Scientific Computing with Python numpy + matplotlib

Eliot Feibush

PICSciE
Princeton Institute for Computational Science and Engineering
Review

Integers

Floating Point

Dynamic Typing – no declarations

\[ x = 5 \]
\[ y = 6.3 \]

Names start with a letter, cAsE SeNsiTiVe. Long names OK.
Review Character Strings

Dynamic typing – no declaration
No memory allocation
Immutable

s = "Good Afternoon"

len(s) # length of string
Review String Slicing

s = “Good Afternoon”

s[0] evaluates to “G”

s[5:10] selects “After”  # string slicing

s[:10] selects “Good After”

s[5:] selects “Afternoon”

s[-4:] selects “noon”  # last 4 characters
String Methods

String is a Class with data & subroutines:

t = s.upper()
pos = s.find("A")

_____________________________________

first = "George"
last = "Washington"
name = first + " " + last

# string concatenation
Review Lists

Ordered sequence of items
  Can be floats, ints, strings, Lists

a = [16, 25.3, “hello”, 45]
a[0] contains 16
a[-1] contains 45
a[0:2] is a list containing [16, 25.3]
Create a List

days = []
days.append("Monday")
days.append("Tuesday")

years = range(2000, 2014)
List Methods

List is a Class with data & subroutines:

d.insert(index, object)

d.remove(value)

d.sort() # sort in place

Can concatenate lists with +
String split

s = "Princeton Plasma Physics Lab"

myList = s.split()  # returns a list of strings

print myList
   [ "Princeton", "Plasma", "Physics", "Lab" ]

help(str.split)    # delimiters, etc.
**Tuple**

Designated by ( ) parenthesis

A List that can not be changed. Immutable. No append.

Good for returning multiple values from a subroutine function.

Can extract slices.
Review math module

import math
dir(math)

math.sqrt(x)
math.sin(x)
math.cos(x)

from math import *
dir()

sqrt(x)

from math import pi
dir()

print pi
import a module

import math  # knows where to find it

import sys
sys.path.append("/u/efeibush/python")
import cubic.py  # import your own code

if task == 3:
    import math  # imports can be anywhere
Looping with the range() function

for i in range(10):  # i gets 0 - 9

range() is limited to integers

numpy provides a range of floats
Summary

Integer, Float
String
List
Tuple

def function

Keywords:  if  elif  else
            while  for in
            import  print

Indenting counts  :
Run python as Interpreter

type()
dir()
help()
Programming = Problem Solving

Take a large problem and break it into a series of small problems.

Write an outline of the steps.

Relate each step to lines of code.
numpy module

ndarray class

Items are all the same type.

Contiguous data storage in memory of items.

Considerably faster than lists.

Class with data and methods (subroutines).
import numpy

dir()
dir(numpy)
help(numpy)
help(numpy.ndarray)  # class
help(numpy.array)    # built-in function
numpy module

import numpy
dir(numpy)
help(numpy.zeros)

```python
da = numpy.zeros((3,5))
#
create 3 rows, 5 columns
[  [ 0., 0., 0., 0., 0. ],
  [ 0., 0., 0., 0., 0. ],
  [ 0., 0., 0., 0., 0. ]
 ]
# default type is float64
numpy Array Access

Access order corresponding to printed order:

[row] [column] index starts with 0

a[0][2] = 5

[ [ 0., 0., 5., 0., 0. ],
  [ 0., 0., 0., 0., 0. ],
  [ 0., 0., 0., 0., 0. ],
  [ 0., 0., 0., 0., 0. ] ]
idle

Integrated Development Environment (IDE)
Color-coded syntax
statement completion
debugger

Written in Python using tkinter GUI module
idle IDE

```python
import numpy

a = numpy.zeros((2,4))
print a
```

Can save text in interpreter window to a file.

control-p  control-n  to  recall commands
Programming Exercise Prep

Mac: Editing source code

Textedit

Preferences

Format: Plain text

Open and Save

Uncheck: Add .txt extension

Save: File Format – Plain Text
Mac: Run python from command line

Spotlight terminal

$ python myprogram.py
Anaconda: “The Distro”

Create your work environments

conda create -n proj1 python=2.7  # ~/proj1

conda create -p /home/efeibush/work/proj1 python=2.7

conda install package_name

pip install package_name

source activate /home/efeibush/proj1  # bash
Array Index Exercise

Write a python program:

Create an array (6, 3)

Set each element to rowIndex + columnIndex

print the array

```
edit index.py

python index.py
```

[ [ 0.  1.  2. ]  
[ 1.  2.  3. ]  
[ 2.  3.  4. ]  
[ 3.  4.  5. ]  
[ 4.  5.  6. ]  
[ 5.  6.  7. ] ]
1. Create Array

```python
a = numpy.linspace(start, stop, nPoints, inclusive)
# array of evenly spaced floats
# begins with start
# ends with stop
# can include/exclude stop   True/False

example: 0., 2.5, 101
         0., 2.5, 100, False
```

**Useful to make “range” of floats**

```python
for i in a:
```

ndarray has `__iter__()`

Arrays are iterable
1a. Create Array

```python
alog = numpy.logspace(startExp, maxExp, nSteps)
```

Example: 2., 10., 9
2. Create Array

\[
b = \text{numpy.array}([2., 4., 6.])\\
\text{# 1-D from list}
\]

\[
\text{# range(start, end, incr) returns a list so}\\
b = \text{numpy.array}(\text{range}(10))\\
\text{array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])}
\]

\[
b = \text{numpy.array}((2., 4., 6.))\\
\text{# 1-D from tuple}
\]
Rotation Matrix Exercise

Write a python program:

Create a 2 x 2 rotation matrix, 30 degrees:

\[
\begin{bmatrix}
\cos(30) & \sin(30) \\
-sin(30) & \cos(30)
\end{bmatrix}
\]

\[\text{radians} = \text{degrees} \times \pi / 180.\]
Add to your python program:

Create 18 xy points around unit circle
(18, 2) array

\[
\begin{align*}
x &= \text{cosine}(\text{angle}) \\
y &= \text{sine}(\text{angle})
\end{align*}
\]

`print a.round(3)`
Pointer vs. Deep Copy

```
a = numpy.zeros((3, 3))
b = a       # b is a pointer to a
c = a.copy()  # c is a new array
```

```
b is a      # True
c is a      # False
```

Views
base
Array Arithmetic

a = numpy.array( range(10, 20) )
a + 5
a - 3
a * 5
a / 3.14
a.sum()  a.min()  a.max()
a > 15
   (a > 15).sum()
Array Arithmetic by Index

```python
a = numpy.array( range(10) )
b = numpy.array( range(0, 1000, 100) )

a + b  # a[0] + b[0], a[1] + b[1] ...
a - b
a * b  # not row, column matrix product
a / b
```

The 2 arrays must be the same shape.
Row, Column Matrix Product

c = numpy.dot(a, b)

Dot product of 2 arrays.
Matrix multiplication for 2D arrays.
Transform Exercise

Add to your python program:
Transform 18 points by the rotation matrix.
Save in new array.
Scale up by factor of 2.

$$\begin{bmatrix} 18x2 \end{bmatrix} \cdot [2 \times 2] \cdot [2 \times 2] \cdot \begin{bmatrix} 18x2 \end{bmatrix}.$$
Cross Product

$$z_A = \texttt{numpy.cross}(x_A, y_A)$$

Note: we have been using \texttt{numpy}. functions
Array Shape

```python
import numpy as np

# Create a 1D array
a = np.linspace(2, 32, 16)

# Reshape the array to a 4x4 matrix
a = a.reshape(4, 4)  # ndarray method

# Get the shape of the array
print(a.shape)  # ndarray attribute tuple (4, 4)

# Create a 2x8 array
a = np.linspace(2, 32, 16).reshape(8, 2)

# Flatten the 2D array into a 1D array
a.flatten()  # return a 1-D array
```

Methods have () Attributes do not.
Array Diagonals

\[ a = \text{numpy.linspace}(1, 64, 64) \]

\[ a = a.\text{reshape}(8, 8) \]

\[ \text{numpy.triu(a)} \quad \# \text{ upper triangle} \]

\[ \text{numpy.tril(a)} \quad \# \text{ lower triangle} \]

\[ \text{numpy.diag(a)} \quad \# \text{ main diagonal} \]

\[ \text{numpy.diag(a, 1)} \quad \# 1 \text{ above} \]

\[ \text{numpy.diag(a, -1)} \quad \# 1 \text{ below} \]
numpy.array Order [row] [column] vs. Internal Storage Order

C is default, Fortran can be specified [contiguous] []

c = numpy.zeros( (2,4), dtype=numpy.int8)
f = numpy.zeros( (2,4), dtype=numpy.int8, order="F")

# show c.flags  f.flags

c[0][1] = 5    # show c.data[:]
f[0][1] = 5    # show f.data[:]

numpy.array [][] access is the same regardless of internal storage order
Interpreter
Look at array flags
dir(a.flags)

Program
status = a.flags.c_contiguous
status = a.flags.f_contiguous
# boolean True or False

ndarray.flatten()  # ‘F’ or ‘C’ (default)
Array Data Types

numpy.float64 is the default type

float32
int8, int16, int32, int64, uint8, uint16, uint32, uint64
complex64, complex128
bool - True or False

a.dtype shows type of data in array

>>> help(numpy.ndarray)  # Parameters Attributes
Multi-Dimensional Indexing

```python
a = numpy.array( range(12) )
a = a.reshape(2,6)  # 2 rows, 6 columns

a[1][5] contains 11

a[1, 5] is equivalent, more efficient
```
1. Array Slicing

```python
a = numpy.array(range(0, 100, 10))
Array([ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90])
```

```python
a[2:4] contains 20, 30
```

```python
a[-4 : -1] contains 60, 70, 80
```

Slicing returns ndarray
2. Array Slicing

```python
a = numpy.array(range(64)).reshape(8,8)

a[3, 4] contains 28

asub = a[3:5, 4:6]

Very useful for looking at data & debugging.

a[:,2]  # all rows, column 2
a[3, 2:5]  # row 3, columns 2 and 3 and 4
```
Array Stuff

a.T
a.min()
a.max()
a.round()
a.var() - variance
a.std() – standard deviation
Organize Arrays

Make a list of arrays named a, b, and c:

```python
w = [ a, b, c ]
```

```python
len(w)  # length of list is 3
```

```python
w[1].max()  # use array method
```
numpy Tutorial

wiki.scipy.org/Tentative_Numpy_Tutorial

docs.scipy.org/doc/numpy/reference/routines.html

numpy for Matlab Users

wiki.scipy.org/NumPy_for_Matlab_Users
1. Plotting

matplotlib – designed to look like MATLAB plot

200 subroutines for various plots.

Generally available with Python

matplotlib.org
gallery
Plotting on raleigh.princeton.edu

$ module load python/2.7
$ ipython --pylab

Bring up plot windows as separate threads, no blocking. Draw commands are displayed sequentially.

ipython --pylab --classic --logfile mytype.txt
dash dash pylab

import matplotlib.pyplot as plt
plt.plot( range(10), range(10) )
Plotting on raleigh.princeton.edu

$ ipython --pylab -i plot2.py

or

$ ipython --pylab

>>> import plot2

# runs plot2.py from current directory

man ipython --no-confirm-exit
Plot Exercise

New python program:

Create a numpy array of ten $X$ values.
Create a numpy array of ten $Y$ values.

```python
import matplotlib.pyplot as g
g.plot(x, y)
g.show()  # optional
```
Add to your python program:
Slice both (18, 2) arrays into:
  x array
  y array

```python
g.plot(ax, ay)
g.plot(bx, by)
```
matplotlib Contour Plot

```python
r = numpy.random.rand(10,10)

g.contour(r)  # contour line plot

fig2 = g.figure()  # start new window
fig2.canvas.manager.window.Move((648,20))

g.contourf(r)  # filled contour plot

clf()  # clear figure, erase
cla()  # clear axis
```
Review Defining a Function

Block of code separate from main.
Define the function before calling it.

def myAdd(a, b):    # define before calling
    return a + b

p = 25             # main section of code
q = 30
r = myAdd(p, q)
Keyword Arguments

Provide default values for optional arguments.

def setLineAttributes(color="black", style="solid", thickness=1):
    ...

# Call function from main program
setLineAttributes(style="dotted")
setLineAttributes("red", thickness=2)
String Escape Sequence

s = ‘\n’  # interpreted as ASCII Linefeed

t = r’\n’  # raw string, t contains \n
matplotlib LaTeX

```python
import matplotlib.pyplot as plt
plt.rc("text", usetex=True)
    # set config to draw text with Tex
plt.xlabel( r"\textbf{Time}" )
    # draw x label "Time" in bold font
    # compare to:  plt.xlabel("Time")
latex2.py example
```
Figure 1

\[ y = \sin \frac{1}{x^2} \]
matplotlib \ f(x,y,t) \\
raleigh.astro.princeton.edu \\
/u/efeibush/python/netcdf \\
readsteps.py \\

Display time steps from psiRZ
More Info & Resources

docs.scipy.org

princeton.edu/~efeibush/python/numpy

Princeton University Python Community

princetonpy.com