CS315 **Selin Fildiş**
Spring 2017 21402228
Homework 1 CS315-1

Part A

In NumPy the variable "otherData" references the same location with "data" while in python's own list, it takes the data and puts it into a different memory location.
This means that when the variable "otherData" is updated in the NumPy version, the "data" variable will be updated as well.

Part B

NumPy matricies do a matrix multiplication in the case $A * B$ and also $A * *3$ would be equal to doing $A * A * A$ in the form of matrix multiplication while 2D arrays would do a simple multiplication such as multipling the element i_{1x1} with j_{1x1} where A_{ixi} and B_{jxj}. This is the same for exponents.

Part C

```
import numpy as np

into = input("Enter the file name")
f = open(into, 'r')
numberofnodes = f.readline()
if numberofnodes == "[num_nodes]":
    numberofnodes = f.readline()
    edgebeg = False
    A = np.zeros(shape=(int(numberofnodes), int(numberofnodes)))
    m = np.asmatrix(A)
    for line in f:
        if line == "[edges]":
            edgebeg = True
            break
    if edgebeg:
        for line in f:
            first = int(line[:1])
            last = int(line[-1:])
            m[first][last] = 1
            m[last][first] = 1
fo = open('output', 'w')
fo.write("Adajency Matrix:\n")
fo.write(m.dumps())


len = input("Enter the length of the path")
len = int(len)
```

```
lenMatrix = m**len
print 'paths: \n'
print lenMatrix.dumps()

fo.write("Paths Algorithm according to given number of paths: " + len + "\n")
fo.write(m.dumps())

f.close()
fo.close()
```

Part D

When the matrices grow large, the NumPy library conducts problems as it takes up too much space. In these cases, Scipy's sparse matrices may be used. Scipy is a library which is built on the Numpy library but has extra mechanisms built upon it. Also the calculation functions are optimized to make the code work faster.[1] As a different approach, the matrices can be stored as PyTables. This stores the Numpy arrays and enables easy access to the array which is now stored in the disk. But this methodology does not work with sparse matrices and must be converted to an array.[2]

Part E

```
def createNameMap(accounts):
    indices = dict()
    if __name__ == '__main__':
        for key in accounts:
            indices[key.getName()] = [key.getId()-1]
            # If we use dict inside this dict as well the above line of code wou
            # indices[key.getName()] = {'id': key.getId(), 'name': key.getName()
    return indices
```

[1] https://docs.scipy.org
[2] http://www.pytables.org