DFT_test2

April 1, 2019

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

```
In [2]: def DFT_fn(x):
            sze_x = np.size(x)
            X_val = np.zeros((sze_x
                                   ,),dtype=np.complex128)
            for m in range(0,sze_x):
                for n in range(0,sze_x):
                    X_val[m] += x[n]*np.exp(-np.pi*2j * m * n / sze_x)
            return X_val
```

```
In [3]: X = np.random.rand(1024,)
```

0.1 Now lets run our DFT algo

```
In [4]: DFT_fn(X)
```

```
Out[4]: array([ 5.19427754e+02+0.j        , -5.15631843e-01-7.50074858j,
                2.87225543e+00+5.39983673j, ..., -3.13363011e-01-3.38512972j,
                2.87225543e+00-5.39983673j, -5.15631843e-01+7.50074858j])
```

0.2 Now lets run the Numpy's FFT for comparision

```
In [5]: np.fft.fft(X)
```

```
Out[5]: array([ 5.19427754e+02+0.j        , -5.15631843e-01-7.50074858j,
                2.87225543e+00+5.39983673j, ..., -3.13363011e-01-3.38512972j,
                2.87225543e+00-5.39983673j, -5.15631843e-01+7.50074858j])
```

0.3 Now lets see if our implimentation of DFT is same as the FFT of numpy's

```
In [6]: np.allclose(DFT_fn(X),np.fft.fft(X))
```

```
Out[6]: True
```

0.4 Hurray!! Yes, both of them are equal