numpy and matplotlib

Jeffrey Salmond

July 3, 2017

Importing numpy

The first thing to do is always to import numpy

```
import numpy as np
```

it is a very common convention to import numpy to the name np

you can also do

```
from numpy import *
```

to call functions without the `np.` prefix

check the numpy version

```
np.__version__
```

Create an array

to create an array

```
np.array([1,2,3]) # array from a list

np.ones(3)  # array full of ones
np.zeros(2) # array full of zeros
```

and in two dimensions

```
np.array([[1,2],[3,4]]) # array from a list

np.ones([3,3])  # array full of ones
np.zeros([2,2]) # array full of zeros
```

Create an array

```
xs = np.linspace(0, 10, 5)
# xs = array([0.0,2.5,5.0,7.5,10.0])

ys = np.arange(0,10)
# ys = array([0,1,2,3,4,5,6,7,8,9])

z2 = np.random.random(3)
# z2 = random numbers in array of 3
z3 = np.random.random([3,4])
# z3 = random numbers in array of 3*4
```

Query an array

```
a = np.zeros([3,4])
np.ndim(a)  # 2
np.shape(a) # [3,4]
np.size(a)  # 12

a = np.random,random([200,200,200])
np.ndim(a)  # 3
np.shape(a) # [200,200,200]
np.size(a)  # 8000000
```

can also access properties of the array object

```
a.ndim, a.shape, a.size
```

Indexing

```
a = np.linspace(0,10,5)
# a = array([0.0,2.5,5.0,7.5,10.0])
a[2] # 5.0

a[2] = 99.0
# a = array([0.0,2.5,99.0,7.5,10.0])
```

negative indices count from the end

```
a[-1] # 10.0
```

Indexing in 2D (slicing)

```
b = np.array([[1,2,3],
              [4,5,6],
              [7,8,9]])
b[1,1] # 5

b[:,1] # array([2,5,8])
b[1,:] # array([4,5,6])
b[1]   # array([4,5,6])

b[:,1] = np.array([12,15,18])
# b = np.array([[1,12,3],
#               [4,15,6]
#               [7,18,9]])
```

Functions on arrays

```
np.max(x)  # maximum value of x
np.min(x)  # minimum value of x
np.mean(x) # mean value of x
```

element-wise arithmetic operations are also easy

```
x = np.array([1,2,3])
y = np.array([3,2,1])
x + y # array([4,4,4])
```

these functions can be broadcasted

```
x = np.array([1,2,3])
x + 10 # array([11,12,13])
```

Masking arrays

```
x = np.array([1,2,3,4,5])
x > 3 # np.array([False, False, False, True, True])
```

the array full of booleans is known as a mask array

can use the mask array to select

```
x[x>3] # array([4,5])
```

or set elements

```
x[x>3] = 0
# x = array([1,2,3,0,0])
```

Importing matplotlib

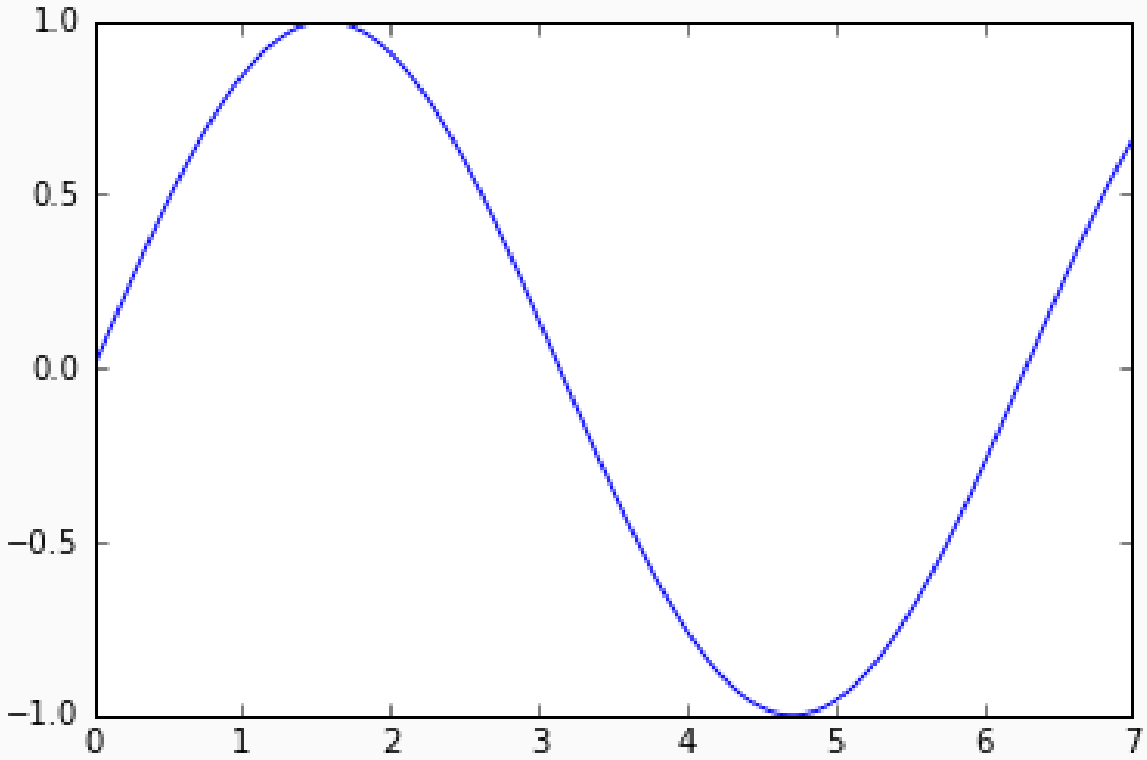
```
import matplotlib.pyplot as plt
```

to make plots appear in a jupyter notebook add

```
%matplotlib inline
```

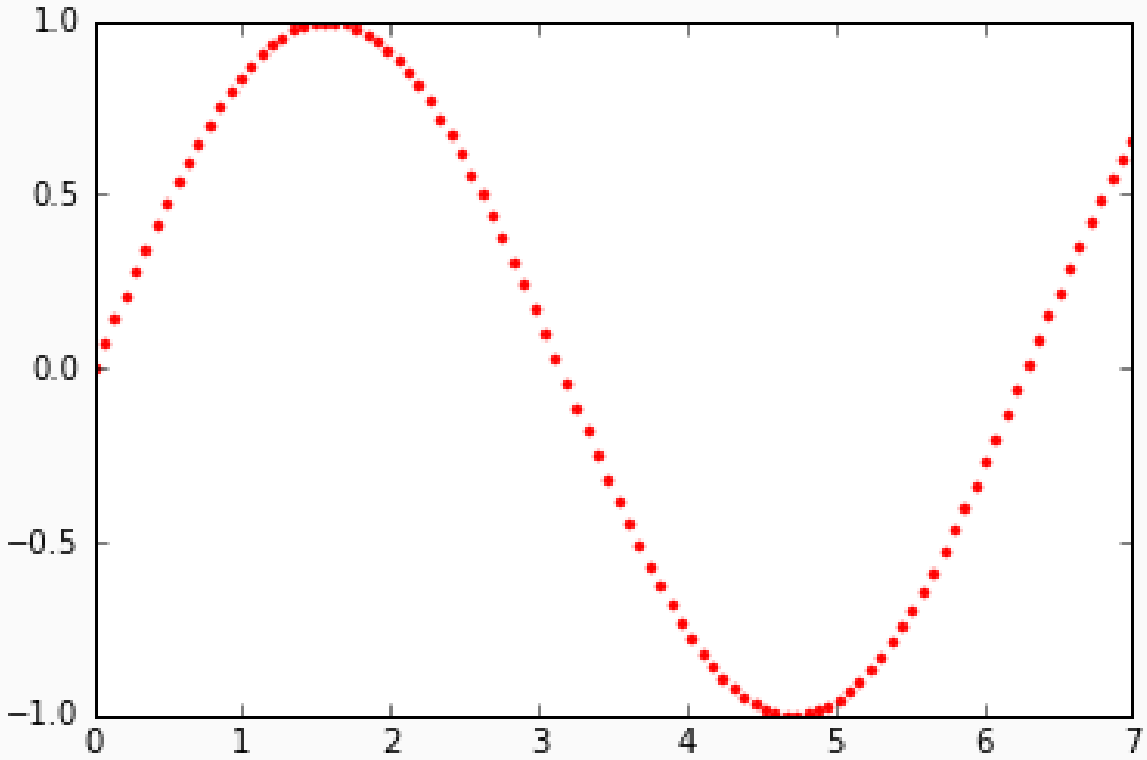
2D plots

```
x = np.linspace(0, 7, 100)
y = np.sin(x)
plt.plot(x, y)
```



2D plots

```
x = np.linspace(0, 7, 100)
y = np.sin(x)
plt.plot(x, y, '.r')
```



matplotlib documentation

This is only skimming the the surface of what plots are possible!

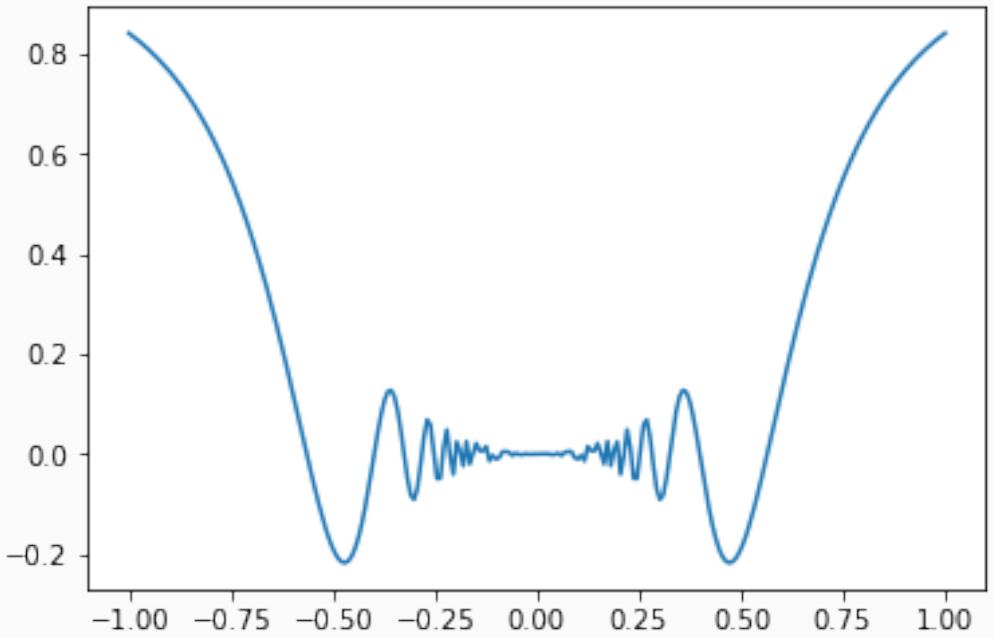
go to

- `matplotlib.org/2.0.2/gallery.html`
- `matplotlib.org/2.0.2/examples`

to see what is possible

Exersise: Some 2D plots

plot $y = x^2 sin(1/x^2)$ between $x = -1$ and $x = 1$



add $y = x^2 cos(1/x^2)$ to the plot

Exersise: Plotting an image

get some image data

```
from scipy import misc
face = misc.face(gray)
```

which can be plotted with

```
plt.imshow(face)
```

- investigate the structure of the 'face' array
- plot in black and white by calculating the average of the three colors and plotting with

  ```
  plt.imshow(face_bw, cmap=plt.cm.gray)
  ```
- zoom in on the face by selecting some of the pixels
- blur the image by averaging neighbouring pixels

Exersise: The Mandelbrot set

the mandelbrot set is calculated by iterating the function

$$f_c(z) = z * z + c$$

to see whether for a given complex number c the recurrence relationship diverges

```
def mandelbrot(z, maxiter):
    c = z
    for n in range(maxiter):
        if abs(z) > 2:
            return n
        z = z*z + c
    return maxiter
```

we plot the number of steps taken for the function to diverge

Exersise: The Mandelbrot set

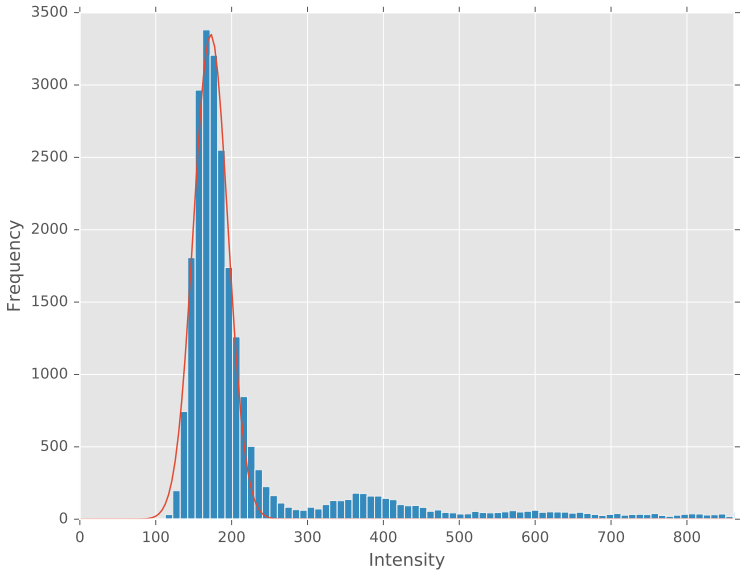
```
def mandelbrot(z, maxiter):
    c = z
    for n in range(maxiter):
        if abs(z) > 2:
            return n
        z = z*z + c
    return maxiter
```

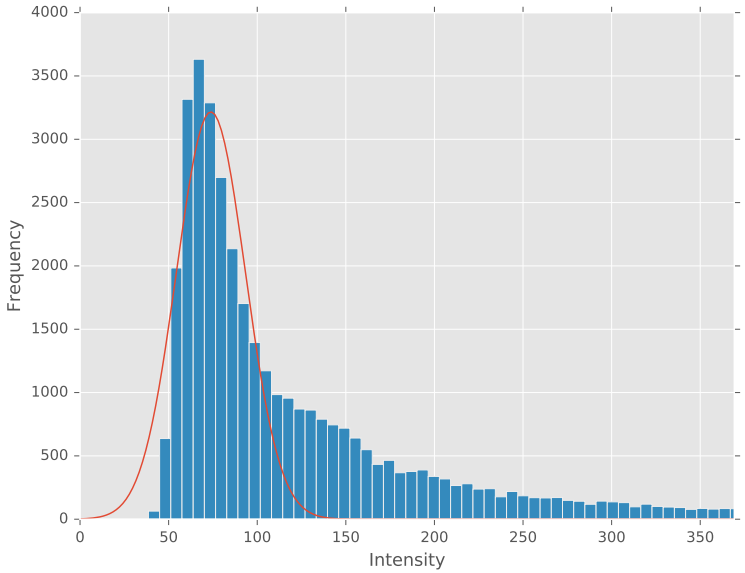
tasks

- plot the mandelbrot function!
- use numpy to speed up the calculation of the divergence time

Projects I've used python for!

atmospheric data analysis





Projects I’ve used python for!

Fluid dynamics