# NumPy and Matplotlib Worksheet

**NumPy** is a Python library for numerical computing with support for multi-dimensional arrays and matrices and that includes a large set of mathematical functions that operate on these arrays.

**Matplotlib** is a Python library for plotting that can be used in conjuction with **NumPy**.

Extensive resources and documentation for both libraries can be found online.

This worksheet consists of several numbered questions and/or blanks on **NumPy** and **Matplotlib** for you to complete .

## Part I. NumPy: Introduction and Matrix Operations

Consider the following matrix and row vector:

3 0 2 -1

A = 2 0 2 x = 1

0 1 1 2

1. Recalling Python tables (a.k.a. multidimensional lists), this matrix and vector can be defined as follows:

myTable1 = [[3,0,2],[2,0,2],[0,1,1]]

myVect1 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ <- fill in the blank

1. The **NumPy** library can view these as matrices or vectors and operate on them accordingly.

First type:

import numpy as np

Then try this:

np.transpose(myTable1)

What happened? What data type was output?

1. **NumPy**’s functions work fairly well on Python lists, but they work even better on a special data type called a **NumPy** ndarray (think: n-dimensional array), also called a **NumPy** array. You can convert a Python table into a **NumPy** array by typing:

myArray1 = np.array(myTable1)

Create another **NumPy** array this way:

myArray2 = np.array([[1,2,3],[4,5,6],[7,8,9]])

1. Try out the **NumPy** functions listed in the following table and fill out the columns on what you think the command will do and what it actually does.

(If you cannot figure out what something does, make a note of that, move on and ask someone about it soon. Note that you could also create some smaller, simpler arrays e.g. myArray3 = np.array([1,0],[0,1])) to help figure out what some of the functions do.)

|  |  |  |
| --- | --- | --- |
| **Function** | **What do you think it**  **will do/return?** | **Did it do what you expected?**  **If not, what did it do?** |
| np.shape(myArray1) |  |  |
| np.sum(myArray1) |  |  |
| np.transpose(myArray1) |  |  |
| np.dot( myArray1, [1,1,1]) |  |  |
| np.linalg.norm(myVect1) |  |  |
| np.dot(myArray1, myArray2) |  |  |
| myArray1 \* myArray2  (or np.multiply(myArray1, myArray2) |  |  |
| myArray1 + myArray2  (or np.add(myArray1, myArray2)) |  |  |
| myA1Inv = np.linalg.inv(myArray1) |  |  |
| np.linalg.det(myArray1) |  |  |
| myArray1[0] |  |  |
| myArray1[:2] |  |  |
| myArray1[0,2]  (yes that is supposed to be a comma) |  | Is this the way list indexing works? |
| myArray1[0][2] |  | Is this the way list indexing works? |
| myArray1[:,1] |  | This is useful. What does it do? |
| np.dot(myArray1, myA1Inv) |  |  |
| eq1 = np.array([[3,1], [1,2]])  eq2 = np.array([9,8])  np.linalg.solve(eq1, eq2) |  |  |
| np.zeros((3,5))  note the 2 parentheses on either side |  |  |
| np.ones((4,2)) |  |  |
| np.eye(3) |  |  |
| np.random.random((4,3)) |  |  |

Notes:

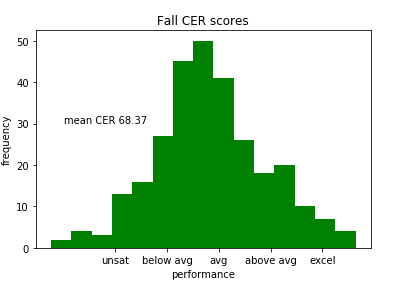
* Many other **NumPy** functions exist.
* Many of these functions are actually methods of the **NumPy** ndarray Class. So they can be called for a given ndarray object called myArray1 as: myArray1.methodName(). E.g, myArray1.transpose(). Try this for some of them.
* Ultimately, for better performance and more functions, you will want to consider using the library **scipy.linalg** instead of **numpy.linalg** for matrix computations. See <https://docs.scipy.org/doc/scipy/reference/tutorial/linalg.html> .

PART II: Plotting  
a) In class we plotted f(x) = 2x + 1 using matplotlib’s pyplot. Produce the following plot, where g(x) = 2x2 + 1  
Note that the plot includes:

* Gridlines
* A title “f(x) and g(x)” if you plot f and g together, otherwise “g(x)”
* g(x) is plotted with red triangles
* A legend
* Labels on the x-axis and y-axis

It may be helpful to use the following resources:

* <https://matplotlib.org/tutorials/introductory/pyplot.html>
* Your text (pgs 138-144, 200-202, 335-338)

b) In class we made a histogram using the data in the first row of the hist.csv file. Produce the following histogram for the second row of the hist.csv file.  
 Note that the histogram includes:

* A title “Fall CER scores”
* Labels on the x-axis and y-axis
* Green bars
* Text on the histogram, stating the mean CER score
* Labels along the x-axis:   
  unsat, below avg, avg, above avg, excel  
  at the x-locations 50, 60, 70, 80, and 90, respectively

It may be helpful to use the following resources:

* <https://matplotlib.org/gallery/pyplots/pyplot_text.html#sphx-glr-gallery-pyplots-pyplot-text-py>
* Your text (pgs 138-144, 200-202, 335-338)