Introductory Scientific Computing with Python

Numpy arrays

FOSSEE

Department of Aerospace Engineering
IIT Bombay

Mumbai, India

Outline

1. `numpy` arrays
   - Slicing arrays
   - Array creation
   - Example: plotting data from file

Outline

1. `numpy` arrays
   - Slicing arrays
   - Array creation
   - Example: plotting data from file

The numpy module

- Efficient, powerful array type
- Abstracts out standard operations on arrays
- Convenience functions
- **`ipython --pylab`** imports part of numpy

Without Pylab

```
In []: from numpy import *
In []: x = linspace(0, 1)
```

Note that we had done this “import” earlier!

```
# Can also do this:
In []: import numpy
In []: x = numpy.linspace(0, 1)
# or
In []: import numpy as np
In []: x = np.linspace(0, 1)
```

Note the use of `numpy.linspace`

`numpy` arrays

- Fixed size (**`arr.size`**)
- Same type (**`arr.dtype`**)
- Arbitrary dimensionality: **`arr.shape`**
- **`shape`**: extent (size) along each dimension
- **`arr.itemsize`**: number of bytes per element
- Note: **`shape`** can change so long as the **`size`** is constant
- Indices start from 0
- Negative indices work like lists

numpy arrays

```
In []: a = array([1,2,3,4])
In []: b = array([2,3,4,5])

In []: print(a[0], a[-1])
(1, 4)

In []: a[0] = -1
In []: a[0] = 1
```

Operations are elementwise

Simple operations

```
In []: a + b
Out[]: array([3, 5, 7, 9])
In []: a*b
Out[]: array([2, 6, 12, 20])
In []: a/b
Out[]: array([0, 0, 0, 0])
```

- Operations are element-wise
- Types matter

10 m

Data type matters

Try again with this:

```
In []: a = array([1.,2,3,4])
In []: a/b
```

Examples

pi and **e** are defined.

```
In []: x = linspace(0.0, 10.0, 200)
In []: x *= 2*pi/10
# apply functions to array.
In []: y = sin(x)
In []: y = cos(x)
In []: x[0] = -1
In []: print(x[0], x[-1])
(-1.0, 10.0)
```

size, shape, rank etc.

```
In []: x = array([1., 2, 3, 4])
In []: size(x)
Out[]: 4
In []: x.dtype
dtype('float64')
In []: x.shape
Out[] (4,)
In []: rank(x)
Out[]: 1
In []: x.itemsize
Out[]: 8
```

Multi-dimensional arrays

```
In []: a = array([[ 0,  1,  2,  3],
   ...:             [10,11,12,13]])
In []: a.shape # (rows, columns)
Out[]: (2, 4)

In []: a[1,3]
Out[]: 13

In []: a[1,3] = -1
In []: a[1] # The second row
array([10,11,12,-1])
In []: a[1] = 0 # Entire row to zero.
```

Outline

1. `numpy` arrays
   - Slicing arrays
   - Array creation
   - Example: plotting data from file

Slicing arrays

```
In []: a = array([[1,2,3], [4,5,6],
   ...:            [7,8,9]])
In []: a[0,1:3]
Out[]: array([2, 3])

In []: a[1:,1:]
Out[]: array([[5, 6],
              [8, 9]])

In []: a[:,2]
Out[]: array([3, 6, 9])
```

Slicing arrays

```
In []: a = array([[1,2,3], [4,5,6],
   ...:            [7,8,9]])
In []: a[0,1:3]
Out[]: array([2, 3])

In []: a[1:,1:]
Out[]: array([[5, 6],
              [8, 9]])

In []: a[:,2]
Out[]: array([3, 6, 9])
```

Slicing arrays

```
In []: a = array([[1,2,3], [4,5,6],
   ...:            [7,8,9]])
In []: a[0,1:3]
Out[]: array([2, 3])

In []: a[1:,1:]
Out[]: array([[5, 6],
              [8, 9]])

In []: a[:,2]
Out[]: array([3, 6, 9])
```

Slicing arrays

```
In []: a = array([[1,2,3], [4,5,6],
   ...:            [7,8,9]])
In []: a[0,1:3]
Out[]: array([2, 3])

In []: a[1:,1:]
Out[]: array([[5, 6],
              [8, 9]])

In []: a[:,2]
Out[]: array([3, 6, 9])
```

Slicing arrays ...

```
In []: a = array([[1,2,3], [4,5,6],
   ...:           [7,8,9]])

In []: a[0::2,0::2] # Striding...
Out[]: array([[1, 3],
              [7, 9]])
# Slices refer to the same memory!
```

Slicing arrays ...

```
In []: a = array([[1,2,3], [4,5,6],
   ...:                 [7,8,9]])

In []: a[0::2,0::2] # Striding...
Out[]: array([[1, 3],
              [7, 9]])
# Slices refer to the same memory!
```

Outline

Array creation functions

- `array(object)`
- `linspace(start, stop, num=50)`
- `ones(shape)`
- `zeros((d1,...,dn))`
- `empty((d1,...,dn))`
- `identity(n)`
- `ones_like(x)`, `zeros_like(x)`, `empty_like(x)`

May pass an optional **`dtype=`** keyword argument
For more dtypes see: **`numpy.typeDict`**

Creation examples

```
In []: a = array([1,2,3], dtype=float)
In []: ones_like(a)
Out[]: array([ 1.,  1.,  1.])

In []: ones( (2, 3) )
Out[]: array([[ 1.,  1.,  1.],
              [ 1.,  1.,  1.]])

In []: identity(3)
Out[]: array([[ 1.,  0.,  0.],
              [ 0.,  1.,  0.],
              [ 0.,  0.,  1.]])
```

35 m

Array math

- Basic elementwise math (given two arrays **a, b**):
  - **a + b** → **add(a, b)**
  - **a - b**, → **subtract(a, b)**
  - **a * b**, → **multiply(a, b)**
  - **a / b**, → **divide(a, b)**
  - **a % b**, → **remainder(a, b)**
  - **a ** b**, → **power(a, b)**
- Inplace operators: **a += b**, or **add(a, b, a)**
  What happens if **a** is **int** and **b** is **float?**

Array math

- Logical operations: `==`, `!=`, `<`, `>`, etc.
- `sin(x)`, `arcsin(x)`, `sinh(x)`, `exp(x)`, `sqrt(x)` etc.
- `sum(x, axis=0)`, `product(x, axis=0)`
- `dot(a, b)`

Convenience functions: `loadtxt`

- `loadtxt(file_name)`: loads a text file
- `loadtxt(file_name, unpack=True)`: loads a text file and unpacks columns

```
In []: x = loadtxt('pendulum.txt')
In []: x.shape
Out[]: (90, 2)

In []: x, y = loadtxt('pendulum.txt',
   ...:                unpack=True)
In []: x.shape
Out[]: (90,)
```

45 m

Advanced

- Only scratched the surface of `numpy`
- **`reduce`**, **`outer`**
- Typecasting
- More functions: **`take`**, **`choose`**, **`where`**, **`compress`**, **`concatenate`**
- Array broadcasting and **`None`**
- Record arrays

Learn more

- https://docs.scipy.org/doc/numpy-dev/user/quickstart.html
- http://numpy.org

Recap

- Basic concepts: creation, access, operations
- 1D, multi-dimensional
- Slicing
- Array creation, dtypes
- Math
- `loadtxt`

50 m

Outline

1. numpy arrays
   - Slicing arrays
   - Array creation
   - Example: plotting data from file

Example: plotting data from file

Data is usually present in a file!

Lets look at the **pendulum.txt** file.

```
In []: cat pendulum.txt
1.0000e-01 6.9004e-01
1.1000e-01 6.9497e-01
1.2000e-01 7.4252e-01
1.3000e-01 7.5360e-01

...
```

Reading `pendulum.txt`

- File contains L vs. T values
- First Column - L values
- Second Column - T values
- Let us generate a plot from the data file

Gotcha and an aside

Ensure you are in the same directory as
pendulum.txt
if not, do the following on IPython:

```
In []: %cd directory_containing_file
# Check if pendulum.txt is there.
In []: ls
# Also try
In []: !ls
```

Note: **%cd** is an IPython magic command. For more information do:

```
In []: ?
In []: %cd?
```

Exercise

- Plot L versus T square with dots
- No line connecting points

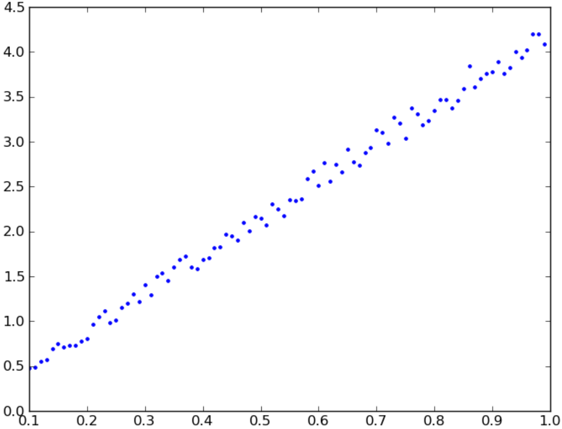
60 m

Solution

```
In []: L, t = loadtxt('pendulum.txt',
 ....:                 unpack=True)
In []: plot(L, t*t, '.')
```

or

```
In []: x = loadtxt('pendulum.txt')
In []: L, t = x[:,0], x[:,1]
In []: plot(L, t*t, '.')
```


4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

Odds and ends

```
In []: mean(L)
Out[]: 0.54499999999999993

In []: std(L)
Out[]: 0.25979158313283879
```

Summary

- Introduction to `numpy` arrays
- Slicing arrays
- Multi-dimensional arrays
- Array operations
- Creating arrays
- Loading data from file

65 m