**Python Advance Assignment 1**

**1. What makes NumPy.shape() different from NumPy.size()?**

*size refers to how big your array will be. Moreover, if we can pass np.zeros(10) or np.zeros((10)). While the difference is subtle, size passed this way will create us a 1D array. You can give size=(n1, n2, ..., nn) which will create an nD array.*

*When we want multi dimensional arrays in python, array.reshape allows us to get from 1D* to an nD array. So, when we call shape, we get the N dimension shape of the array, *so we can see exactly how our array looks like.*

*In essence, size is equal to the product of the elements of shape.*

*The difference in name can be attributed to 2 parts: firstly, you can initialise your array with a size. However, you do not know the shape of it. So size is only for total number of elements. Secondly, how numpy was developed, different people worked on different parts of the code, giving different names to roughly the same element, depending on their personal vision for the code.*

**2. In NumPy, describe the idea of broadcasting.**

*Python | Broadcasting with NumPy Arrays. The term broadcasting refers to how numpy treats arrays with different Dimension during arithmetic operations which lead to certain constraints, the smaller array is broadcast across the larger array so that they have compatible shapes.*

***Broadcasting Rules:***   
***Broadcasting two arrays together follow these rules:***

1. *If the arrays don’t have the same rank then prepend the shape of the lower rank array with 1s until both shapes have the same length.*
2. *The two arrays are compatible in a dimension if they have the same size in the dimension or if one of the arrays has size 1 in that dimension.*
3. *The arrays can be broadcast together iff they are compatible with all dimensions.*
4. *After broadcasting, each array behaves as if it had shape equal to the element-wise maximum of shapes of the two input arrays.*
5. *In any dimension where one array had size 1 and the other array had size greater than 1, the first array behaves as if it were copied along that dimension.*

*import numpy as np*

*a = np.array([17, 11, 19]) # 1x3 Dimension array*

*print(a)*

*b = 3*

*print(b)*

*# Broadcasting happened because of*

*# miss match in array Dimension.*

*c = a + b*

*print(c)*

**3. What makes Python better than other libraries for umerical computation?**

***The following are some libraries makes python better than other for numerical computation:***

### *1|* [*SciPy (Scientific Numeric Library)*](https://www.scipy.org/)

*Officially released in 2000-01, SciPy is free and open source library used for scientific computing and technical computing. The library consists of modules for optimisation,*

*image processing, FFT, special functions and signal processing.*

*The SciPy package includes algorithms and functions which are the crux of Python scientific computing capabilities. The sub-package includes:*

* *io: used for the standard input and output*
* *lib: this function is used to wrap python external libraries*
* *signal: used for processing signal tools*
* *optimise: used to optimise algorithms which include linear programming.*
* *fftpack: this subpackage helps for the discretion Fourier to transform algorithms*

### *2|* [*Pandas (Data Analytics Library)*](https://pandas.pydata.org/)

*Pandas is the most important data analysis library of Python. Being open source, it is used for analysing data with Python. It can take data formats of CSV or TSV files, or a SQL database and convert it into Python data frames with rows and columns which is similar to tables in statistical formats.*

*The advantage of using Pandas is :*

* *df.max()-to get the highest value of a column*
* *df.min()-to get the minimum value of a column*
* *df.std()-to get the standard deviation of each column.*
* *df.count()-to get the number of non-null values in each column of the data frame.*

### *4|* [*Numeric Python (Fundamental Numeric Package)*](http://www.numpy.org/)

*Better known as Numpy, numeric Python has developed a module for Python, Numpy assures accurate calculations with matrices and arrays.*

*We need to import Numpy into memory to perform numerical operations.*

* *Import numpy as np (to import Numpy into memory)*
* *A\_values=[20,30,40,50] (defining a list)*
* *A=np.array(A\_values) (to convert list into one dimensional numpy array)*
* *print(A) (to get one dimensional array displayed)*
* *print(A\*9/5 +32) (to turn values in the list into degrees fahrenheit)*

*Not only these but aslo so many libraries makes python better.*

**4. How does NumPy deal with files?**

## ***Specifying the file path***

*Let’s look at how we can specify the path of the file from which we want to read data.*

*We’ll use a sample text file for our code examples, which lists the weights (in kg) and heights (in cm) of 100 individuals, each on a row.*

*I will use various variants in this file for explaining different features of the loadtxt function.*

*Let’s begin with the simplest representation of the data in a text file. We have 100 lines (or rows) of data in our text file, each of which comprises two floating-point numbers separated by a space.*

*The first number on each row represents the weight, and the second number represents the height of an individual.*

*Here’s a little glimpse from the file:*

*110.90 146.03*  
*44.83 211.82*  
*97.13 209.30*  
*105.64 164.21*

*This file is stored as `*[*weight\_height\_1.txt*](https://drive.google.com/open?id=1jzNfeyS37z5xTNbfTfdPnQfKU1o9EBDY)*`.*  
*Our task is to read the file and parse the data in a way that we can represent in a NumPy array.*  
*We’ll import the NumPy package and call the loadtxt method, passing the file path as the value to the first parameter filePath.*

**5. Mention the importance of NumPy.empty().**

*numpy.empty() in Python. The numpy module of Python provides a function called numpy.empty(). This function is used to create an array without initializing the entries of given shape and type. Just like numpy.zeros(), the numpy.empty() function doesn't set the array values to zero, and it is quite faster than the numpy.zeros().*