Numpy Basics

Abhishek Mukundan J021

J1

```
In [1]: import numpy as np
        import time
```

Matrix addition and substraction

```
In [2]: m1 = np.array([[3,2],[5,6]])
        m2 = np.array([[9,4],[1,8]])
        m1, m2

Out[2]: (array([[3, 2],
                [5, 6]]),
         array([[9, 4],
                [1, 8]]))
```

```
In [3]: print('Addition:')
        m1+m2

        Addition:

Out[3]: array([[12,  6],
               [ 6, 14]])
```

```
In [4]: print('Substraction:')
        m1-m2

        Substraction:

Out[4]: array([[-6, -2],
               [ 4, -2]])
```

Scalar Addition

```
In [6]: m1 + 200

Out[6]: array([[203, 202],
               [205, 206]])
```

Matrix Vector multiplication

In [7]:

```
m3 = np.random.randint(low=1, high=100, size=(3,3))
m4 = np.random.randint(low=1, high=100, size=(3,3))
print('Matrix 1:')
print(m3)
print('Matrix 2:')
print(m4)
print('Dot Product:')
print(np.dot(m3,m4))
```

```
Matrix 1:
[[33 10 67]
 [34 21 42]
 [62 70 28]]
Matrix 2:
[[92 36 75]
 [33 71  7]
 [ 9 33 95]]
Dot Product:
[[3969 4109 8910]
 [4199 4101 6687]
 [8266 8126 7800]]
```

Matrix Inversion

In [8]:

```
m5 = np.random.randint(low=1, high=10, size=(3,3))
print('Original Matrix:')
print(m5)
print('Determinant:')
print(np.linalg.det(m5))
print('Inverse Matrix:')
m6 = np.linalg.inv(m5)
print(m6)
```

```
Original Matrix:
[[4 4 3]
 [9 8 3]
 [6 3 3]]
Determinant:
-38.99999999999999
Inverse Matrix:
[[-0.38461538  0.07692308  0.30769231]
 [ 0.23076923  0.15384615 -0.38461538]
 [ 0.53846154 -0.30769231  0.1025641 ]]
```

Matrix Properties

In [9]:

```
m7 = np.random.randint(low=1, high=100, size=(3,3))
m8 = np.random.randint(low=1, high=100, size=(3,3))
print('Matrix 1')
print(m7)
print('Matrix 2')
print(m8)
```

```
Matrix 1
[[66 81 75]
 [77 49 17]
 [31 93 33]]
Matrix 2
[[59 89  7]
 [ 8 40 91]
 [83 69  4]]
```

- Verifying A + B = B + A

In [10]:

```
print('A+B')
print(m7+m8)
print('B+A')
print(m8+m7)
```

```
A+B
[[125 170  82]
 [ 85  89 108]
 [114 162  37]]
B+A
[[125 170  82]
 [ 85  89 108]
 [114 162  37]]
```

- Verifying A x B != B x A

In [11]:

```
print('A*B')
print(np.dot(m7,m8))
print('B*A')
print(np.dot(m8,m7))
```

```
A*B
[[10767 14289  8133]
 [ 6346  9986  5066]
 [ 5312  8756  8812]]
B*A
[[10964  9791  6169]
 [ 6429 11071  4283]
 [10915 10476  7530]]
```

- Verifying (A x B) x C = A x (B x C)

In [12]:

```
m9 = np.random.randint(low=1, high=100, size=(3,3))
print('Matrix 3')
print(m9)
```

```
Matrix 3
[[ 1  9 25]
 [74 89 96]
 [91 45 54]]
```

In [12]:

```
print('A*(B*C)')
print(np.dot(m7, np.dot(m8,m9)))
```

```
A*(B*C)
[[1734648 1925075  386096]
 [1135422 1260000  250524]
 [1179785 1310285  264603]]
```

In [13]:

```
print('(A*B)*C')
print(np.dot(np.dot(m7,m8),m9))
```

```
(A*B)*C
[[1808256 1734609 2080101]
 [1206316 1173838 1390870]
 [1455148 1223632 1449224]]
```

Matrix Addition by loops

In [14]:

```
m10 = np.random.randint(low=1, high=100, size=(10000,10000))
m11 = np.random.randint(low=1, high=100, size=(10000,10000))
print('Matrix 1')
print(m10)
print('Matrix 2')
print(m11)
```

```
Matrix 1
[[79 32 64 ... 61  8 55]
 [63 62 92 ... 77  7 95]
 [75 43 96 ... 29 60 70]
 ...
 [ 4 56 30 ... 94 82 76]
 [25 26 78 ... 50 58 33]
 [28 21 67 ... 83 17 32]]
Matrix 2
[[64 40 16 ...  6  3 72]
 [62 81 52 ... 72 63 10]
 [77 37 35 ... 42 43 10]
 ...
 [13 97 59 ... 89 31 56]
 [80 11 92 ... 77 57 83]
 [93 26 73 ... 46 61 27]]
```

In [16]:

```
r,c = m10.shape
print(r,c)
```

```
10000 10000
```

In [17]:

```
ans = []
start = time.time()
for i in range(r):
    row = []
    for j in range(c):
        row.append(m10[i,j] + m11[i,j])
    ans.append(row)
ans = np.array(ans).flatten().reshape(r,c)
print(time.time() - start)
```

```
75.96846675872803
```

In [18]:

```
start2 = time.time()
ans2 = m10 + m11
print(time.time() - start2)
```

```
0.2745342254638672
```

Matrix Multilication with loops

In [19]:

```
m12 = np.random.randint(low=1, high=100, size=(100,100))
m13 = np.random.randint(low=1, high=100, size=(100,100))
print('Matrix 1')
print(m12)
print('Matrix 2')
print(m13)
```

```
Matrix 1
[[13 29 44 ... 84 32 23]
 [95 65 71 ... 65  9 16]
 [53 15 78 ... 93 72 71]
 ...
 [31 84 10 ... 36 53 97]
 [ 1 96  7 ... 38 52 42]
 [18 70 48 ... 97 91  3]]
Matrix 2
[[19 37 10 ... 68 17 69]
 [93 38 35 ... 71 36 93]
 [40 83 64 ... 88 31 63]
 ...
 [66 62 18 ... 23 23 92]
 [72 13 81 ... 11 69 18]
 [23 46 99 ... 82 40 42]]
```

In [20]:

```
r,c = m12.shape
print(r,c)
```

```
100 100
```

In [21]:

```
l,t= m13.shape
print(l,t)
```

```
100 100
```

In [22]:

```
ans3 = np.zeros((r,t))
start3 = time.time()
for i in range(r):
    for j in range(t):
        for k in range(c):
            ans3[i][j] += m10[i][k] * m11[k][j]

print(time.time() - start3)
```

```
1.4436516761779785
```

In [23]:

```
start4 = time.time()
ans4 = np.dot(m12,m13)
print(time.time() - start4)
```

```
0.000997781753540039
```

In [ ]: