Last name ________________________

First name ________________________

LARSON—MATH 310–BONUS WORKSHEET 01
Numpy Basics.

1. Start the Chrome browser.
2. Go to `https://cocalc.com`
3. Log in.
4. You should see an existing Project for our class. Click on that.
5. Click "New", then "Jupyter Notebook", then call it **310-b01**.
6. Make sure you have PYTHON as the *kernel*.

Instructions: Run all of the code chunks. **Annotate** your Jupyter notebook cells so that it is clear to anyone what problem your work corresponds to.

`Numpy` is the standard library for numerical linear algebra computing. It is based on *arrays* holding uniform type data (floats). Under the hood are the well-researched `BLAS` and `Lapack` libraries. Read more here: `https://en.wikipedia.org/wiki/NumPy`.

Matrix basics

1. A matrix can be represented as an array of arrays.

```python
from numpy import array
A = array([[1, 2, 3], [4, 5, 6]])
print(A)
type(A)

```

2. We can add matrices.

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

```

3. We can multiply a matrix times a vector.

```python
from numpy import array, dot
A = array([[1, 2], [3, 4]])
A.dot([5, 6])

```

4. We can multiply a 2×3 matrix times a 3×2 matrix.

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

```

5. We can also multiply these matrices using the overloaded "*" operator

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

```

6. It can be useful to initialize a vector or matrix with all 0's.

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

```

7. We can create and *identity matrix* of any size.

```python
 import numpy as np
np.identity(3)

```

Matrix inversion

8. One way to find the inverse of an invertible matrix is with the "inv" function.

```python
A = [[1, 2], [2, 1]]
from numpy.linalg import inv
inv(A)

```

9. Let's check that what we got really is the inverse.

```python
A = [[1, 2], [2, 1]]
from numpy.linalg import inv
B=inv(A)
np.matrix(A) * np.matrix(B)

```

10. We can also find inverses with the overloaded Python exponentiation operator "**".

```python
A = [[1, 2], [2, 1]]
from numpy import matrix
matrix(A)**(-1)

```

Matrix rank

11. We can find the rank (which is the same as both the column rank and row rank). The matrix here is 3×3, so if the rank is less than 3 we know that both the rows are linearly dependent and the columns are linearly dependent.

```python
from numpy import *
A = array([[1, 2, 3], [2, 3, 4], [3, 4, 5]])
linalg.matrix_rank(A)

```

Solving matrix equations

12. Given a matrix A and a vector $\hat{b}$ we want to find a vector $\hat{x}$ so that $A\hat{x} = \hat{b}$. You should check (by hand) that the produced solution works.

```python
import numpy as np
A = np.array([[1, 2], [3, 4]])
b = np.array([8, 18])
print(("Solution of linear equations:", np.linalg.solve(A, b)))

```

Eigenvalues

13. We can find eigenvalues of a square matrix. The eigenvalues are, in general comples, so the output is represented as complex numbers.

```python
from numpy.linalg import eig
A = [[3, -2, 2], [2, -1, 4], [2, -2, 1]]
lmd, vec = eig(A)
lmd

```

14. We can also print the corresponding eigenvectors.

```python
print(vec)

```

15. Let's check that the first listed eigenvalue and first listed eigenvector actually work. We should have, for instance, that $A * v0 = lambda * v0$.

```python
from numpy.linalg import eig
A = [[3, -2, 2], [2, -1, 4], [2, -2, 1]]
lmd, vec = eig(A)
lambda0 = lmd[0]
v0 = vec[0]
print(np.dot(A,v0))
D = lambda0*np.identity(3)
print(np.dot(D,v0))

```

Getting your homework recorded

When you are done,

1. Click the “Print” menu choice (under “File”) and make a pdf of this worksheet (html is OK too).
2. Send me an email (`clarson@vcu.edu`) with an informative header like
   “Math 310 - b01 worksheet attached” (so that it will be properly recorded).
3. Remember to attach your homework worksheet!