
06-Solutions

Stephen Pascoe

March 17, 2014

1 Solutions

```
In [1]: import numpy as np
from pylab import *
from matplotlib import pyplot as plt
```

1.1 Solution 1.1 : Starting Matplotlib

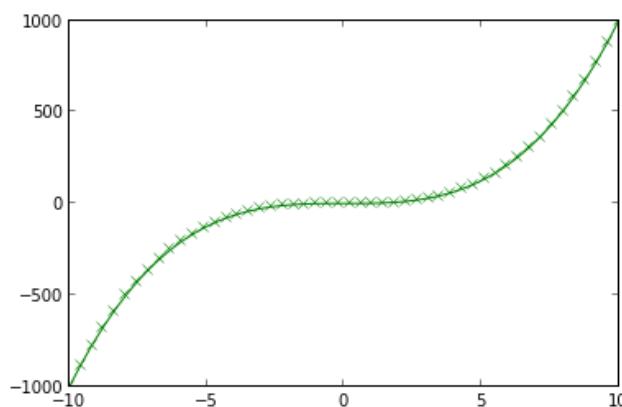
Part 1

See presented material.

Part 2

```
In [2]: x = np.linspace(-10, 10)
y = x**3
plot(x, y, 'gx-')
```

```
Out [2]: [
```

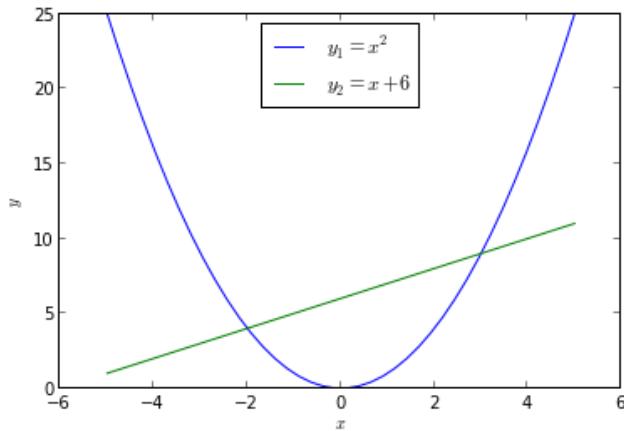


1.2 Solution 1.2 : Plot layout and saving

Part 1

```
In [3]: from matplotlib import pyplot as plt

x = np.linspace(-5, 5, 40)
y = x**2
fig, axis = plt.subplots(1)
axis.plot(x, y, label='$y_1 = x^2$')
axis.plot(x, x + 6, label='$y_2 = x + 6$')
axis.legend(loc='upper center')
axis.set_xlabel('$x$')
axis.set_ylabel('$y$')
fig.savefig('data/ex_1.2.pdf')
```



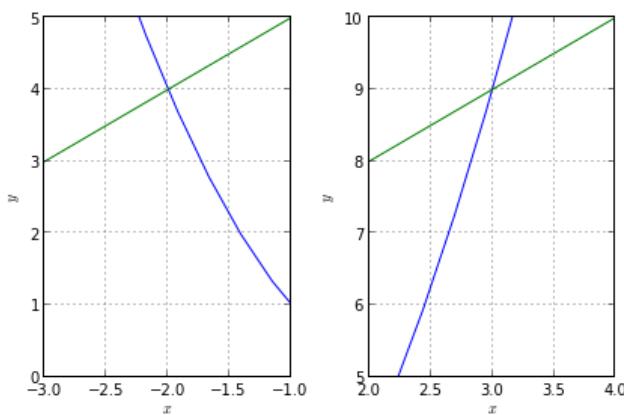
Part 2

```
In [4]: fig, axes = plt.subplots(1, 2)

# [(xlim, ylim), ...]
regions = [((-3, -1), (0, 5)),
           ((2, 4), (5, 10))]

for axis, (xlim, ylim) in zip(axes, regions):
    axis.plot(x, y)
    axis.plot(x, x + 6)
    axis.set_xlim(xlim)
    axis.set_ylim(ylim)
    axis.set_xlabel('$x$')
    axis.set_ylabel('$y$')
    axis.grid()

fig.tight_layout()
```



The plots indicate intersections at $x = -2$ and $x = 3$, consistent with $y = (x + 2)(x - 3) = x^2 - x - 6$.

1.3 Solution 1.3 : Plotting 2D data

```
In [5]: # From the Exercise
from scipy.stats import norm

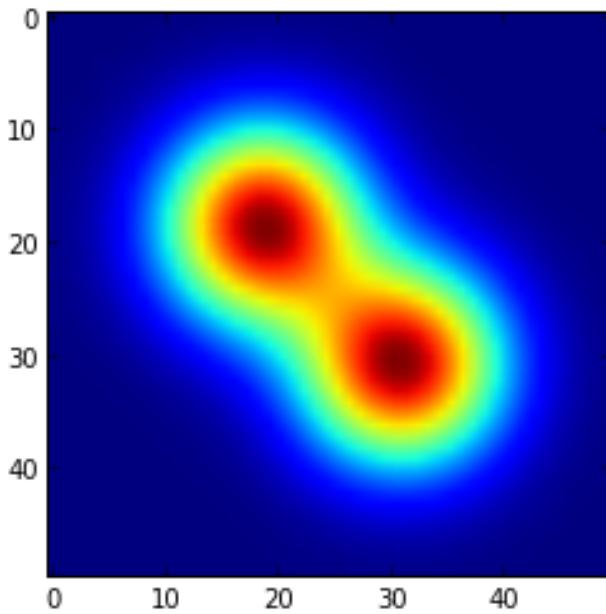
def f(a, x):
    y = norm(loc=a).pdf(x)
    Y = np.reshape(y, (1, len(y)))
    return Y.T * Y

x = np.linspace(-4, 4, 50)
Z = f(-1, x) + f(1, x)
```

Part 1

```
In [6]: plt.imshow(Z)
```

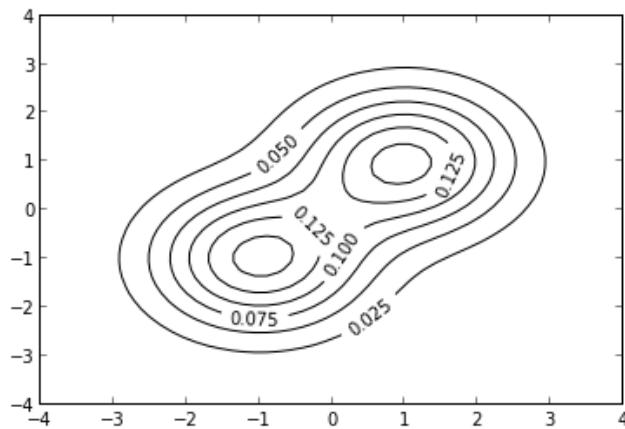
```
Out [6]: <matplotlib.image.AxesImage at 0x4636910>
```



Part 2

```
In [7]: fig, ax = plt.subplots()
X, Y = np.meshgrid(x, x)
p = ax.contour(X, Y, Z, colors='k')
ax.clabel(p)
```

```
Out [7]: <a list of 6 text.Text objects>
```



Part 3

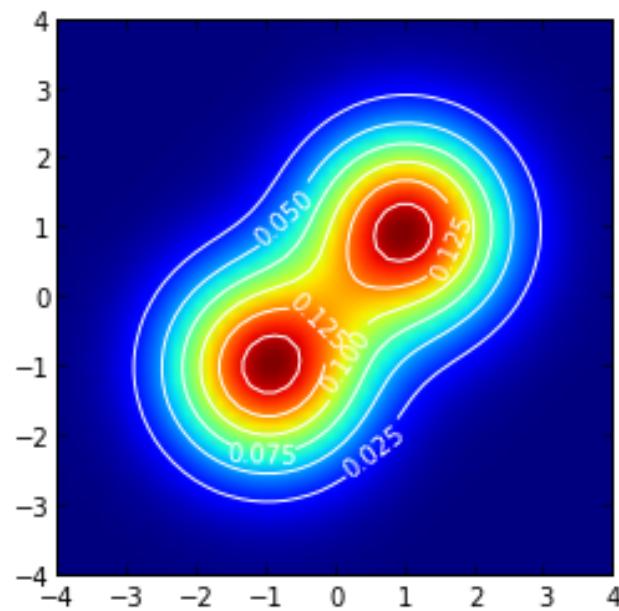
```
In [8]: fig, ax = plt.subplots()

# Flip Z vertically
Z_vflip = Z[:,::-1]
ax.imshow(Z_vflip, extent=[x[0], x[-1], x[0], x[-1]])

# Alternatively this will use the lower-left corner as the image origin
# ax.imshow(Z, origin='lower', extent=[x[0], x[-1], x[0], x[-1]])

X, Y = np.meshgrid(x, x)
p = ax.contour(X, Y, Z, colors='w')
ax.clabel(p)
```

Out [8]: <a list of 6 text.Text objects>



1.4 Solution 2.1 : Numpy arrays

Part 1

```
In [9]: print np.arange(20) + 1      # 1.
print np.arange(0, 30, 3)        # 2.
print np.linspace(0, 1, 8)       # 3.
n = np.arange(10) + 1           #
print 2*n + n**2                # 4.

[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
[ 0  3  6  9 12 15 18 21 24 27]
[ 0.          0.14285714  0.28571429  0.42857143  0.57142857
 0.71428571
 0.85714286  1.          ]
[ 3  8  15  24  35  48  63  80  99 120]
```

Part 2. Broadcasting

```
In [10]: A = np.arange(1, 9)
A = A.reshape(2, 4)
print A

[[1 2 3 4]
 [5 6 7 8]]
```

```
In [11]: B = np.array([1, 2])
print B

[1 2]
```

```
In [12]: print A + B.reshape((2, 1))

[[ 2  3  4  5]
 [ 7  8  9 10]]
```

1.5 Solution 2.2 : trapezoidal integration

Part 1

```
In [13]: x = np.linspace(0, 3, 10)
y = x**2
```

Part 2

```
In [14]: y[1:] + y[:-1]

Out [14]: array([ 0.11111111,  0.55555556,  1.44444444,  2.77777778,
 4.55555556,  6.77777778,  9.44444444, 12.55555556,
16.11111111])
```

Part 3

```
In [15]: def trapz(x, y):
    dx = x[1:] - x[:-1]
    dy = y[1:] + y[:-1]
    return np.sum(dx * dy) / 2
```

Part 4

```
In [16]: trapz(x, y)
```

```
Out [16]: 9.055555555555554
```

Part 5 (extension)

```
In [17]: # Solution 2.2.5
import scipy.integrate

scipy.integrate.trapz(y, x)
```

```
Out [17]: 9.055555555555554
```

1.6 Solution 3 : Creating NetCDF

Part 1

```
In [18]: from netcdftime import num2date, date2num
from datetime import datetime, timedelta

dates = []
units = 'seconds since 2013-03-19 00:00:00'
dt = datetime(2013, 3, 19, 0, 0, 0)
while dt < datetime(2013, 3, 19, 12, 0, 0):
    dt = dt + timedelta(minutes=5)
    dates.append(dt)

times = date2num(dates, units=units, calendar='gregorian')
print num2date(times[:5], units)
print num2date(times[-5:], units)
```

```
[datetime.datetime(2013, 3, 19, 0, 5) datetime.datetime(2013, 3, 19,
0, 10)
 datetime.datetime(2013, 3, 19, 0, 15)
 datetime.datetime(2013, 3, 19, 0, 20)
 datetime.datetime(2013, 3, 19, 0, 25)]
[datetime.datetime(2013, 3, 19, 11, 40)
 datetime.datetime(2013, 3, 19, 11, 45)
 datetime.datetime(2013, 3, 19, 11, 50)
 datetime.datetime(2013, 3, 19, 11, 55)
 datetime.datetime(2013, 3, 19, 12, 0)]
```

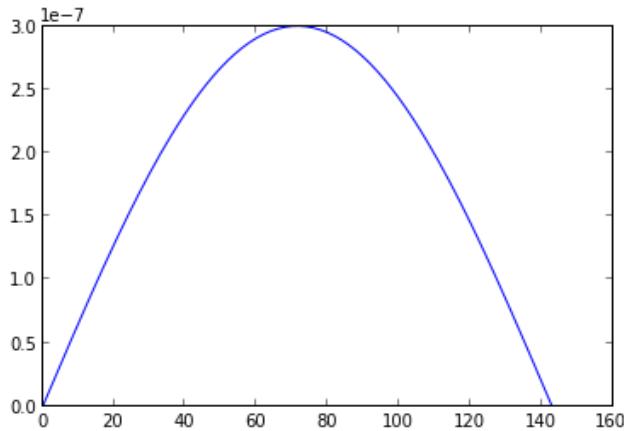
Part 2

```
In [19]: ! rm data/ex5.nc
```

```
In [20]: from netCDF4 import Dataset  
  
# 1.  
d = Dataset('data/ex5.nc', 'w', format='NETCDF4_CLASSIC')  
time_d = d.createDimension('time', len(times))  
time_v = d.createVariable('time', np.int32, ('time', ))  
o3_v = d.createVariable('o3', np.float64, ('time', ))
```

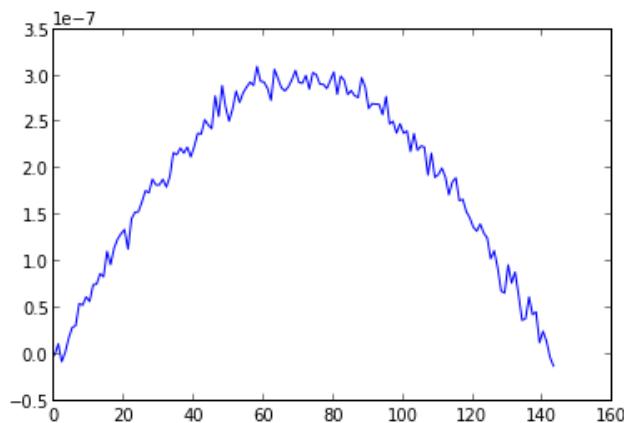
```
In [21]: # 2.  
o3_v[:] = sin(np.linspace(0, pi, len(times))) * 300 * 1e-9  
time_v[:] = np.array(times)  
plot(o3_v)
```

```
Out [21]: [<matplotlib.lines.Line2D at 0x4e11710>]
```



```
In [22]: # 3.  
o3_v[:] = o3_v + np.random.normal(size=len(times)) * 10 * 1e-9  
plot(o3_v)
```

```
Out [22]: [<matplotlib.lines.Line2D at 0x5058110>]
```



Part 3

```
In [23]: d.Conventions = 'CF-1.6'

o3_v.standard_name = 'mass_fraction_of_ozone_in_air'
o3_v.units = "1"
time_v.standard_name = 'time'
time_v.units = units
time_v.calendar = 'gregorian'

d.close()
```

```
In [24]: %%sh
cf-checker data/ex5.nc
```

```
CHECKING NetCDF FILE: data/ex5.nc
=====
Using CF Checker Version 2.0.5
Checking against CF Version CF-1.6
Using Standard Name Table Version 26 (2013-11-08T06:09:34Z)
Using Area Type Table Version 2 (10 July 2013)

-----
Checking variable: o3
-----

-----
Checking variable: time
-----

ERRORS detected: 0
WARNINGS given: 0
INFORMATION messages: 0
```

1.7 Solution 4.1 : Converting PP files to NetCDF

Part 1

```
In [25]: import cf
from glob import glob

pp_files = glob('data/aatzja.pm90*.pp')
datasets = [cf.read(f) for f in pp_files]
fields = [d.select('cloud_area_fraction') for d in datasets]
```

Part 2

```
In [26]: cloud_agg = cf.aggregate(fields)
print cloud_agg

cloud_area_fraction field summary
-----
Data           : cloud_area_fraction(time(12), latitude(73),
longitude(96)) 1
Cell methods   : time: mean
Dimensions     : time(12) = [1890-01-16 00:00:00, ..., 1890-12-16
00:00:00] 360_day
                  : latitude(73) = [90.0, ..., -90.0] degrees_north
                  : longitude(96) = [0.0, ..., 356.25] degrees_east
```

Part 3

```
In [27]: cf.write(cloud_agg, 'data/ex_4.1.nc')
```

```
In [28]: %%sh  
ncdump -h data/ex_4.1.nc
```

```
netcdf ex_4.1 {  
dimensions:  
    time = 12 ;  
    bounds2 = 2 ;  
    latitude = 73 ;  
    longitude = 96 ;  
variables:  
    double time_bounds(time, bounds2) ;  
    double time(time) ;  
        time:units = "days since 1890-1-1" ;  
        time:standard_name = "time" ;  
        time:bounds = "time_bounds" ;  
        time:calendar = "360_day" ;  
        time:axis = "T" ;  
    double latitude_bounds(latitude, bounds2) ;  
    double latitude(latitude) ;  
        latitude:units = "degrees_north" ;  
        latitude:standard_name = "latitude" ;  
        latitude:bounds = "latitude_bounds" ;  
        latitude:axis = "Y" ;  
    double longitude_bounds(longitude, bounds2) ;  
    double longitude(longitude) ;  
        longitude:units = "degrees_east" ;  
        longitude:standard_name = "longitude" ;  
        longitude:bounds = "longitude_bounds" ;  
        longitude:axis = "X" ;  
    float cloud_area_fraction(time, latitude, longitude) ;  
        cloud_area_fraction:_FillValue = -1073741824. ;  
        cloud_area_fraction:long_name = "TOTAL CLOUD AMOUNT IN  
LW RADIATION" ;  
        cloud_area_fraction:standard_name =  
"cloud_area_fraction" ;  
        cloud_area_fraction:cell_methods = "time: mean" ;  
        cloud_area_fraction:units = "1" ;  
  
// global attributes:  
    :Conventions = "CF-1.5" ;  
    :source = "UM" ;  
    :runid = "aatzj" ;  
    :stash_code = 2204 ;  
    :lbproc = 0 ;  
    :submodel = 1 ;  
    :history = "Converted from PP by cf-python v0.9.8.1" ;  
}
```

1.8 Solution 4.2 : Averaging and plotting gridded data

Part 1

```
In [29]: %%sh  
cdo zonmean data/ex_4.1.nc data/ex_4.1_zmean.nc
```

```
cdo zonmean: Processed 84096 values from 1 variable over 12 timesteps  
( 0.00s )
```

Part 2

```
In [30]: import iris  
from iris import quickplot as qplt  
  
cloud_zmean = iris.load_cube('data/ex_4.1_zmean.nc')  
qplt.contourf(cloud_zmean[:, :, 0], 20)
```

```
Out [30]: <matplotlib.contour.QuadContourSet instance at 0xf8b31b8>
```

