Python Numpy Programming

Eliot Feibush
Zach Kaplan  Bum Shik Kim
Princeton Plasma Physics Laboratory

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:

\[
t = s.\text{upper}()
pos = s.\text{find}("A")
\]

\[\text{first} = "George"
\text{last} = "Washington"
\text{name} = \text{first} + " " + \text{last}
\]

# 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( )
d.remove( )
d.sort( )

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
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
Review Defining a Function

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

```python
def myAdd(a, b):
    return a + b

p = 25
q = 30
r = myAdd(p, q)
```

# define before calling

# main section of code
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)
Looping with the range() function

```python
for i in range(10):  # i gets 0 - 9
    #
    # 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()
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).
numpy module

ndarray class

import numpy

dir()
dir(numpy)
help(numpy)
help(numpy.ndarray)  # class
help(numpy.array)    # built-in function
import numpy
dir(numpy)
help(numpy.zeros)
a = 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

Can save text in interpreter window to a file.

control-p  control-n to recall commands

```python
import numpy

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

```
[[ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]]
```
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
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
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:
```
1a. Create Array

alog = numpy.logspace(start, maxExp, nSteps)

Example: 0., 10., 11
2. Create Array

```python
b = numpy.array([ 2., 4., 6. ])  # 1-D from list

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

b = numpy.array(( 2., 4., 6. ))  # 1-D from tuple
```
Write a python program:

Create a 2 x 2 rotation matrix, 30 degrees:

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

radians = degrees * pi / 180.
Add to your python program:
Create 18 xy points around unit circle
(18, 2) array

\[ x = \cos(\text{angle}) \]
\[ y = \sin(\text{angle}) \]

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

\[
a = \text{numpy.zeros}( (3, 3) )
b = a \quad \# b \text{ is a pointer to } a
c = a.\text{copy}() \quad \# c \text{ is a new array}
\]

\[
b \text{ is a} \quad \# \text{ True}
c \text{ is a} \quad \# \text{ False}
\]

Views
base
Array Arithmetic

```python
import numpy as np

a = np.array(range(10, 20))

a + 5
a - 3
a * 5
a / 3.14
a.sum()
(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.
Add to your python program:
Transform 18 points by the rotation matrix.
Save in new array.
Scale up by factor of 2.

\[
\begin{bmatrix}
18x2
\end{bmatrix}
\begin{bmatrix}
2x2
\end{bmatrix}
\begin{bmatrix}
2x2
\end{bmatrix}
\begin{bmatrix}
18x2
\end{bmatrix}
\]
Cross Product

\[ z_A = \text{numpy.cross}(x_A, y_A) \]

Note: we have been using *numpy.* functions
Array Shape

```python
a = numpy.linspace(2, 32, 16)

a = a.reshape(4, 4)  # ndarray method

a.shape  # ndarray attribute  tuple (4, 4)

a = numpy.linspace(2, 32, 16).reshape(8, 2)
```
Array Diagonals

```python
import numpy as np

# Create a 64-element array
a = np.linspace(1, 64, 64)

# Reshape the array to 8x8
a = a.reshape(8, 8)

# Extract the upper triangle
np.triu(a)  # upper triangle

# Extract the lower triangle
np.tril(a)  # lower triangle

# Extract the main diagonal
np.diag(a)  # main diagonal

# Extract 1 element above the main diagonal
np.diag(a, 1)  # 1 above

# Extract 1 element below the main diagonal
np.diag(a, -1)  # 1 below
```
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
ndarray.flags

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

da[1][5] contains 11

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

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

a[2:4] contains 20, 30

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()
a.var()
a.std()
Organize Arrays

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

```python
w = [a, b, c]
len(w) # length of list is 3
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 nobel.princeton.edu

> ipython27 -pylab

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

import myplot
reload(myplot)
dir(myplot)

ipython27 --pylab --classic --logfile mytype.txt
dash dash dash pylab
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()
```
Plot Circles Exercise

Add to your python program:
Slice both (18, 2) arrays into:

x array

y array

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

```
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
```
matplotlib LaTeX

```python
import matplotlib.pyplot as plt
plt.rc("text", usetex=True)
plt.xlabel( r"\textbf{Time}"

# plt.xlabel("Time")
```

latex.py example
$y = \sin \frac{1}{x^2}$
Python at princeton.edu

ssh nobel.princeton.edu
compton% which python
/usr/bin/python
version 2.6.6

/usr/bin/python2.7
version 2.7.3
More Info & Resources

docs.scipy.org

princeton.edu/~efeibush/python/numpy

Princeton University Python Community
princetonpy.com
Where to?

Graphing & visualization
Writing new classes
scipy – algorithms & math tools
   Image Processing
Visualization toolkit – python scripting
Eclipse IDE
Multiprocessing
Python → GPU, CUDA
Reading a netCDF File

Popular file format for scientific data
Multi-dimensional arrays

scipy – netcdf_file class for read/write
numpy – n-dimensional data arrays
Read a Text File

gFile = open("myfile.txt", "r")

for j in gFile:  # python magic: text file iterates on lines
    print j  # print each line

gFile.close()
Write a Text File

```python
f = open("myfile.txt", "w")
a = 1
b = 2
f.write("Here is line " + str(a) + "\n");
f.write("Next is line " + str(b) + "\n");
f.close()
```
import sys
print sys.argv

sys.argv is a list
sys.argv[0] has the name of the python file
Subsequent locations have command line args

>>> help(sys)
Command Line Scripts

Upgrade to csh or bash shell scripts
shell agnostic

Much better text handling
Process control - popen()
Shell Scripting

```python
import os

fileL = []  # set up a list

for f in os.listdir("."):  
  if f.endswith(".py"):  
    print f 
    fileL.append(f)

fileL.sort()  # list function, sort in place

print fileL
```

```csh
#!/bin/csh

foreach file (*.py) 
  echo $file
end
```
Python + GUI

tkinter
pyqt
wxpython