Ch2 : Beginning with NumPy Fundamentals

December 28, 2013

[1]

1 ndarray

- multi-dimensional array object called ndarray

- consists of 1) The actual data 2) Some metadata describing the data

- many operations change only the metadata

- arange is 1D array

- ∵ NumPy array is homogeneous (items must be same type)

- ∴ easy to determine storage size for array

- indexing like Python

- ndarry has two attributes:

- dtype: NumPy datatypes represented by special objects [2]

- shape [3] The shape attribute of the array is a tuple, contains the length in each dimension.

- the `array` function takes in any object that is array-like (e.g. Python lists) then create an array from the object.[4]

2 Data Types

- NumPy support much more data types than Python library ∴ program can be optimized (least memory) to specific numerical usage, also ∵ complex numbers

- There is conversion methods from each type to another [5]
  ex) In: float64(42)
  !Out:42.0!

- Many functions have a data type argument, which is often optional. [6]

---

[1]NumPy 1.5 Beginner Guide by Ivan Idris
[2]myobject.dtype(): function that returns the datatype of the objects
[3]myobject.shape(): returns the shape of the object
[4]NumPy function tends to have many various optional args with predefined defaults.
[5]Exception: TypeError if try to convert ComplexNumber into an integer, or a float
[6]ex) for array ∵ only one data type ∴ autodetect type

- Character codes are included for backward compatibility with Numeric.

- Their use is not recommended, should instead use dtype objects.

- dtype constructor: create data types, can take character code or just its general name

- A listing of all full data type names can be found in sctypeDict.keys()

- dtype attribute:

    - d.char : return the character code of dtype object
    - d.type: attribute corresponds to the type of object of the array elements
    - d.str : string representation of the data type.
      `'<endianness-optional> <character-code><number-of-bytes-each-array-item-requires>'`
      Endianness = the way bytes are ordered within a 32 or 64-bit word. In big-endian order, the most significant byte is stored first. In little-endian order, the least significant byte is stored first.