LARSON—MATH 310–CLASSROOM WORKSHEET 10
Getting started with Numpy in Jupyter Notebooks.

Set up your JUPYTER notebook for this work.

1. Create a new Jupyter Notebook with Python as the kernel.

   If you don't have a preferred Jupyter Notebooks et-up, use Google Colab (`colab.google.com`), login with your VCU credentials, and use that. It's free, and part of the Google suite VCU pays for. Google Coalb has all the packages we'll need from our text including `numpy` and `matplotlib`.

2. Call this file **310-c10.ipynb**.

3. Make sure you have PYTHON as the *kernel*.

Python Basics

1. What is a `list` in Python?

2. What is *list slicing*?

3. Why are lists *mutable*?

4. What is a similar data structure that is **not** mutable?

5. What is a *list comprehension* in Python?

6. What is a `dictionary` in Python?

7. How are dictionaries *different* than lists?

8. How are dictionaries similar to lists?

9. How do you code a `for loop` in Python? Write a for loop that prints the even integers from 0 to 20.

Coding Linear Algebra with Numpy and Scipy

10. What is a Numpy `array`?

```
from numpy import array
A = [[1, 2, 3], [4, 5, 6]]; A
```

11. Cast list-of-lists A to a Numpy array.

```
A = array(A); A
```

Then `print` A.

12. We can recover our list-of-lists data structure with the Numpy array `tolist` method.

```
L = A.tolist(); L
```

13. We can add two matrices that have the same *shape*.

```
from numpy import array
A = array([[1, 2, 3], [4, 5, 6]])
B = array([[1, 3, 5], [2, 4, 6]])
A + B
```

14. How can you *scale* (multiply) A by 2?

15. We can generate random matrices, with random shapes and random entries.

```
from numpy.random import seed, randint, choice
from sympy import Matrix
seed(2021)
m, n = randint(2, 4, 2)
X = [-3, -2, -1, 1, 2, 3, 4, 5]
A = Matrix(choice(X, (m, n)))
B = Matrix(choice(X, (m, n)))
```

Print m, n, A and B.

16. We can multiply a matrix times a vector with Numpy's `dot` function.

```
from numpy import array, dot
y = dot([[1, 2], [3, 4]], [5, 6]); y
```

`dot` finds a linear combination of the input matrix. Explain the calculation we see.

17. How is matrix multiplication defined?

18. One way to multiply matrices in Numpy is with the `dot` function/method applied directly to arrays.

```
import numpy as np
A = [[1, 2, 3], [4, 5, 6]]
B = [[1, 2], [3, 4], [5, 6]]
np.dot(A, B)
```

Explain what Numpy calculated.

19. Now *cast* A to a Numpy array and calculate again.

```
np.array(A).dot(B)
```

20. Now cast A and B to Numpy's `matrix` class, and repeat.

```
np.matrix(A) * np.matrix(B)
```

21. Generate matrices A, B, C, D and find their products - if this makes sense. What does this code test?

```
from numpy import array
A = array([[1, 2], [3, 4]])
B = array([[1, 2, 3], [4, 5, 6]])
C = array([[1, 2], [3, 4], [5, 6]])
D = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
for X in (A, B, C, D):
    for Y in (A, B, C, D):
        if X.shape[1] == Y.shape[0]:
            print(f'{X}\n{Y}\n= {X.dot(Y)}\n')
```

22. Now let's see how we can keep *exact* (non-rounded) data and symbolic variables with Sympy.

```
from sympy import Matrix
from sympy.abc import a, b, c, d
A = Matrix([[1, 2], [3, 4]])
A/2
```

Notice there are fractions and no decimals.

23. Now let's create a matrix with symbolic variables.

```
B = Matrix([[a, b], [c, d]])
B/2
```

24. Find the sum `A+B`. What do you notice?

25. Find the product `A*B`. What do you notice?

26. All-zero ("zero") matrices are important in linear algebra. There are built-in constructors for these matrices in both Numpy and Sympy.

```
import numpy as np
import sympy as sp
np.zeros((2,3))
```

What do you notice?

27. Here's Sympy's 0-matrix constructor.

```
sp.zeros(2,3)
```

What do you notice?

28. The square matrix of all-zeros with 1's on the diagonal is the *identity matrix* (called I, or I_n if the shape is important). These are important in linear algebra. There are built-in constructors for these matrices in both Numpy and Sympy.

```
np.eye(3)
```

What do you notice?

29. Here's Sympy's identity (I) matrix constructor.

```
sp.eye(3)
```

30. The following code uses the `matplotlib` graphics library to produce a picture. Run it, and then let's try to explain it.

```
from numpy import matrix, sin, cos, tan, pi, eye
import matplotlib.pyplot as plt
t = pi / 6
A = matrix([[cos(t), sin(t)], [-sin(t), cos(t)]])
B = matrix([[1, 0], [0, -1]])
C = matrix([[cos(t), -sin(t)], [sin(t), cos(t)]])
D = C * B * A
E = (eye(2)+D) / 2
x = matrix([[5], [5]])
y = D * x
z = E * x
plt.plot([0, 10], [0, 10 * tan(t)])
plt.plot([x[0, 0], y[0, 0]], [x[1, 0], y[1, 0]])
plt.plot([x[0, 0], z[0, 0]], [x[1, 0], z[1, 0]])
plt.text(x[0, 0], x[1, 0], 'x', fontsize=18)
plt.text(y[0, 0], y[1, 0], 'y', fontsize=18)
plt.text(z[0, 0], z[1, 0], 'z', fontsize=18)
plt.axis('scaled'), plt.xlim(0, 10), plt.ylim(0, 6)
```

Getting your classwork recorded

1. Save your Jupyter Notebook file (`310-c10.ipynb` file).
2. Send me an email (`clarson@vcu.edu`) with an informative header like "Math 310 - c10 homework attached" (so that it will be properly recorded and findable when I search for these emails).
3. Remember to **attach** the `.ipynb` file for this assignment.