ISE 314X
Computer Programing for Engineers

NumPy Quick Start Tutorial

Yong Wang
Assistant Professor
Systems Science & Industrial Engineering
Binghamton University

Objectives

- To understand multi-dimensional arrays
- To be able to create and manipulate these arrays

The Basics

- NumPy is the fundamental package for scientific computing
- It is a Python library that provides fast operations on multi-dimensional arrays

The Basics

- Such operations include mathematical, logical, shape manipulation, as well as sorting, selecting, linear algebra, statistics, and simulation

The Basics

- At the core of the NumPy package, is the `ndarray` object
- An `ndarray` object is a homogeneous multi-dimensional array

Array Creation

- Create an array from a list

```
>>> import numpy as np
>>> a = np.array([0, 1, 2, 3])
>>> a
array([0, 1, 2, 3])
>>> type(a)
<class 'numpy.ndarray'>
```

Array Creation

- Create an array from a tuple

```
>>> import numpy as np
>>> a = np.array((0, 1, 2, 3))
>>> a
array([0, 1, 2, 3])
```

Array Creation

```
>>> import numpy as np
>>> a = np.array(0, 1, 2, 3)
???
```

Array Creation

```
>>> import numpy as np
>>> b = np.array([[0, 1], [2, 3]])
>>> b
array([[0, 1],
       [2, 3]])
```

Array Creation

```
>>> import numpy as np
>>> c = np.array([3,'4.5'])
>>> c
array(['3', '4.5'],
      dtype='<U11')
```

Array Creation

```
>>> import numpy as np
>>> c = np.array([3,'4.5',[1,-2]])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: setting an array element with a sequence
```

How to modify it and make it a legal statement?

Array Creation

- `arange` returns arrays instead of lists

```
>>> import numpy as np
>>> c = np.arange(10)
>>> c
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Array Creation

```
>>> import numpy as np
>>> np.arange(0, 2, 0.3) #interval is 0.3
array([0., 0.3, 0.6, 0.9, 1.2, 1.5, 1.8])
```

- Different from `range`:

```
>>> range(0, 2, 0.3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'float' object cannot be interpreted
as an integer
```

Array Creation

- Created arrays with a specific shape

```
>>> import numpy as np
>>> c = np.arange(9).reshape(3, 3)
>>> c
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

Array Creation

- `linspace` creates a specific number of elements between two values

```
>>> np.linspace(0, 2, 5) #5 numbers from 0 to 2
array([ 0. ,  0.5,  1. ,  1.5,  2. ])
```

Array Creation

```
>>> np.zeros((2,3))
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>> np.ones((2,3))
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
>>> help(np.ones)
```

Array Creation

- The parameter can be a single number, a list, or a tuple

```
>>> np.zeros(3)
array([ 0.,  0.,  0.])
>>> np.zeros([2,3])
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>> np.zeros((2,3))
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```

Array Creation

```
>>> np.zeros(2,3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: data type not understood
```

Array Creation

- Create a 2D unit array

```
>>> u = np.eye(2)
>>> u
array([[ 1.,  0.],
       [ 0.,  1.]])
```

Array Creation

- Create a 2D diagonal array

```
>>> a = np.array([1, 2, 3, 4])
>>> d = np.diag(a)
>>> d
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
```

Array Attributes

```
>>> c = np.arange(9).reshape(3, 3)
>>> c
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
>>> c.ndim  #number of dimensions
2
>>> c.shape #length of each dimension
(3, 3)
>>> c.size  #total number of elements
9
```

Array Reshaping

```
>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> a
array([[1, 2, 3],
       [4, 5, 6]])
>>> a.ravel()   #Flattening
array([1, 2, 3, 4, 5, 6])
>>> a.reshape((3, -1))  #-1 means unspecified
array([[1, 2],
       [3, 4],
       [5, 6]])
```

- Reshaping is not done in-place

Basic Operations

- Arithmetic operators on arrays apply elementwise

```
>>> a = np.array([20, 30, 40, 50])
>>> b = np.arange(4)
>>> b
array([0, 1, 2, 3])
>>> c = a - b
>>> c
array([20, 29, 38, 47])
>>> b ** 2
array([0, 1, 4, 9])
>>> 10 * np.sin(a)
array([9.1294, -9.8803, 7.4511, -2.6237])
>>> a <= 35
array([True, True, False, False], dtype=bool)
```