NUMpy & Matplotlib

Prepared by
- Prof. Prachi Pancholi
- Ganpat University

Contents:

- Introduction
- Ndarray obj
- Data types
- Array attributes
- Creation of array
- Array from existing data
- Array from numerical ranges
- Slicing and Indexing
- Advance Indexing
- Broadcasting
- Iterating over array
- Array manipulation
- Binary operations , String functions, Mathematical functions, Arithmetic operations, Statistical functions, Search /Sort and Counting functions
- Linear algebra

- NumPy,---- Numerical Python
- A library consisting of multidimensional array objects and a collection of routines for processing those arrays
- **Numeric--** the ancestor of NumPy,
- Operations using NumPy:
    - Mathematical and logical operations on arrays.
    - Fourier transforms and routines for shape manipulation.
    - Operations related to linear algebra.

    (NumPy has in-built functions for linear algebra and random number generation.)

NumPy – A Replacement for MatLab

- NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library)
- It is open source
- supports a much greater variety of numerical types than Python

Installation of Numpy

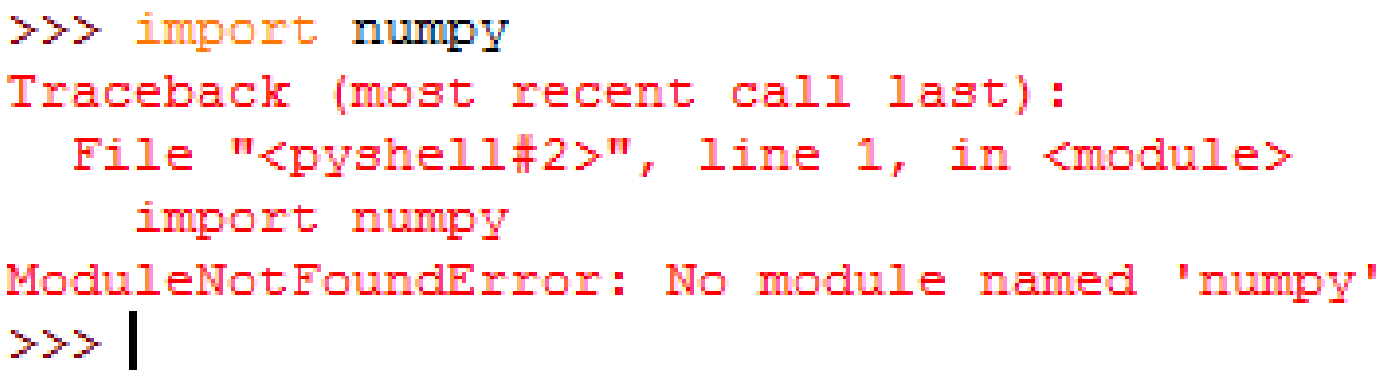
- pip install numpy
- Windows:
  - Anaconda(free Python distribution for SciPy stack)
  - Canopy(free as well as commercial distribution with full SciPy stack for Windows, Linux and Mac.)
  - Python(It is a free Python distribution with SciPy stack and Spyder IDE for Windows OS)
- Linux
  - sudo apt-get install python-numpy
  - python-scipy python-matplotlibipythonipythonnotebook python-pandas
  - python-sympy python-nose

- To test whether NumPy module is properly installed, try to import it from Python prompt.

>>>import numpy

Alternatively, >>> import numpy as np

If it is not installed, the following error message will be displayed:



```
>>> import numpy
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    import numpy
ModuleNotFoundError: No module named 'numpy'
>>> 
```

NumPy - Ndarray Object

- important object defined in NumPy is an N-dimensional array type called **ndarray**
- collection of items of the same type +accessed using a zero-based index
- Each element in ndarray is an object of data-type object (called **dtype**).
- The basic ndarray is created using an array function in NumPy as follows −

```
>>>numpy.array
```

Example: numpyarray.py

Data Type Objects (dtype)

- NumPy numerical types are instances of dtype (data-type) objects, each having unique characteristics
- A data type object describes interpretation of fixed block of memory corresponding to an array, depending on the following aspects −
    - Type of data (integer, float or Python object)
    - Size of data
    - Byte order (little-endian or big-endian)
    - If data type is a subarray, its shape and data type
    - In case of structured type, the names of fields, data type of each field and part of the memory block taken by each field.

- A structured data type called student with a string field 'name', an integer field 'age' and a float field 'marks'. This dtype is applied to ndarray object:

```
>>>import numpy as np
>>>student = np.dtype([('name','S20'), ('age', 'i1'), ('marks', 'f4')])
>>>a = np.array([('abc', 21, 50),('xyz', 18, 75)], dtype = student)
>>> print a
```

ndarray.shape:

- This array attribute returns a tuple consisting of array dimensions. It can also be used to resize the array.

reshape():NumPy also provides a reshape function to resize an array.

ndarray.ndim:

- This array attribute returns the number of array dimensions.

numpy.itemsize

- This array attribute returns the length of each element of array in bytes.

numpy.empty

- It creates an uninitialized array of specified shape and dtype. It uses the following constructor −

numpy.empty(shape, dtype = float, order = 'C')

numpy.zeros

- Returns a new array of specified size, filled with zeros

numpy.ones/eye

- Returns a new array of specified size and type, filled with ones.

numpy.as array

- Similar to numpy.array, used for creating an array from existing data.

Arrays from numerical ranges

- numpy.arange
  - This function returns an ndarray object containing evenly spaced values within a given range. The format of the function is as follows −

  >>>numpy.arange(start, stop, step, dtype)

- numpy.linspace
  - This function is similar to **arange()** function. In this function, instead of step

  size, the number of evenly spaced values between the interval is specified

  >>>numpy.linspace(start, stop, num, endpoint, retstep, dtype)

  (i.e **starting** point,**ending** point,evenly spaced samples(default 50), endpointset to true or false , retstep returns samples alongwith step)

- numpy.logspace
  - This function returns an ndarray object that contains the numbers that are evenly spaced on a log scale. Start and stop endpoints of the scale are indices of the base, usually 10.

```
>>>numpy.logspace(start, stop, num, endpoint, base, dtype)
```

Numpy- Indexing & Slicing

- follows zero-based index
- 3 types of indexing methods − **field access, basic slicing** and **advanced indexing**.
- Basic slicing--- extension of Python's basic concept of slicing to n dimensions.
- Slicing can also include ellipsis (…) to make a selection tuple of the same length as the dimension of an array. If ellipsis is used at the row position, it will return an ndarray comprising of items in rows.
- Two types of advanced indexing − **Integer** and **Boolean**.

Advanced indexing – **Integer** and **Boolean**.

- Integer Indexing
  - Selects any arbitrary item in an array based on its Ndimensional index.
  - Each integer array represents the number of indexes into that dimension.
  - Advanced and basic indexing can be combined by using one slice (:) or ellipsis (…) with an index array.
- Boolean Array Indexing
  - Used when the resultant object is meant to be the result of Boolean operations, such as comparison operators.
  - NaN (Not a Number) elements are omitted by using ~ (complement operator)

Difference:

- Advanced indexing always returns a copy of the data. As against this, the slicing only presents a view.

broadcasting refers to the ability of NumPy to treat arrays of different shapes during arithmetic operations.

- operations on arrays **of non-similar shapes** is still possible in NumPy, because of the broadcasting capability. The smaller array is **broadcast** to the size of the larger array so that they have compatible shapes.
- A set of arrays is said to be **broadcastable** if :
    - Arrays have exactly the same shape.
    - Arrays have the same number of dimensions and the length of each dimension is either a common length or 1

Iterating over array

- **numpy.nditer**.
  - It is an efficient multidimensional iterator object using which it is possible to iterate over an array. Each element of an array is visited using Python’s standard Iterator interface.
- **Broadcasting Iteration:**
  - If two arrays are **broadcastable**, a combined **nditer** object is able to iterate upon them concurrently. Eg. an array **a** has dimension 3X4, and there is another array **b** of dimension 1X4, the iterator of following type is used (array **b** is broadcast to size of **a**).

Array manipulation

- Shape----reshape, flat, flatten, ravel
- Operation/s----transpose, ndarray.T , swapaxes
- Changing Dimensions----broadcast, expand
- Joining arrays
- Splitting arrays---split,vsplit,hsplit
- Add/remove element/s in array ---resize, delete, insert

Array manipulation(w.r.t Shape)

- Reshape
- Flat

```
>>>a = np.arange(8).reshape(2,4)
>>>print (a.flat[5])
```

- Flattened

```
>>>print (a.flatten())
>>>print (a.flatten(order = 'F'))
```

- Ravel

```
>>>print a.ravel(order = 'F')
```

Binary operators

- Bitwise_and
- Bitwise_or
- Invert
- Left-shift
- Right_shift

String Functions

arrays of dtype numpy.string_ or numpy.unicode_

- Add
- Multiply
- Center
- Capitalize/ Title
- Lower/ Upper /Split

Mathematical Functions

- Trignometric functions
- Arithmetic operations :add(), subtract(), multiply(), and divide()
- Rounding function---ceil / floor/ around
- Statistical functions: median/ mean/ amin / amax / percentile /ptp

** amin /amax returns min and max values across axis(0 indicates column; 1 indicates row)

** ptp : returns a range (i.e. max-min value along axis)

Sort, Search & Counting Functions

- numpy.sort(): returns a sorted copy of the input array.

  numpy.sort(a, axis, kind, order): kind is default ‘quicksort’

3 ways: sort()/argsort()/lexsort()

- numpy.nonzero(): returns the indices of non-zero elements in the input array.
- numpy.where(): returns the indices of elements in an input array where the given condition is satisfied.
- numpy.extract():returns the elements satisfying any condition.

Difference: sort()/ argsort()/lexsort()

- **a.sort()**
  (i) Sorts the array in-place & returns None
  (ii) Return type is None

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

Print(np.sort(a))   or a.sort()

- **np.argsort(a)**
  (i) Returns the indices that would sort an array
  (ii) Return type is numpy array
  (iii) Occupies space as a new array of sorted indices is returned

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

b = np.argsort(a) --------→[2 1 5 4 6 3 0]

- **np.lexsort((b, a))**
  (i) Perform an indirect sort using a sequence of keys
  (ii) Sort by a, then by b
  (iii) Return type ndarray of ints Array of indices that sort the keys along the specified axis
  (iv) Occupies space as a new array of sorted indices pair wise is returned.

```
              0   1   2  3   4 5 6
a = np.array([9, 3, 1, 3, 4, 3, 6]) # First column –     [1,3,3,3,4,6,9]
b = np.array([4, 6, 9, 2, 1, 8, 7]) # Second column   [9,2,6,8,1,7,4]
ind = np.lexsort((b, a)) # Sort by a then by b
 [2 3 1 5 4 6 0]
```

Argmax(): Returns indices of the max element of the array in a particular axis.

- array =np.random.randint(16, size=(4, 4))
- print("\nIndices of Max element : ", np.argmax(array, axis=0)) →[1 3 0 0]
- print("\nIndices of Max element : ", np.argmax(array, axis=1)) →[3 0 1 1]

```
                        ELEMENT   INDEX
->[[ 0  3  8 13]          13        3
 ->[12 11  2 11]          12       0
 ->[ 5 13  8  3]          13        1
 ->[12 15  3  4]]         15        1
```

Byte swapping and other matrix functions

- numpy.ndarray.byteswap(): toggles between the two representations
- **matlib.empty()** : returns a new matrix without initializing the entries
- **numpy.matlib.zeros():**returns the matrix filled with zeros.
- **numpy.matlib.ones():** returns the matrix filled with 1s.
- **numpy.matlib.eye():** returns a matrix with 1 along the diagonal elements and the zeros elsewhere

numpy.matlib.eye(n, M,k, dtype)

- **numpy.matlib.identity()** : returns the Identity matrix of the given size
- **numpy.matlib.rand()** :returns a matrix of the given size filled with random values

** matlib module returns matrices instead of ndarray

Linear Algebra

- Dot()---2 arrays
- Vdot()—2 vectors(the dot product of the two vectors. If the first argument is complex, then its conjugate is used for calculation.)
- Inner()
- Determinant()
- Solve()
- Inv()
- Matmul()

MATPLOTLIB
(plotting library for Python)

- open source alternative for MatLab.
- used with graphics toolkits like PyQt and wxPython.
- **pyplot()** :to plot 2D data.
- The graphical representation is displayed by **show()** function
- Add an argument indicating line style of graph :
- ‘-’/ ‘—’ / ‘ -.’ / ‘ : ‘ /, ‘o’ / , ‘ ; ‘ etc.
- color abbreviations are also defined:
- 'b‘(Blue) /'g ‘(Green) /'r‘(Red) / ‘k’(Black) etc.

Subplot

- plot different things in the same figure
- Firstly define range for x and y axis
- Use pyplot.subplot(rows, columns quadrant)
- Use pyplot.plot(x,y)
- Pyplot.show()

**Remember : Whenever subplot is applied the quadrant is formed on the basis of rows and columns divisions made.

Eg. pyplot.subplot(1, 2 ,1) will plot the graph in 1st quadrant(left side of framing window) .this subpotting will have atmost 2 quadrants only.