Introduction to Scientific Computing in Python

Numpy, Scipy and Matplotlib packages

Oscar Dalmau

Centro de Investigación en Matemáticas
CIMAT A.C. Mexico

September 5, 2020

Outline

Overview

- Numpy: provides powerful numerical arrays objects, and routines to manipulate them
- Matplotlib: 2-D visualization
- Mayavi: 3-D visualization
- Scipy: high-level data processing routines. Optimization, regression, interpolation, etc

Creating arrays

- Manual construction of array 1D:

```
>>> import numpy as np
>>> a = np.array([1.,21.,3.])
>>> a.ndim
>>> a.shape
>>> a.size
>>> a.max()
>>> a.argmax()
```

Creating arrays

- Manual construction of array 2D:

```
>>> import numpy as np
>>> a = np.array([[1.,21.,3.], [2,44,6]])
>>> a.ndim
>>> a.shape
>>> a.size
>>> a.max()
>>> a.argmax()
```

Creating arrays

- Manual construction of array 3D:

```
>>> import numpy as np
>>> a = np.array([[[1.,21.,3.], [2,44,6]],
      [[1.,2.,1.], [20,4,2]]])
>>> a.ndim
>>> a.shape
>>> a.size
>>> a.max()
>>> a.argmax()
```

Creating arrays

- Functions for creating arrays:

```
>>>a=np.arange(10) # 0 .. n-1
>>>b=np.arange(1,9,2)#start,end(exclusive),step
>>>c=np.linspace(0,1,6)#start,end,num-points
>>>d=np.linspace(0,1,5,endpoint=False)
>>>e=np.ones((3,3))#(3, 3) is a tuple
>>>f=np.zeros((3,3))
>>>g=np.eye(3)
>>>h=np.diag(np.array([1,2,3,4]))
>>>i=np.random.rand(4,3)#uniform in [0, 1]
>>>j=np.random.randn(2,5)#Gaussian
>>>np.random.seed(1234)#Setting the random seed
```

Basic data types

- bool: bool, bool8, bool_
- Integers: byte, short, int8, int16, int 32, int64, ...
- Unsigned integers: ubyte, ushort, uint8, uint16, uint 32, uint64, ...
- Floating: single, double, float16, float32, float64, float96, float128
- Complex Floating: csingle, complex64, complex128, complex192, complex256

Basic data types

```
>>> import numpy as np
>>> a = np.array([[1.,21.,3.], [2,44,6]],
          dtype=int8)
>>> b = np.array([[1.,21.,3.], [2,44,6]],
          uint64)

>>> a.ndim
>>> a.shape
>>> a.dtype
>>> b.dtype
```

Indexing, slicing and assignment 1D

The items of an array can be accessed and assigned to the same way as other Python sequences (e.g. lists):

- One dimensional arrays:

```
>>>a=np.arange(10)
>>>a[0]
>>>a[-1]  # a[len(a)-1]
>>>a[::2]
>>>a[::-1]
>>>a[0]=10
>>>a[0:2]=[12,16]
```

deleting and inserting 1D

The items of an array can be accessed and assigned to the same way as other Python sequences (e.g. lists):

- One dimensional arrays:

```
>>>a[0]=[]#Error!!!
>>>a=np.delete(a,0)#delete the first element
>>>a=np.delete(a,-1)#delete the last element
>>>a=np.insert(a,0,5)#array,index,value
   # insert the 'value' before the 'index'
```

Indexing and slicing ND

The items of an array can be accessed and assigned to the same way as other Python sequences (e.g. lists):

- Multi dimensional arrays:

```
>>>a=(10*np.random.rand(5,5)).astype(int)
>>>a[0,0]
>>>a[-1,-1]
>>>a[:,1]
>>>a[::2,::2]
>>>a[0:2,1:4]
>>>a[1:4,3:5]#a[1:4,:][:,3:5]
>>>a[np.ix_([1,3,4],[0,2])]
```

Assignment ND

The items of an array can be accessed and assigned to the same way as other Python sequences (e.g. lists):

- Multi dimensional arrays:

```
>>>a[0,0]*=10
>>>a[1:4,3:5]=100*np.ones((3,2))#a[1:4,3:5]=100
>>>a[np.ix_([1,3],[0,4])]=np.random.rand((3,2))
>>>(a>4).choose(a,4)
>>>a.clip(min=None, max=4)
>>>a.clip(min=3, max=7)
>>>a[:]=3
```

Numpy Functions

Deleting and inserting in ND

- Multi dimensional arrays:

```
>>>np.delete(a,0,0)#arr,obj(int),axis
>>>np.delete(a,np.s_[::2],0)#arr,obj(slice),axis
>>>np.delete(a,np.arange(0,4,2),1)#obj(array)
>>>np.delete(a,[1,3])#np.delete(a,[1,3],None)
>>>np.delete(a,[1,3],0)
>>>np.insert(a,1,100,1)#arr,obj(int,slice,seq),
                       #value,axis
>>>np.insert(a,(1,4),100,1)
>>>np.insert(a,1,[1,2,3,4,5],1)
>>>ia=np.zeros((5,2))
>>>np.insert(a,(1,4),ia,1)
```

Numpy Functions

- Multi dimensional arrays:

```
>>>a.take([0,2,2], axis=0).take([2,4], axis=1)
>>>a.diagonal(offset=0)#a.diagonal(0)
>>>a.sum(axis=0)#a.sum(0)
>>>a.sum(axis=1)#a.sum(1)
>>>a.sum()
>>>a.trace(offset=0)#a.trace(0)
>>>a.max()# a.max(0), a.max(1)
>>>a.T #a.transpose()
```

- Statistical functions: mean, median, std, var, correlate, cov

Numpy Functions

- Convert ND array into a vector (column or row):

```
>>>c=a.flatten()#convert into an 1D array
>>>c=a.flatten()[:,np.newaxis]#column vector
>>>c=np.array([2,3,4,5]).reshape(-1,1)#column ve
>>>c=np.r_['c',1:10]#column vector,(matrix obj)
>>>(c.T*c)[0,0]#inner product
>>>c*c.T#outer product
```

Numpy Functions

- Concatenation

```
>>>a=np.array([[0, 1, 2], [3, 4, 5]])
>>>c=np.concatenate((a,a),0)#np.vstack((a,a))
>>>c=np.concatenate((a,a),1)#np.hstack((a,a))
>>>c=np.r_['0', a,a]
>>>c=np.r_['1', a,a]
```

Vector and matrix mathematics

- NumPy provides many functions for performing standard vector and matrix multiplication routines

```
>>>a=np.array([1, 2, 3], float)
>>>b=np.array([0, 1, 1], float)
>>>np.dot(a, b)
>>>np.inner(a,b)
>>>np.outer(a,b)
```

Vector and matrix mathematics

- Linear Algebra

```
>>>a=np.array([[4, 2, 0], [9, 3, 7], [1, 2, 1]],
>>>b=np.linalg.inv(a)
>>>np.dot(a,b)#np.matrix(a)*np.matrix(b)
>>>U,D,V=np.linalg.svd(a)
>>>np.dot(np.dot(U,np.diag(D)),V)#===a
>>>np.matrix(U)*np.matrix(np.diag(D))*\
   np.matrix(V)#===a
```

Commonly-used numpy.linalg functions

Function	Description
diag	Return the diagonal (or off-diagonal) elements
dot	Matrix multiplication
trace	Compute the sum of the diagonal elements
det	Compute the matrix determinant
eig	Compute the eigenvalues and eigenvectors of a square matrix
inv	Compute the inverse of a square matrix
pinv	Compute the Moore-Penrose pseudo-inverse
qr	Compute the QR decomposition
svd	Compute the singular value decomposition
solve	Solve the linear system Ax = b for x
lstsq	Compute the least-squares solution to y = Xb, See exampleleastsquare.py

Boolean or ‘mask’ index arrays

- Boolean index array (result 1D array)

```
>>>y = np.arange(35).reshape(5,7)
>>>y[y>10] # the result is an ndarray of 1D
```

- Boolean index array (result 2D array)

```
>>>y = np.arange(35).reshape(5,7)
>>>row1=np.array(np.arange(5))
>>>y[row1>2,:] # select rows with index > 2
>>>row2=np.array(['Bob','Joe','Will',
      'Bob','Will'])
>>>y[row2=='Will',:]
```

Universal Functions: Fast Element-wise Array Functions

- A universal function, or ufunc, is a function that performs elementwise operations on data in ndarrays. You can think of them as fast vectorized wrappers for simple functions that take one or more scalar values and produce one or more scalar results.
- Examples of universal function: abs, fabs, sqrt, square, exp, log, log10, log2, sign, ceil, floor, modf, isnan, isfinite, isinf, cos, cosh, sin, sinh, tan, tanh, arccos, arccosh, arcsin, arcsinh, arctan, arctanh

```
>>>a = np.array([1,2,3])
>>>np.sqrt(a)
```

Matplolib

- Matplotlib is 2D desktop plotting package. The project was started by John Hunter in 2002 to enable a MATLAB-like plotting interface in Python.
- Now, matplotlib has a number of add-on toolkits, such as mplot3d for 3D plots
- Matplotlib is (very) well documented, see http://matplotlib.org/index.html

First Example

```
>>>import numpy as np
>>>import matplotlib.pyplot as plt
>>>x=np.linspace(-1,1,100);y=x**2
>>>plt.plot(x,y)
>>>plt.show(False)
```

First Example

```
>>>x=np.linspace(-1,1,100);y=x**2
>>>plt.plot(x,y)
>>>plt.show(False)
```

First Example.

Adding graphics in the same figure

```
x=np.linspace(-pi,pi,100);
ys,yc,y2=np.sin(x),np.cos(x),x**2
plt.plot(x,ys)
plt.plot(x,yc)
plt.plot(x,y2)
```

Or simply

```
x=np.linspace(-pi, pi, 100)
ys,yc,y2 = np.sin(x),np.cos(x),x**2
plt.plot(x,ys,x,yc,x,y2)
```

First Example.

Adding style

```
#num,figsize,dpi
plt.figure(1,figsize=(8,6), dpi=100)
x=np.linspace(-pi,pi,100);
plt.plot(x,np.cos(x),color="blue",linewidth=1.0,
linestyle="-")
#Set x, y  limits
plt.xlim(-4.0,4.0); plt.ylim(-1.0,1.0)
#Set x, y ticks
plt.xticks(linspace(-4,4,9)),
plt.yticks(linspace(-1,1,5))
#Save figure using 80 dots per inch
plt.savefig("exercice1.png",dpi=80)
```

First Example.

MATLAB-style

```
#num,figsize,dpi
plt.figure(1,figsize=(8,6), dpi=100)
x=np.linspace(-pi,pi,100);
s,c=np.sin(x),np.cos(x)
plt.plot(x,s,'b-',x,c,'g-')
#Set x, y  limits
plt.xlim(-4.0,4.0); plt.ylim(-1.0,1.0)
#Set x, y ticks
plt.xticks(linspace(-4,4,9))
plt.yticks(linspace(-1,1,5))
#Save figure using 80 dots per inch
np.savefig("exercice1.png",dpi=80)
```

First Example.

Adding information: title, labels, legend

```
plt.title('Cos and Sin Functions')
plt.xlabel('Eje X')
plt.xlabel('Eje Y')
plt.grid()
plt.legend(lnp.oc='upper left')
plt.xticks( [-pi,-pi/2,0,pi/2,pi])
plt.yticks([-1, 0, +1])
plt.xticks([-pi,-pi/2, 0,pi/2,pi],
       [r'$-\pi$',r'$-\pi/2$',r'$0$',r'$+\pi/2$',
       r'$+\pi$'])
plt.yticks([-1,0,+1],
       [r'$-1$',r'$0$',r'$+1$'])
```

Subplot

- Subplots allow us to arrange plots in a regular grid. One needs to specify the number of rows and columns and the number of the plot.
- Axes are very similar to subplots but allow placement of plots at any location in the figure. So if we want to put a smaller plot inside a bigger one we do so with axes.

Example Subplot

```
#num,figsize,dpi
plt.figure(1,figsize=(8,6), dpi=100)
x=np.linspace(-pi,pi,100);
s,c,t,y2=np.sin(x),np.cos(x),np.tan(x),x**2
plt.subplot(2,2,1);plt.plot(x,s,'r-')
plt.subplot(2,2,2);plt.plt.plot(x,c,'b-')
plt.subplot(2,2,3);plt.plot(x,t,'g-')
plt.subplot(2,2,4);plt.plot(x,y2,'k-')
```

Some Pyplot functions

figure	xlim	plot	stem
savefig	xticks	loglog	step
subplot	ylabel	semilogx	pie
gca	ylim	semilogy	contour
gcf	yscale	hist	contourf
grid	yticks	hist2d	quiver
title	annotate	bar	imshow
axes	arrow	barbs	spy
axis	Circle	barh	pcolor
legend	Polygon	boxplot	cm
xlabel	Rectangle	scatter	ginput

Matplotlib 3D

```
import matplotlib.pyplot as plt
import numpy as np
grd=np.linspace(-1,1,20)
x,y=np.meshgrid(grd,grd)
z=x**2+y**2
plt.contour(x,y,z)
plt.contourf(x,y,z)
plt.pcolor(x,y,z)
plt.pcolor(x,y,z, cmap=plt.cm.hot)
plt.imshow(z)
```

Scipy

scipy package contains the following subpackages:

- optimize: routines for optimization
- misc: Various utilities that dont have another home (e.g. lena, imfilter, imresize, imrotate, imread, imshow, imsave, etc).
- ndimage: Multidimensional image processing (e.g., filters, interpolation, measurements, io).
- signal: Signal processing (N-dimensional arrays, e.g. convolution, correlation, filtering, etc.).
- fftpack: Discrete Fourier transforms.
- ...