Vector Manipulation and Numpy tutorial

What is Numpy?

- NumPy is a Python package. It stands for 'Numerical Python'. It also has functions for working in the domain of linear algebra and matrices.
- Numpy Features:
  - Typed multidimentional arrays (matrices)
  - Fast numerical computations (matrix math)
  - High-level math functions

Why Use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Install Numpy (If you haven't already)

- Pip install numpy

Import Numpy

- import numpy

Numpy as np

- NumPy is usually imported under the np alias.
- import numpy as np

numpy Examples

Creating Array

```
import numpy as np

arr = np.array([2, 3, 4])
print (arr)

print (type(arr))
```

Dimensions in Arrays

```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3],
[4, 5, 6]]])
```

```
print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

Exercise 1
Try creating Arrays using (DON'T forget to give dimensions to your arrays)

- np.ones, np.zeros
- np.arange
- np.concatenate
- np.astype
- np.zeros_like, np.ones_like
- Np.random.random
- np.random.randint(range, size = ())

What do you see?

Pivoting Arrays

- Changing dimensions

1. Transpose

- Transpose is used to reverse or permute the axes of the array. You can specify which axes you want to use for your transpose operation using the "axes"parameter in np.tranpose.

```
transposed_array = np.transpose(array)
```

```
print(transposed_array)
```

2. Reshape

- Reshaping means changing the shape of an array.

```
Create a 1D Array containing 12 elements
newarr = arr.reshape(4, 3)
Newarr2 = arr.reshape(4,4)
print(newarr)
print(Newarr2) - What is wrong here?

Reshape into Unknown dimension
Try
Newarr3 = arr.reshape(2,2,-1)
```

3. Flatten

- The flatten operation changes the shape of the array to a 1-D format, and will contain all the elements which existed in the original array.

```
Create a 3D array
flattened_array = array.flatten()
print (flattened_array)
```

Indexing Arrays

- Indexing is nothing but using the location of an element in an array to extract it.
- Starts from 0

Create a random array with size (3,4,5)

```
print( array[0][0][0])
print( array[1,2,3])
print( array[-1,-1][-1])
```

What do you see?

Mathematical Operations of Arrays

Create two array with the same dimension
Try

```
np.add(array1,array2)
np.subtract(array1,array2)
np.multiply(array1,array2)
np.divide(array1,array2)
np.power(array1,array2)
np.mod(array1,array2)
```

Next try to create two arrays with different dimensions

When does the operations work and when does it not?

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

b = np.array([[2,3,4],
              [5,6,7],
              [8,9,10]])

o = np.matmul(a, b)
o1 = np.dot(a,b)
o2 = np.vdot(a,b)
```

What is this?