05-Examples

Unknown Author

March 10, 2014

In [24]:
```
import numpy as np
from pylab import *
```

0.1 Exercise 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.

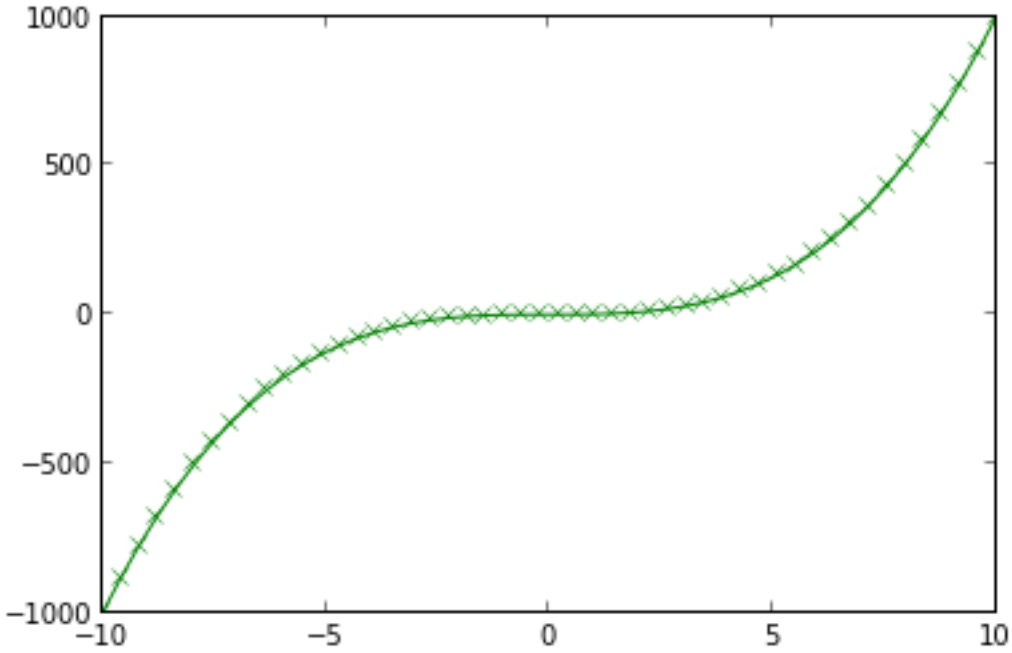
1. Start IPython and import pylab. Reproduce the plot of $y = x^2$ as demonstrated in the introduction.

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

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.

In [25]:
```
tut_xy = (x, y)
x = np.linspace(-10, 10)
y = x**3
plot(x, y, 'gx-')
(x, y) = tut_xy
```



0.2 Exercise 2 : Plot layout and saving

TODO

0.3 Exercise 3 : Numpy arrays

1. Create and print the following arrays

a) An array counting from 1 to 20 inclusive

b) The array of multiples of 3 greater than 0 and less than 30

c) The array of 8 equally spaced floats x where $0 \leq x \leq 1$

d) The array of integers $\sum_{n=1}^{10} 2n + n^2$

In [12]:
```
print np.arange(20) + 1
print np.arange(0, 30, 3)
print np.linspace(0, 1, 8)
n = np.arange(10) + 1
print 2*n + n**2
```

```
[ 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]
```

2. Broadcasting

a) Use `np.arange` and `reshape` to create the array

A = [[1 2 3 4][5 6 7 8]]

In [41]:
```
A = np.arange(1, 9)
A = A.reshape(2,4)
print A
```

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

b) Use `np.array` to create the array

B = [1 2]

In [43]:
```
B = np.array([1,2])
print B
```

```
[1 2]
```

c) 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?

In [46]:
```
print A + B.reshape((2,1))
```

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

0.4 Exercise 3 : Plotting 2D data

TODO

0.5 Exercise 4 : 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:

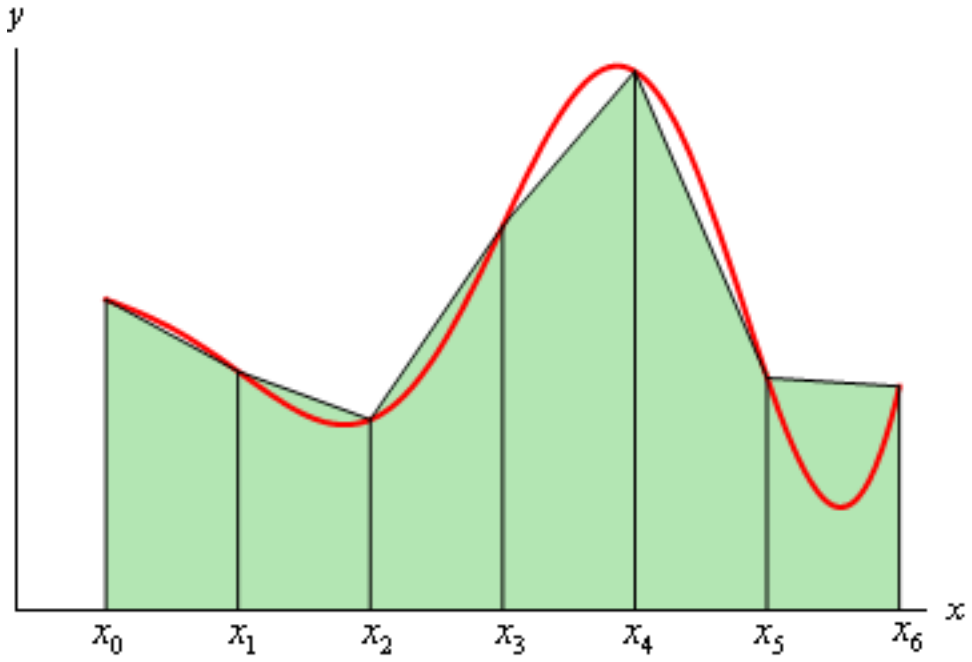


Figure 1: Illustration of the trapezoidal rule

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.

In [4]:
```
x = np.linspace(0, 3, 10)
y = x**2
```

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.

In [11]:
```
y[1:] + y[:-1]
```

Out [11]:
```
array([  0.11111111,   0.55555556,   1.44444444,   2.77777778,
         4.55555556,   6.77777778,   9.44444444,  12.55555556,
        16.11111111])
```

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.

In [21]:
```
def trapz(x, y):
    dx = x[1:] - x[:-1]
    dy = y[1:] + y[:-1]
    return np.sum(dx * dy) / 2
```

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.

In [22]:
```
trapz(x, y)
```

Out [22]:
```
9.055555555555554
```

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:

In [20]:
```
import scipy.integrate

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

Out [20]:
```
9.055555555555554
```

In []: