05-Exercises

Stephen Pascoe

March 17, 2014

1 Exercises

In [18]:
```
import numpy as np
from pylab import *
from matplotlib import pyplot as plt
```

1.1 Exercise 1.1 : Starting Matplotlib

When run from the command-line IPython behaves slightly differently to within a notebook. We will now try running the code from the Matplotlib introduction within an IPython terminal.

Part 1

Start IPython from the UNIX shell and import pylab. Reproduce the plot of $y = x^2$ as demonstrated in the introduction.

How is the plot displayed? What options do you have for interacting with the plot?

Part 2

Create a plot of the function $y = x^3$ between $-10 < x < -10$.

Try to make the style a green continuous line with cross-shaped point markers.

1.2 Exercise 1.2 : Plot layout and saving

Part 1

1. Using Matplotlib's object orientated API, plot the two functions on the same graph:
   - $y_1 = x$
   - $y_2 = x^2 + 6$
2. Add suitable legend and axis labels.
3. Save the figure as a PDF file and open it in a PDF viewer to check it was successful.

Part 2

Create 2 plots side by side showing a close-up of the two intersecting points between y_1 and y_2. In both plots display the gridlines to help identify the intersects. Thus visually confirm the solution to the equation $x^2 - x - 6 = 0$

1.3 Exercise 1.3 : Plotting 2D data

Part 1

Use the code below to generate the martix Z. Plot Z using the `imshow()`.

In [17]:
```
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 2

Plot Z using `contour()` with labelled contour lines. Ensure the axes covers the range $[-4, 4]$ in x and y. How does this plot differ from the previous one and why?

Part 3

Combine these two plots into a single overlayed plot showing a colour field and contour plots whist maintaining the correct x and y axis ranges.

HINT: you will need to flip Z vertically

1.4 Exercise 2.1 : Numpy arrays

Part 1

Create and print the following arrays

1. An array counting from 1 to 20 inclusive
2. The array of multiples of 3 greater than 0 and less than 30
3. The array of 8 equally spaced floats x where $0 \leq x \leq 1$
4. The array of integers $\sum_{n=1}^{10} 2n + n^2$

Part 2. Broadcasting

1. Use `np.arange` and `reshape` to create the array `A = [[1 2 3 4] [5 6 7 8]]`
2. Use `np.array` to create the array `B = [1 2]`
3. Use broadcasting to add `B` to `A` to create the final array `A + B = [[2 3 4 5] [7 8 9 10]]`

Hint: what shape does `B` have to be changed to?

1.5 Exercise 2.2 : trapezoidal integration

In this exercise, you are tasked with implementing the simple trapezoid rule formula for numerical integration. If we want to compute the definite integral

$$\int_a^b f(x)dx$$

we can partition the integration interval $[a, b]$ into smaller subintervals. We then approximate the area under the curve for each subinterval by calculating the area of the trapezoid created by linearly interpolating between the two function values at each end of the subinterval:

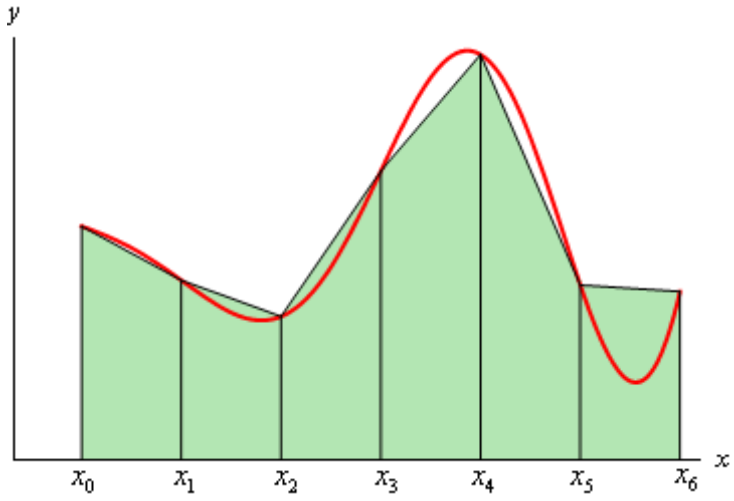

Figure 1: Illustration of the trapezoidal rule

For a pre-computed y array (where $y = f(x)$ at discrete samples) the trapezoidal rule equation is:

$$\int_a^b f(x)dx \approx \frac{1}{2}\sum_{i=1}^{n}(x_i - x_{i-1})(y_i + y_{i-1}).$$

In pure python, this can be written as:

```
def trapz_slow(x, y):
    area = 0.
    for i in range(1, len(x)):
        area += (x[i] - x[i-1]) * (y[i] + y[i-1])
    return area / 2
```

Part 1

Create two arrays x and y, where x is a linearly spaced array in the interval $[0, 3]$ of length 10, and y represents the function $f(x) = x^2$ sampled at x.

Part 2

Use indexing (not a for loop) to find the 9 values representing $y_i + y_{i-1}$ for i between 1 and 9 (where y_0 is the first element of y).

Hint: What indexing would be needed to get all but the last element of the 1d array y. Similarly what indexing would be needed to get all but the first element of a 1d array.

Part 3

Write a function `trapz(x, y)` that applies the trapezoid formula to pre-computed values, where `x` and `y` are 1-d arrays. The function should not use a for loop.

Part 4

Verify that your function is correct by using the arrays created in #1 as input to `trapz`. Your answer should be a close approximation of $\int_0^3 x^2$ which is 9.

Part 5 (extension)

`scipy.integrate` provides many common integration schemes. Find a suitable function to perform the trapezoidal integration implemented above and check its result with your own:

1.6 Exersice 3 : Creating NetCDF

In this exercise we will create a CF-NetCDF file containing a time series of synthetic Ozone data.

Part 1

1. Use the function `netcdftime.date2num()` to create a numpy array containing time in seconds from `2013-03-19 00:00:00` until `2013-03-19 12:00:00` at 5 minute resolution.
2. Print the first 5 timepoints in date and time format.
3. Print the last 5 timepoints in date and time format.

Part 2

1. Create a NetCDF file containing a single data variable `o3` using the time axis created in part 1.
2. Fill the data variable with a sin wave between 0 and π radians with a peak at 300 ppb. Plot the resulting variable using `pyplot.plot()`.
3. Add noise of $+/-$ 10ppb to the data using the function `np.random.normal()`. Plot the resulting variable using `pyplot.plot()`.

Note: When plotting don't worry about displaying dates on the x axis. This is just for verification.

Part 3

1. Add the CF Metadata global attribute `Conventions`, and the variable attributes `standard_name` and `units` to the file.
2. Check the CF-compliance of your file with the `cf-checker` command. What extra metadata is required?
3. Amend your metadata until the file is CF compliant.

1.7 Exercise 4.1 : Converting PP files to NetCDF

Part 1

Open each PP file in the data directory of the form `aatzja.pm90*.pp` using cf-python. Create a list containing the field `cloud_area_fraction` from each file. *HINT: You can use the function glob.glob to select every PP*

file in the data directory.

Part 2

Apply the function `cf.aggregate()` to the list of fields and save the result in the varaible `cloud_agg`. Print this variable; what has the `aggregate()` function returned?

Part 3

Save the result in the file `data/ex_4.1.nc`. Use `ncdump -h` to investigate the result.

1.8 Exercise 4.2 : Averaging and plotting gridded data

Although both cf-python and IRIS can calculate aggregate statistics CDO offers the most convenient solution to averaging whole files.

Part 1

Use the CDO operator `zonmean` to calculate the zonal mean of the `ex_4.1.nc`. You can use the the command `cdo -h zonmean` to find out the syntax.

Part 2

Load the output of Part 1 into IRIS and plot the result with `iris.quickplot.contourf()`.