Overview of the NumPy arrays and its features

- Creating arrays and initializing
- Reading arrays from files
- Special initializing functions
- Slicing and indexing
- reshaping arrays
- Numpy Maths
- Combining arrays
- Basic algebraic operations using numpy arrays
  - Solving linear equations
  - Matix inversions
  - Calculating eigen vectors

Exercise on NumPy

- Importing numpy library
- Creating one dimensional numpy arrays and initializing
- Numpy Arrays - dtypes & shapes
- Creating two dimensional numpy arrays and initializing
- Reading an array from a file
- Special Initializing functions
- Slicing & Indexing an array
- Reshaping an array
- Updating an array
- Numpy Maths - Adding, substracting, multiplying, tranposing arrays
- Calculating column sums and row sums
- Combine arrays
- Linear Algebra.. Advanced Matrix Operation - Solving a set of linear equations
- Matrix Inversion - Calculating an eigen value and vector for a matrix

Numpy Arrays

- Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.
- Numpy arrays are great alternatives to Python Lists. Some of the key advantages of Numpy arrays are that they are fast, easy to work with, and give users the opportunity to perform calculations across entire arrays.
- A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.
- :

Numpy Arrays

There are 5 general mechanisms for creating arrays:

- Conversion from other Python structures (e.g., lists, tuples)
- Intrinsic numpy array array creation objects (e.g., arange, ones, zeros, etc.)
- Reading arrays from disk, either from standard or custom formats
- Creating arrays from raw bytes through the use of strings or buffers
- Use of special library functions (e.g., random)

Numpy – N-Dimesional Arrays

- An ndarray is a (usually fixed-size) multidimensional container of items of the same type and size. The number of dimensions and items in an array is defined by its shape, which is a tuple of *N* positive integers that specify the sizes of each dimension. The type of items in the array is specified by a separate *data-type object (dtype)*, one of which is associated with each ndarray.
- As with other container objects in Python, the contents of an ndarray can be accessed and modified by *indexing or slicing* the array (using, for example, *N* integers), and via the methods and attributes of the ndarray.
- Different ndarrays can share the same data, so that changes made in one ndarray may be visible in another. That is, an ndarray can be a *"view"* to another ndarray, and the data it is referring to is taken care of by the *"base"* ndarray. ndarrays can also be views to memory owned by Python strings or objects implementing the buffer or *array* interfaces.

Thank you