain elements of “uint8” data type

$$\begin{bmatrix} 49 & 49 & \cdots & 34 & 35 & 35 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 40 & 34 & \cdots & 51 & 49 & 46 \end{bmatrix}$$



Python:

```
>>> import matplotlib.pyplot as plt
>>> img = plt.imread('name')
>>> plt.imshow(img, cmap=plt.cm.hot)
>>> plt.show()
```

python™

What is a Matrix ?

DATA

- Arranged in ROWS and COLUMNS
- Typically carries a MEANING

DATA

- Rectangular ARRAY of numbers

ARRAYS

- Two-dimensional arrays
- *m* rows and *n* columns

Source: http://giphy.com/search/matrix-gif

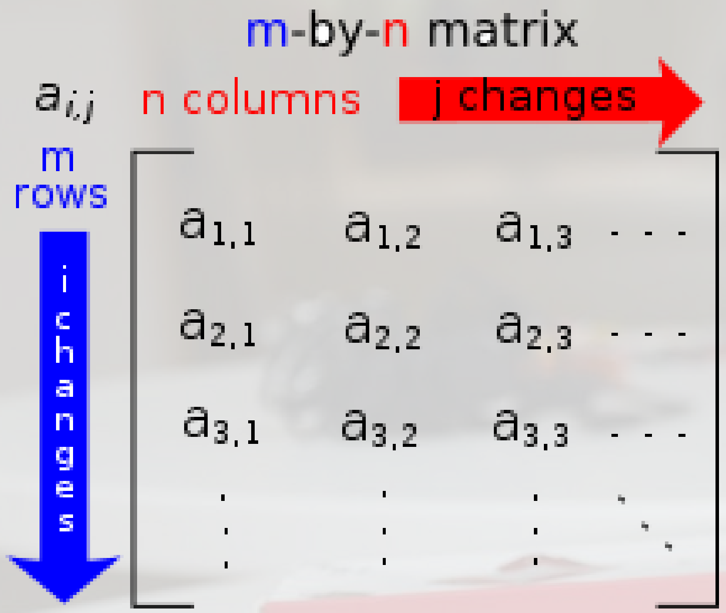
$$\begin{bmatrix} 1 & -4 \\ 9 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 15 & 3 & 9 \\ 2 & 5 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 11 & 7 \\ 4 & 2 \\ 6 & 9 \\ 3 & 1 \end{bmatrix}$$

Photo: Kartik Bulusu

Bookkeeping in a Matrix



m-by-n matrix
$a_{i,j}$
n columns
j changes
m rows
i changes
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ . . .
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$ . . .
$a_{3,1}$ $a_{3,2}$ $a_{3,3}$ . . .

Source: http://en.wikipedia.org/wiki/Matrix_(mathematics)

The ORDER of a matrix

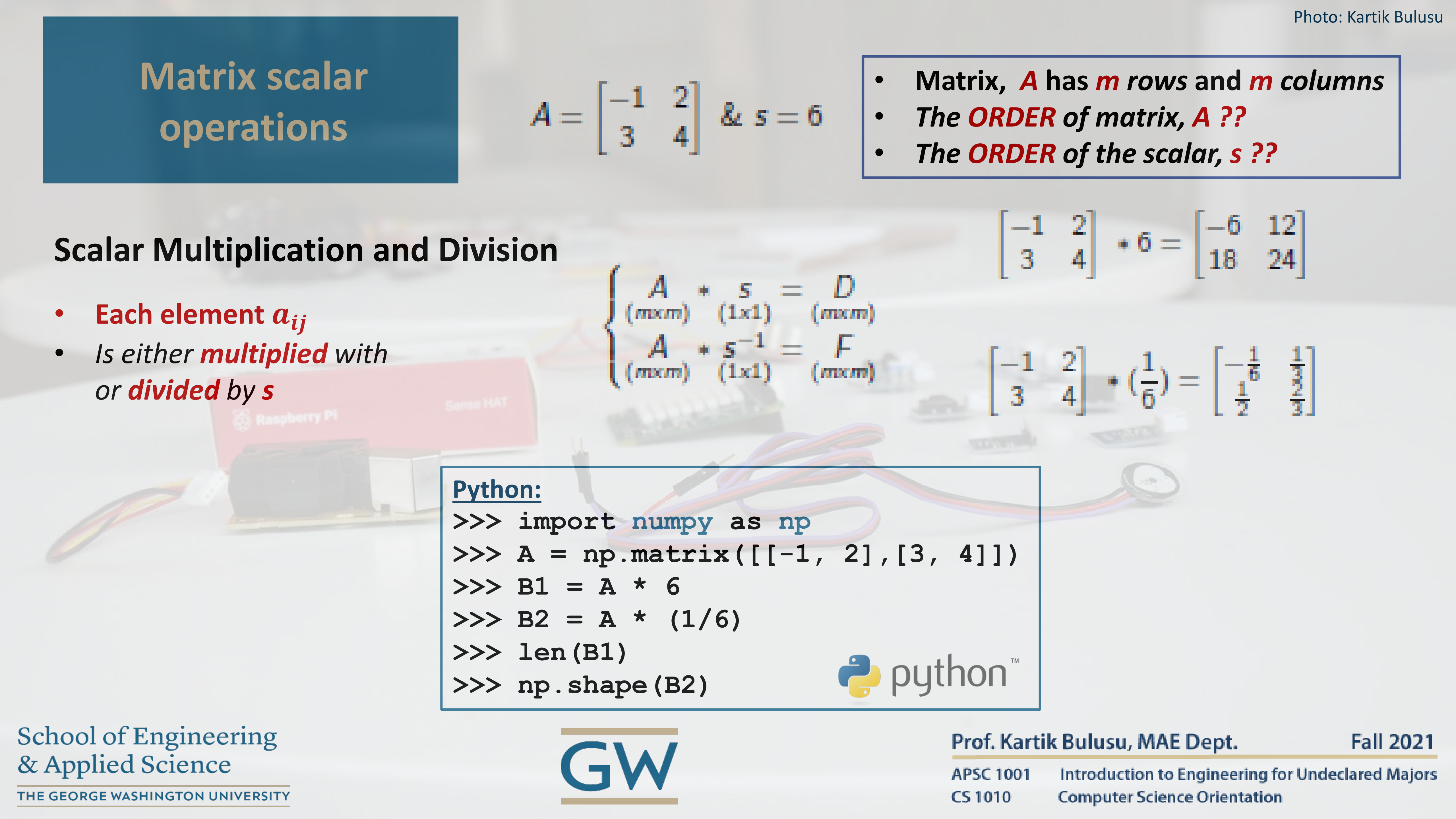
- $A_{m \times n}$ *is* $m \times n$
- *Read as "m-by-n"*

a_{ij} *is called an ELEMENT*

- *at the ith row and jth column of A*

Python:

```
>>> import numpy as np
>>> A = np.matrix([[-1, 2],[3, 4]])
>>> A[0,0]
>>> A[0,:]
>>> A[:,0]
>>> A[1,0]
```



Matrix scalar operations

$$A = \begin{bmatrix} -1 & 2 \\ 3 & 4 \end{bmatrix} \;\&\; s = 6$$

- Matrix, *A* has *m rows* and *m columns*
- *The ORDER of matrix, A ??*
- *The ORDER of the scalar, s ??*

Scalar Multiplication and Division

- **Each element a_{ij}**
- *Is either **multiplied** with or **divided** by **s***

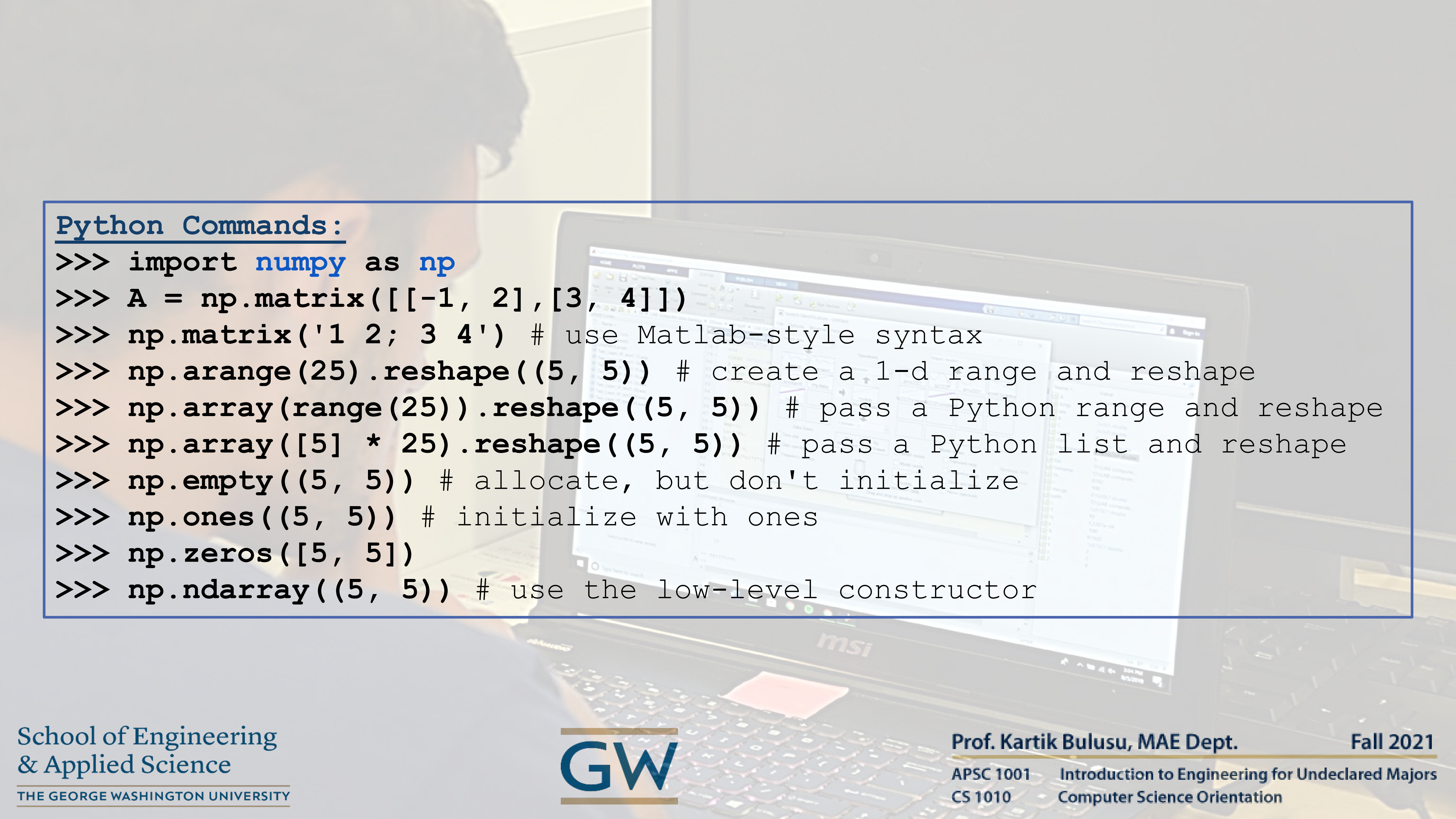
$$\begin{cases} \underset{(mxm)}{A} * \underset{(1x1)}{s} = \underset{(mxm)}{D} \\ \underset{(mxm)}{A} * \underset{(1x1)}{s^{-1}} = \underset{(mxm)}{F} \end{cases}$$

$$\begin{bmatrix} -1 & 2 \\ 3 & 4 \end{bmatrix} * 6 = \begin{bmatrix} -6 & 12 \\ 18 & 24 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 2 \\ 3 & 4 \end{bmatrix} * \left(\frac{1}{6}\right) = \begin{bmatrix} -\frac{1}{6} & \frac{1}{3} \\ \frac{1}{2} & \frac{2}{3} \end{bmatrix}$$

Python:

```
>>> import numpy as np
>>> A = np.matrix([[-1, 2],[3, 4]])
>>> B1 = A * 6
>>> B2 = A * (1/6)
>>> len(B1)
>>> np.shape(B2)
```



Python Commands:

```
>>> import numpy as np
>>> A = np.matrix([[-1, 2],[3, 4]])
>>> np.matrix('1 2; 3 4') # use Matlab-style syntax
>>> np.arange(25).reshape((5, 5)) # create a 1-d range and reshape
>>> np.array(range(25)).reshape((5, 5)) # pass a Python range and reshape
>>> np.array([5] * 25).reshape((5, 5)) # pass a Python list and reshape
>>> np.empty((5, 5)) # allocate, but don't initialize
>>> np.ones((5, 5)) # initialize with ones
>>> np.zeros([5, 5])
>>> np.ndarray((5, 5)) # use the low-level constructor
```