Introduction to Scientific Computing in Python

Robert Johansson

June 18, 2020

Contents

In [1]:
```
# you only need to do this once
!pip install --upgrade version_information
```

```
Collecting version_information
  Using cached version_information-1.0.3.tar.gz (3.3 kB)
Using legacy setup.py install for version-information, since package 'wheel' is not installed.
Installing collected packages: version-information
    Running setup.py install for version-information: started
    Running setup.py install for version-information: finished with status 'done'
Successfully installed version-information-1.0.3


DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as
```

you only need to do this once

In [1]:
```
%load_ext version_information

%version_information numpy, scipy, matplotlib, sympy, version_information
```

Out[1]:

In [ ]:

```
In [3]: ls scripts

 Volume in drive C has no label.
 Volume Serial Number is D052-E858

 Directory of C:\Users\kumar\Code\scientific-python-lectures\scripts

12-02-2020  18:02    <DIR>          .
12-02-2020  18:02    <DIR>          ..
12-02-2020  18:02                48 hello-world.py
12-02-2020  18:02                74 hello-world-in-swedish.py
               2 File(s)            122 bytes
               2 Dir(s)  30,739,820,544 bytes free


In [4]: cat scripts/hello-world.py


          File "<ipython-input-4-e58e421cca7c>", line 1
        cat scripts/hello-world.py
            ^
    SyntaxError: invalid syntax



In [ ]: !python scripts/hello-world.py

In [ ]: cat scripts/hello-world-in-swedish.py

In [ ]: !python scripts/hello-world-in-swedish.py

In [ ]: import math

In [ ]: import math

        x = math.cos(2 * math.pi)

        print(x)

In [ ]: from math import *

        x = cos(2 * pi)

        print(x)

In [ ]: from math import cos, pi

        x = cos(2 * pi)

        print(x)

In [ ]: import math

        print(dir(math))

In [ ]: help(math.log)
```

```
In [ ]: log(10)

In [ ]: log(10, 2)

In [ ]: # variable assignments
        x = 1.0
        my_variable = 12.2

In [ ]: type(x)

In [ ]: x = 1

In [ ]: type(x)

In [ ]: print(y)

In [ ]: # integers
        x = 1
        type(x)

In [ ]: # float
        x = 1.0
        type(x)

In [ ]: # boolean
        b1 = True
        b2 = False

        type(b1)

In [ ]: # complex numbers: note the use of `j` to specify the imaginary part
        x = 1.0 - 1.0j
        type(x)

In [ ]: print(x)

In [ ]: print(x.real, x.imag)

In [ ]: import types

        # print all types defined in the `types` module
        print(dir(types))

In [ ]: x = 1.0

        # check if the variable x is a float
        type(x) is float

In [ ]: # check if the variable x is an int
        type(x) is int

In [ ]: isinstance(x, float)

In [ ]: x = 1.5

        print(x, type(x))
```

In [ ]:
```
x = int(x)

print(x, type(x))
```

In [ ]:
```
z = complex(x)

print(z, type(z))
```

In [ ]:
```
x = float(z)
```

In [ ]:
```
y = bool(z.real)

print(z.real, " -> ", y, type(y))

y = bool(z.imag)

print(z.imag, " -> ", y, type(y))
```

In [ ]:
```
1 + 2, 1 - 2, 1 * 2, 1 / 2
```

In [ ]:
```
1.0 + 2.0, 1.0 - 2.0, 1.0 * 2.0, 1.0 / 2.0
```

In [ ]:
```
# Integer division of float numbers
3.0 // 2.0
```

In [ ]:
```
# Note! The power operators in python isn't ^, but **
2 ** 2
```

In [ ]:
```
True and False
```

In [ ]:
```
not False
```

In [ ]:
```
True or False
```

In [ ]:
```
2 > 1, 2 < 1
```

In [ ]:
```
2 > 2, 2 < 2
```

In [ ]:
```
2 >= 2, 2 <= 2
```

In [ ]:
```
# equality
[1,2] == [1,2]
```

In [ ]:
```
# objects identical?
l1 = l2 = [1,2]

l1 is l2
```

In [ ]:
```
s = "Hello world"
type(s)
```

In [ ]:
```
# length of the string: the number of characters
len(s)
```

In [ ]:
```
# replace a substring in a string with something else
s2 = s.replace("world", "test")
print(s2)
```

```
In [ ]: s[0]

In [ ]: s[0:5]

In [ ]: s[4:5]

In [ ]: s[:5]

In [ ]: s[6:]

In [ ]: s[:]

In [ ]: s[::1]

In [ ]: s[::2]

In [ ]: print("str1", "str2", "str3")  # The print statement concatenates strings with a space

In [ ]: print("str1", 1.0, False, -1j)  # The print statements converts all arguments to strings

In [ ]: print("str1" + "str2" + "str3") # strings added with + are concatenated without space

In [ ]: print("value = %f" % 1.0)       # we can use C-style string formatting

In [ ]: # this formatting creates a string
        s2 = "value1 = %.2f. value2 = %d" % (3.1415, 1.5)

        print(s2)

In [ ]: # alternative, more intuitive way of formatting a string
        s3 = 'value1 = {0}, value2 = {1}'.format(3.1415, 1.5)

        print(s3)

In [ ]: l = [1,2,3,4]

        print(type(l))
        print(l)

In [ ]: print(l)

        print(l[1:3])

        print(l[::2])

In [ ]: l[0]

In [ ]: l = [1, 'a', 1.0, 1-1j]

        print(l)

In [ ]: nested_list = [1, [2, [3, [4, [5]]]]]

        nested_list

In [ ]: start = 10
        stop = 30
        step = 2

        range(start, stop, step)
```

In [ ]:
```
# in python 3 range generates an iterator, which can be converted to a list using 'list(...)'.
# It has no effect in python 2
list(range(start, stop, step))
```

In [ ]:
```
list(range(-10, 10))
```

In [ ]:
```
s
```

In [ ]:
```
# convert a string to a list by type casting:
s2 = list(s)

s2
```

In [ ]:
```
# sorting lists
s2.sort()

print(s2)
```

In [ ]:
```
# create a new empty list
l = []

# add an elements using `append`
l.append("A")
l.append("d")
l.append("d")

print(l)
```

In [ ]:
```
l[1] = "p"
l[2] = "p"

print(l)
```

In [ ]:
```
l[1:3] = ["d", "d"]

print(l)
```

In [ ]:
```
l.insert(0, "i")
l.insert(1, "n")
l.insert(2, "s")
l.insert(3, "e")
l.insert(4, "r")
l.insert(5, "t")

print(l)
```

In [ ]:
```
l.remove("A")

print(l)
```

In [ ]:
```
del l[7]
del l[6]

print(l)
```

In [ ]:
```
point = (10, 20)

print(point, type(point))
```

```
In [ ]: point = 10, 20

        print(point, type(point))

In [ ]: x, y = point

        print("x =", x)
        print("y =", y)

In [ ]: point[0] = 20

In [ ]: params = {"parameter1" : 1.0,
                  "parameter2" : 2.0,
                  "parameter3" : 3.0,}

        print(type(params))
        print(params)

In [ ]: print("parameter1 = " + str(params["parameter1"]))
        print("parameter2 = " + str(params["parameter2"]))
        print("parameter3 = " + str(params["parameter3"]))

In [ ]: params["parameter1"] = "A"
        params["parameter2"] = "B"

        # add a new entry
        params["parameter4"] = "D"

        print("parameter1 = " + str(params["parameter1"]))
        print("parameter2 = " + str(params["parameter2"]))
        print("parameter3 = " + str(params["parameter3"]))
        print("parameter4 = " + str(params["parameter4"]))

In [ ]: statement1 = False
        statement2 = False

        if statement1:
            print("statement1 is True")

        elif statement2:
            print("statement2 is True")

        else:
            print("statement1 and statement2 are False")

In [ ]: statement1 = statement2 = True

        if statement1:
            if statement2:
                print("both statement1 and statement2 are True")

In [ ]: # Bad indentation!
        if statement1:
            if statement2:
            print("both statement1 and statement2 are True")  # this line is not properly indented
```

```
In [ ]: statement1 = False

        if statement1:
            print("printed if statement1 is True")

            print("still inside the if block")
```

```
In [ ]: if statement1:
            print("printed if statement1 is True")

        print("now outside the if block")
```

```
In [ ]: for x in [1,2,3]:
            print(x)
```

```
In [ ]: for x in range(4): # by default range start at 0
            print(x)
```

```
In [ ]: for x in range(-3,3):
            print(x)
```

```
In [ ]: for word in ["scientific", "computing", "with", "python"]:
            print(word)
```

```
In [ ]: for key, value in params.items():
            print(key + " = " + str(value))
```

```
In [ ]: for idx, x in enumerate(range(-3,3)):
            print(idx, x)
```

```
In [ ]: l1 = [x**2 for x in range(0,5)]

        print(l1)
```

```
In [ ]: i = 0

        while i < 5:
            print(i)

            i = i + 1

        print("done")
```

```
In [ ]: def func0():
            print("test")
```

```
In [ ]: func0()
```

```
In [ ]: def func1(s):
            """
            Print a string 's' and tell how many characters it has
            """

            print(s + " has " + str(len(s)) + " characters")
```

```
In [ ]: help(func1)
```

```
In [ ]: func1("test")

In [ ]: def square(x):
            """
            Return the square of x.
            """
            return x ** 2

In [ ]: square(4)

In [ ]: def powers(x):
            """
            Return a few powers of x.
            """
            return x ** 2, x ** 3, x ** 4

In [ ]: powers(3)

In [ ]: x2, x3, x4 = powers(3)

        print(x3)

In [ ]: def myfunc(x, p=2, debug=False):
            if debug:
                print("evaluating myfunc for x = " + str(x) + " using exponent p = " + str(p))
            return x**p

In [ ]: myfunc(5)

In [ ]: myfunc(5, debug=True)

In [ ]: myfunc(p=3, debug=True, x=7)

In [ ]: f1 = lambda x: x**2

        # is equivalent to

        def f2(x):
            return x**2

In [ ]: f1(2), f2(2)

In [ ]: # map is a built-in python function
        map(lambda x: x**2, range(-3,4))

In [ ]: # in python 3 we can use `list(...)` to convert the iterator to an explicit list
        list(map(lambda x: x**2, range(-3,4)))

In [ ]: class Point:
            """
            Simple class for representing a point in a Cartesian coordinate system.
            """

            def __init__(self, x, y):
                """
                Create a new Point at x, y.
                """
```

```
            self.x = x
            self.y = y

        def translate(self, dx, dy):
            """
            Translate the point by dx and dy in the x and y direction.
            """
            self.x += dx
            self.y += dy

        def __str__(self):
            return("Point at [%f, %f]" % (self.x, self.y))
```

In [ ]:
```
p1 = Point(0, 0) # this will invoke the __init__ method in the Point class

print(p1)         # this will invoke the __str__ method
```

In [ ]:
```
p2 = Point(1, 1)

p1.translate(0.25, 1.5)

print(p1)
print(p2)
```

In [ ]:
```
%%file mymodule.py
"""
Example of a python module. Contains a variable called my_variable,
a function called my_function, and a class called MyClass.
"""

my_variable = 0

def my_function():
    """
    Example function
    """
    return my_variable

class MyClass:
    """
    Example class.
    """

    def __init__(self):
        self.variable = my_variable

    def set_variable(self, new_value):
        """
        Set self.variable to a new value
        """
        self.variable = new_value

    def get_variable(self):
        return self.variable
```

In [ ]:
```
import mymodule
```

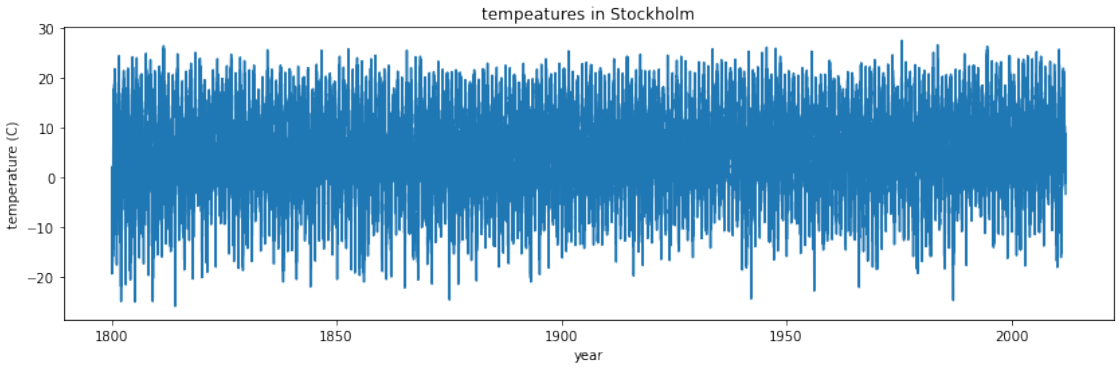
```
In [ ]: help(mymodule)

In [ ]: mymodule.my_variable

In [ ]: mymodule.my_function()

In [ ]: my_class = mymodule.MyClass()
        my_class.set_variable(10)
        my_class.get_variable()

In [ ]: reload(mymodule)  # works only in python 2

In [ ]: raise Exception("description of the error")

In [ ]: try:
            print("test")
            # generate an error: the variable test is not defined
            print(test)
        except:
            print("Caught an exception")

In [ ]: try:
            print("test")
            # generate an error: the variable test is not defined
            print(test)
        except Exception as e:
            print("Caught an exception:" + str(e))

In [ ]: %reload_ext version_information

        %version_information
```

```
In [1]: # what is this line all about?!? Answer in lecture 4
        %matplotlib inline
        import matplotlib.pyplot as plt

In [2]: from numpy import *

In [3]: # a vector: the argument to the array function is a Python list
        v = array([1,2,3,4])

        v

Out[3]: array([1, 2, 3, 4])

In [4]: # a matrix: the argument to the array function is a nested Python list
        M = array([[1, 2], [3, 4]])

        M

Out[4]: array([[1, 2],
               [3, 4]])

In [5]: type(v), type(M)

Out[5]: (numpy.ndarray, numpy.ndarray)

In [6]: v.shape

Out[6]: (4,)

In [7]: M.shape

Out[7]: (2, 2)

In [8]: M.size

Out[8]: 4

In [9]: shape(M)

Out[9]: (2, 2)

In [10]: size(M)

Out[10]: 4

In [11]: M.dtype

Out[11]: dtype('int32')

In [12]: M[0,0] = "hello"

         ---------------------------------------------------------------------------

         ValueError                                Traceback (most recent call last)

         <ipython-input-12-e1f336250f69> in <module>
    ----> 1 M[0,0] = "hello"


         ValueError: invalid literal for int() with base 10: 'hello'
```

```
In [13]: M = array([[1, 2], [3, 4]], dtype=complex)

         M

Out[13]: array([[1.+0.j, 2.+0.j],
                [3.+0.j, 4.+0.j]])

In [14]: # create a range

         x = arange(0, 10, 1) # arguments: start, stop, step

         x

Out[14]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [15]: x = arange(-1, 1, 0.1)

         x

Out[15]: array([-1.00000000e+00, -9.00000000e-01, -8.00000000e-01, -7.00000000e-01,
                -6.00000000e-01, -5.00000000e-01, -4.00000000e-01, -3.00000000e-01,
                -2.00000000e-01, -1.00000000e-01, -2.22044605e-16,  1.00000000e-01,
                 2.00000000e-01,  3.00000000e-01,  4.00000000e-01,  5.00000000e-01,
                 6.00000000e-01,  7.00000000e-01,  8.00000000e-01,  9.00000000e-01])

In [16]: # using linspace, both end points ARE included
         linspace(0, 10, 25)

Out[16]: array([ 0.        ,  0.41666667,  0.83333333,  1.25      ,  1.66666667,
                 2.08333333,  2.5       ,  2.91666667,  3.33333333,  3.75      ,
                 4.16666667,  4.58333333,  5.        ,  5.41666667,  5.83333333,
                 6.25      ,  6.66666667,  7.08333333,  7.5       ,  7.91666667,
                 8.33333333,  8.75      ,  9.16666667,  9.58333333, 10.        ])

In [17]: logspace(0, 10, 10, base=e)

Out[17]: array([1.00000000e+00, 3.03773178e+00, 9.22781435e+00, 2.80316249e+01,
                8.51525577e+01, 2.58670631e+02, 7.85771994e+02, 2.38696456e+03,
                7.25095809e+03, 2.20264658e+04])

In [18]: x, y = mgrid[0:5, 0:5] # similar to meshgrid in MATLAB

In [19]: x

Out[19]: array([[0, 0, 0, 0, 0],
                [1, 1, 1, 1, 1],
                [2, 2, 2, 2, 2],
                [3, 3, 3, 3, 3],
                [4, 4, 4, 4, 4]])

In [20]: y

Out[20]: array([[0, 1, 2, 3, 4],
                [0, 1, 2, 3, 4],
                [0, 1, 2, 3, 4],
                [0, 1, 2, 3, 4],
                [0, 1, 2, 3, 4]])
```

```
In [21]: from numpy import random

In [22]: # uniform random numbers in [0,1]
         random.rand(5,5)

Out[22]: array([[0.20656589, 0.96440751, 0.33386388, 0.18873014, 0.05489782],
                [0.79548386, 0.08561849, 0.81373461, 0.3984876 , 0.1148569 ],
                [0.56145769, 0.58773648, 0.09979363, 0.15892703, 0.34946945],
                [0.50327789, 0.32181119, 0.91201925, 0.83376678, 0.66508015],
                [0.10289733, 0.35016121, 0.84715757, 0.90493504, 0.3627392 ]])

In [23]: # standard normal distributed random numbers
         random.randn(5,5)

Out[23]: array([[ 1.17062311e+00, -1.89027292e-01, -1.28514235e-01,
                  -2.91760135e-03, -1.54576572e-01],
                [-2.36814957e-01,  7.27160674e-01,  7.60848541e-01,
                   1.61074650e-01, -2.32172002e+00],
                [-1.36925226e+00, -1.25315831e+00,  1.09573136e+00,
                  -1.21711459e-01, -4.89806004e-01],
                [ 7.89509420e-01, -3.34464633e+00,  1.07718247e+00,
                  -1.28342886e-01,  1.03721330e+00],
                [ 4.56487990e-02,  2.12876959e-01,  7.31530348e-01,
                  -9.62340313e-01,  6.84866049e-01]])

In [24]: # a diagonal matrix
         diag([1,2,3])

Out[24]: array([[1, 0, 0],
                [0, 2, 0],
                [0, 0, 3]])

In [25]: # diagonal with offset from the main diagonal
         diag([1,2,3], k=1)

Out[25]: array([[0, 1, 0, 0],
                [0, 0, 2, 0],
                [0, 0, 0, 3],
                [0, 0, 0, 0]])

In [26]: zeros((3,3))

Out[26]: array([[0., 0., 0.],
                [0., 0., 0.],
                [0., 0., 0.]])

In [27]: ones((3,3))

Out[27]: array([[1., 1., 1.],
                [1., 1., 1.],
                [1., 1., 1.]])

In [101]: !head stockholm_td_adj.dat

'head' is not recognized as an internal or external command,
operable program or batch file.
```

```
In [29]: data = genfromtxt('stockholm_td_adj.dat')

In [30]: data.shape

Out[30]: (77431, 7)

In [31]: fig, ax = plt.subplots(figsize=(14,4))
         ax.plot(data[:,0]+data[:,1]/12.0+data[:,2]/365, data[:,5])
         ax.axis('tight')
         ax.set_title('tempeatures in Stockholm')
         ax.set_xlabel('year')
         ax.set_ylabel('temperature (C)');
```




```
In [32]: M = random.rand(3,3)

         M

Out[32]: array([[0.02182742, 0.9439656 , 0.55812415],
                [0.37941086, 0.4968775 , 0.55390142],
                [0.32145129, 0.05711751, 0.37056502]])

In [33]: savetxt("random-matrix.csv", M)

In [34]: !cat random-matrix.csv

'cat' is not recognized as an internal or external command,
operable program or batch file.


In [35]: savetxt("random-matrix.csv", M, fmt='%.5f') # fmt specifies the format

         !cat random-matrix.csv

'cat' is not recognized as an internal or external command,
operable program or batch file.


In [36]: save("random-matrix.npy", M)

         !file random-matrix.npy
```

```
'file' is not recognized as an internal or external command,
operable program or batch file.


In [37]: load("random-matrix.npy")

Out[37]: array([[0.02182742, 0.9439656 , 0.55812415],
                [0.37941086, 0.4968775 , 0.55390142],
                [0.32145129, 0.05711751, 0.37056502]])

In [38]: M.itemsize # bytes per element

Out[38]: 8

In [39]: M.nbytes # number of bytes

Out[39]: 72

In [40]: M.ndim # number of dimensions

Out[40]: 2

In [41]: # v is a vector, and has only one dimension, taking one index
         v[0]

Out[41]: 1

In [42]: # M is a matrix, or a 2 dimensional array, taking two indices
         M[1,1]

Out[42]: 0.4968774976428517

In [43]: M

Out[43]: array([[0.02182742, 0.9439656 , 0.55812415],
                [0.37941086, 0.4968775 , 0.55390142],
                [0.32145129, 0.05711751, 0.37056502]])

In [44]: M[1]

Out[44]: array([0.37941086, 0.4968775 , 0.55390142])

In [45]: M[1,:] # row 1

Out[45]: array([0.37941086, 0.4968775 , 0.55390142])

In [46]: M[:,1] # column 1

Out[46]: array([0.9439656 , 0.4968775 , 0.05711751])

In [47]: M[0,0] = 1

In [48]: M

Out[48]: array([[1.        , 0.9439656 , 0.55812415],
                [0.37941086, 0.4968775 , 0.55390142],
                [0.32145129, 0.05711751, 0.37056502]])
```

```
In [49]: # also works for rows and columns
         M[1,:] = 0
         M[:,2] = -1

In [50]: M

Out[50]: array([[ 1.        ,  0.9439656 , -1.        ],
                [ 0.        ,  0.        , -1.        ],
                [ 0.32145129,  0.05711751, -1.        ]])

In [51]: A = array([1,2,3,4,5])
         A

Out[51]: array([1, 2, 3, 4, 5])

In [52]: A[1:3]

Out[52]: array([2, 3])

In [53]: A[1:3] = [-2,-3]

         A

Out[53]: array([ 1, -2, -3,  4,  5])

In [54]: A[::] # lower, upper, step all take the default values

Out[54]: array([ 1, -2, -3,  4,  5])

In [55]: A[::2] # step is 2, lower and upper defaults to the beginning and end of the array

Out[55]: array([ 1, -3,  5])

In [56]: A[:3] # first three elements

Out[56]: array([ 1, -2, -3])

In [57]: A[3:] # elements from index 3

Out[57]: array([4, 5])

In [58]: A = array([1,2,3,4,5])

In [59]: A[-1] # the last element in the array

Out[59]: 5

In [60]: A[-3:] # the last three elements

Out[60]: array([3, 4, 5])

In [61]: A = array([[n+m*10 for n in range(5)] for m in range(5)])

         A

Out[61]: array([[ 0,  1,  2,  3,  4],
                [10, 11, 12, 13, 14],
                [20, 21, 22, 23, 24],
                [30, 31, 32, 33, 34],
                [40, 41, 42, 43, 44]])
```

```
In [62]: # a block from the original array
         A[1:4, 1:4]

Out[62]: array([[11, 12, 13],
                [21, 22, 23],
                [31, 32, 33]])

In [63]: # strides
         A[::2, ::2]

Out[63]: array([[ 0,  2,  4],
                [20, 22, 24],
                [40, 42, 44]])

In [64]: row_indices = [1, 2, 3]
         A[row_indices]

Out[64]: array([[10, 11, 12, 13, 14],
                [20, 21, 22, 23, 24],
                [30, 31, 32, 33, 34]])

In [65]: col_indices = [1, 2, -1] # remember, index -1 means the last element
         A[row_indices, col_indices]

Out[65]: array([11, 22, 34])

In [66]: B = array([n for n in range(5)])
         B

Out[66]: array([0, 1, 2, 3, 4])

In [67]: row_mask = array([True, False, True, False, False])
         B[row_mask]

Out[67]: array([0, 2])

In [68]: # same thing
         row_mask = array([1,0,1,0,0], dtype=bool)
         B[row_mask]

Out[68]: array([0, 2])

In [69]: x = arange(0, 10, 0.5)
         x

Out[69]: array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. , 5.5, 6. ,
                6.5, 7. , 7.5, 8. , 8.5, 9. , 9.5])

In [70]: mask = (5 < x) * (x < 7.5)

         mask

Out[70]: array([False, False, False, False, False, False, False, False, False,
                False, False,  True,  True,  True,  True, False, False, False,
                False, False])

In [71]: x[mask]
```

```
Out[71]: array([5.5, 6. , 6.5, 7. ])

In [72]: indices = where(mask)

         indices

Out[72]: (array([11, 12, 13, 14], dtype=int64),)

In [73]: x[indices] # this indexing is equivalent to the fancy indexing x[mask]

Out[73]: array([5.5, 6. , 6.5, 7. ])

In [74]: diag(A)

Out[74]: array([ 0, 11, 22, 33, 44])

In [75]: diag(A, -1)

Out[75]: array([10, 21, 32, 43])

In [76]: v2 = arange(-3,3)
         v2

Out[76]: array([-3, -2, -1,  0,  1,  2])

In [77]: row_indices = [1, 3, 5]
         v2[row_indices] # fancy indexing

Out[77]: array([-2,  0,  2])

In [78]: v2.take(row_indices)

Out[78]: array([-2,  0,  2])

In [79]: take([-3, -2, -1,  0,  1,  2], row_indices)

Out[79]: array([-2,  0,  2])

In [80]: which = [1, 0, 1, 0]
         choices = [[-2,-2,-2,-2], [5,5,5,5]]

         choose(which, choices)

Out[80]: array([ 5, -2,  5, -2])

In [81]: v1 = arange(0, 5)

In [82]: v1 * 2

Out[82]: array([0, 2, 4, 6, 8])

In [83]: v1 + 2

Out[83]: array([2, 3, 4, 5, 6])

In [84]: A * 2, A + 2
```

```
Out[84]: (array([[ 0,  2,  4,  6,  8],
                 [20, 22, 24, 26, 28],
                 [40, 42, 44, 46, 48],
                 [60, 62, 64, 66, 68],
                 [80, 82, 84, 86, 88]]),
          array([[ 2,  3,  4,  5,  6],
                 [12, 13, 14, 15, 16],
                 [22, 23, 24, 25, 26],
                 [32, 33, 34, 35, 36],
                 [42, 43, 44, 45, 46]]))

In [85]: A * A # element-wise multiplication

Out[85]: array([[   0,    1,    4,    9,   16],
                [ 100,  121,  144,  169,  196],
                [ 400,  441,  484,  529,  576],
                [ 900,  961, 1024, 1089, 1156],
                [1600, 1681, 1764, 1849, 1936]])

In [86]: v1 * v1

Out[86]: array([ 0,  1,  4,  9, 16])

In [87]: A.shape, v1.shape

Out[87]: ((5, 5), (5,))

In [88]: A * v1

Out[88]: array([[  0,   1,   4,   9,  16],
                [  0,  11,  24,  39,  56],
                [  0,  21,  44,  69,  96],
                [  0,  31,  64,  99, 136],
                [  0,  41,  84, 129, 176]])

In [89]: dot(A, A)

Out[89]: array([[ 300,  310,  320,  330,  340],
                [1300, 1360, 1420, 1480, 1540],
                [2300, 2410, 2520, 2630, 2740],
                [3300, 3460, 3620, 3780, 3940],
                [4300, 4510, 4720, 4930, 5140]])

In [90]: dot(A, v1)

Out[90]: array([ 30, 130, 230, 330, 430])

In [91]: dot(v1, v1)

Out[91]: 30

In [92]: M = matrix(A)
         v = matrix(v1).T # make it a column vector

In [93]: v
```

```
Out[93]: matrix([[0],
                 [1],
                 [2],
                 [3],
                 [4]])

In [94]: M * M

Out[94]: matrix([[ 300,  310,  320,  330,  340],
                 [1300, 1360, 1420, 1480, 1540],
                 [2300, 2410, 2520, 2630, 2740],
                 [3300, 3460, 3620, 3780, 3940],
                 [4300, 4510, 4720, 4930, 5140]])

In [95]: M * v

Out[95]: matrix([[ 30],
                 [130],
                 [230],
                 [330],
                 [430]])

In [96]: # inner product
         v.T * v

Out[96]: matrix([[30]])

In [97]: # with matrix objects, standard matrix algebra applies
         v + M*v

Out[97]: matrix([[ 30],
                 [131],
                 [232],
                 [333],
                 [434]])

In [98]: v = matrix([1,2,3,4,5,6]).T

In [99]: shape(M), shape(v)

Out[99]: ((5, 5), (6, 1))

In [100]: M * v


         ---------------------------------------------------------------------------

         ValueError                                Traceback (most recent call last)

         <ipython-input-100-e8f88679fe45> in <module>
    ----> 1 M * v


         c:\program files\python38\lib\site-packages\numpy\matrixlib\defmatrix.py in __mul__(self, other)
         218         if isinstance(other, (N.ndarray, list, tuple)) :
         219             # This promotes 1-D vectors to row vectors
    --> 220             return N.dot(self, asmatrix(other))
```

```
        221         if isscalar(other) or not hasattr(other, '__rmul__') :
        222             return N.dot(self, other)


        <__array_function__ internals> in dot(*args, **kwargs)


        ValueError: shapes (5,5) and (6,1) not aligned: 5 (dim 1) != 6 (dim 0)
```

```
In [102]: C = matrix([[1j, 2j], [3j, 4j]])
          C

Out[102]: matrix([[0.+1.j, 0.+2.j],
                  [0.+3.j, 0.+4.j]])

In [103]: conjugate(C)

Out[103]: matrix([[0.-1.j, 0.-2.j],
                  [0.-3.j, 0.-4.j]])

In [104]: C.H

Out[104]: matrix([[0.-1.j, 0.-3.j],
                  [0.-2.j, 0.-4.j]])

In [105]: real(C) # same as: C.real

Out[105]: matrix([[0., 0.],
                  [0., 0.]])

In [106]: imag(C) # same as: C.imag

Out[106]: matrix([[1., 2.],
                  [3., 4.]])

In [107]: angle(C+1) # heads up MATLAB Users, angle is used instead of arg

Out[107]: matrix([[0.78539816, 1.10714872],
                  [1.24904577, 1.32581766]])

In [108]: abs(C)

Out[108]: matrix([[1., 2.],
                  [3., 4.]])

In [109]: linalg.inv(C) # equivalent to C.I

Out[109]: matrix([[0.+2.j , 0.-1.j ],
                  [0.-1.5j, 0.+0.5j]])

In [110]: C.I * C

Out[110]: matrix([[1.00000000e+00+0.j, 0.00000000e+00+0.j],
                  [1.11022302e-16+0.j, 1.00000000e+00+0.j]])

In [111]: linalg.det(C)

Out[111]: (2.0000000000000004+0j)
```

```
In [112]: linalg.det(C.I)

Out[112]: (0.49999999999999967+0j)

In [113]: # reminder, the tempeature dataset is stored in the data variable:
          shape(data)

Out[113]: (77431, 7)

In [114]: # the temperature data is in column 3
          mean(data[:,3])

Out[114]: 6.197109684751585

In [115]: std(data[:,3]), var(data[:,3])

Out[115]: (8.282271621340573, 68.59602320966341)

In [116]: # lowest daily average temperature
          data[:,3].min()

Out[116]: -25.8

In [117]: # highest daily average temperature
          data[:,3].max()

Out[117]: 28.3

In [118]: d = arange(0, 10)
          d

Out[118]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [119]: # sum up all elements
          sum(d)

Out[119]: 45

In [120]: # product of all elements
          prod(d+1)

Out[120]: 3628800

In [121]: # cummulative sum
          cumsum(d)

Out[121]: array([ 0,  1,  3,  6, 10, 15, 21, 28, 36, 45], dtype=int32)

In [122]: # cummulative product
          cumprod(d+1)

Out[122]: array([      1,       2,       6,      24,     120,     720,    5040,
                  40320,  362880, 3628800], dtype=int32)

In [123]: # same as: diag(A).sum()
          trace(A)

Out[123]: 110

In [124]: !head -n 3 stockholm_td_adj.dat
```

```
'head' is not recognized as an internal or external command,
operable program or batch file.
```

```
In [125]: unique(data[:,1]) # the month column takes values from 1 to 12
```

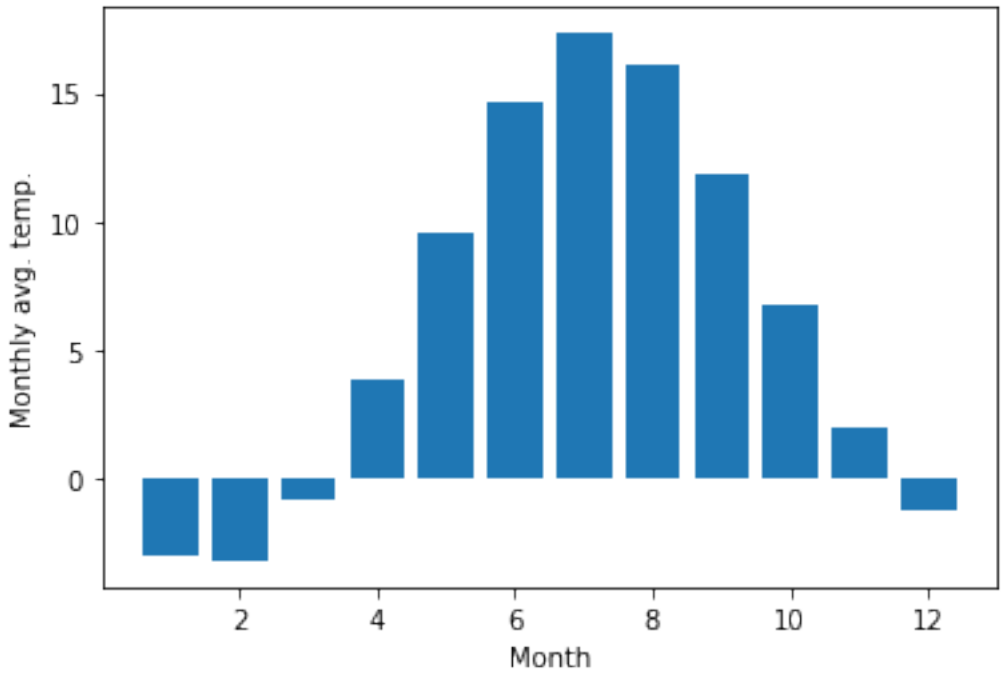
```
Out[125]: array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.])
```

```
In [126]: mask_feb = data[:,1] == 2
```

```
In [127]: # the temperature data is in column 3
          mean(data[mask_feb,3])
```

```
Out[127]: -3.212109570736596
```

```
In [128]: months = arange(1,13)
          monthly_mean = [mean(data[data[:,1] == month, 3]) for month in months]

          fig, ax = plt.subplots()
          ax.bar(months, monthly_mean)
          ax.set_xlabel("Month")
          ax.set_ylabel("Monthly avg. temp.");
```




```
In [129]: m = random.rand(3,3)
          m
```

```
Out[129]: array([[0.89356485, 0.69778664, 0.60680164],
                 [0.02668988, 0.25966432, 0.50842785],
                 [0.90148382, 0.65283816, 0.71717646]])
```

```
In [130]: # global max
          m.max()

Out[130]: 0.9014838154382184

In [131]: # max in each column
          m.max(axis=0)

Out[131]: array([0.90148382, 0.69778664, 0.71717646])

In [132]: # max in each row
          m.max(axis=1)

Out[132]: array([0.89356485, 0.50842785, 0.90148382])

In [133]: A

Out[133]: array([[ 0,  1,  2,  3,  4],
                 [10, 11, 12, 13, 14],
                 [20, 21, 22, 23, 24],
                 [30, 31, 32, 33, 34],
                 [40, 41, 42, 43, 44]])

In [134]: n, m = A.shape

In [135]: B = A.reshape((1,n*m))
          B

Out[135]: array([[ 0,  1,  2,  3,  4, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24, 30,
                  31, 32, 33, 34, 40, 41, 42, 43, 44]])

In [136]: B[0,0:5] = 5 # modify the array

          B

Out[136]: array([[ 5,  5,  5,  5,  5, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24, 30,
                  31, 32, 33, 34, 40, 41, 42, 43, 44]])

In [137]: A # and the original variable is also changed. B is only a different view of the same data

Out[137]: array([[ 5,  5,  5,  5,  5],
                 [10, 11, 12, 13, 14],
                 [20, 21, 22, 23, 24],
                 [30, 31, 32, 33, 34],
                 [40, 41, 42, 43, 44]])

In [138]: B = A.flatten()

          B

Out[138]: array([ 5,  5,  5,  5,  5, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24, 30, 31,
                 32, 33, 34, 40, 41, 42, 43, 44])

In [139]: B[0:5] = 10

          B

Out[139]: array([10, 10, 10, 10, 10, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24, 30, 31,
                 32, 33, 34, 40, 41, 42, 43, 44])
```

```
In [140]: A # now A has not changed, because B's data is a copy of A's, not refering to the same data

Out[140]: array([[ 5,  5,  5,  5,  5],
                 [10, 11, 12, 13, 14],
                 [20, 21, 22, 23, 24],
                 [30, 31, 32, 33, 34],
                 [40, 41, 42, 43, 44]])

In [141]: v = array([1,2,3])

In [142]: shape(v)

Out[142]: (3,)

In [143]: # make a column matrix of the vector v
          v[:, newaxis]

Out[143]: array([[1],
                 [2],
                 [3]])

In [144]: # column matrix
          v[:,newaxis].shape

Out[144]: (3, 1)

In [145]: # row matrix
          v[newaxis,:].shape

Out[145]: (1, 3)

In [146]: a = array([[1, 2], [3, 4]])

In [147]: # repeat each element 3 times
          repeat(a, 3)

Out[147]: array([1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4])

In [148]: # tile the matrix 3 times
          tile(a, 3)

Out[148]: array([[1, 2, 1, 2, 1, 2],
                 [3, 4, 3, 4, 3, 4]])

In [149]: b = array([[5, 6]])

In [150]: concatenate((a, b), axis=0)

Out[150]: array([[1, 2],
                 [3, 4],
                 [5, 6]])

In [151]: concatenate((a, b.T), axis=1)

Out[151]: array([[1, 2, 5],
                 [3, 4, 6]])

In [152]: vstack((a,b))
```

```
Out[152]: array([[1, 2],
                 [3, 4],
                 [5, 6]])

In [153]: hstack((a,b.T))

Out[153]: array([[1, 2, 5],
                 [3, 4, 6]])

In [154]: A = array([[1, 2], [3, 4]])

          A

Out[154]: array([[1, 2],
                 [3, 4]])

In [155]: # now B is referring to the same array data as A
          B = A

In [156]: # changing B affects A
          B[0,0] = 10

          B

Out[156]: array([[10,  2],
                 [ 3,  4]])

In [157]: A

Out[157]: array([[10,  2],
                 [ 3,  4]])

In [158]: B = copy(A)

In [159]: # now, if we modify B, A is not affected
          B[0,0] = -5

          B

Out[159]: array([[-5,  2],
                 [ 3,  4]])

In [160]: A

Out[160]: array([[10,  2],
                 [ 3,  4]])

In [161]: v = array([1,2,3,4])

          for element in v:
              print(element)

1
2
3
4
```

```
In [162]: M = array([[1,2], [3,4]])

          for row in M:
              print("row", row)

              for element in row:
                  print(element)
```

```
row [1 2]
1
2
row [3 4]
3
4
```

```
In [163]: for row_idx, row in enumerate(M):
              print("row_idx", row_idx, "row", row)

              for col_idx, element in enumerate(row):
                  print("col_idx", col_idx, "element", element)

                  # update the matrix M: square each element
                  M[row_idx, col_idx] = element ** 2
```

```
row_idx 0 row [1 2]
col_idx 0 element 1
col_idx 1 element 2
row_idx 1 row [3 4]
col_idx 0 element 3
col_idx 1 element 4
```

```
In [164]: # each element in M is now squared
          M
```

```
Out[164]: array([[ 1,  4],
                 [ 9, 16]])
```

```
In [165]: def Theta(x):
              """
              Scalar implemenation of the Heaviside step function.
              """
              if x >= 0:
                  return 1
              else:
                  return 0
```

```
In [166]: Theta(array([-3,-2,-1,0,1,2,3]))
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-166-2cb2062a7e18> in <module>
```

```
----> 1 Theta(array([-3,-2,-1,0,1,2,3]))


<ipython-input-165-f72d7f42be84> in Theta(x)
      3     Scalar implemenation of the Heaviside step function.
      4     """
----> 5     if x >= 0:
      6         return 1
      7     else:


ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or
```

```
In [167]: Theta_vec = vectorize(Theta)

In [168]: Theta_vec(array([-3,-2,-1,0,1,2,3]))

Out[168]: array([0, 0, 0, 1, 1, 1, 1])

In [169]: def Theta(x):
              """
              Vector-aware implemenation of the Heaviside step function.
              """
              return 1 * (x >= 0)

In [170]: Theta(array([-3,-2,-1,0,1,2,3]))

Out[170]: array([0, 0, 0, 1, 1, 1, 1])

In [171]: # still works for scalars as well
          Theta(-1.2), Theta(2.6)

Out[171]: (0, 1)

In [172]: M

Out[172]: array([[ 1,  4],
                 [ 9, 16]])

In [173]: if (M > 5).any():
              print("at least one element in M is larger than 5")
          else:
              print("no element in M is larger than 5")

at least one element in M is larger than 5


In [174]: if (M > 5).all():
              print("all elements in M are larger than 5")
          else:
              print("all elements in M are not larger than 5")

all elements in M are not larger than 5


In [175]: M.dtype
```

```
Out[175]: dtype('int32')

In [176]: M2 = M.astype(float)

          M2

Out[176]: array([[ 1.,  4.],
                 [ 9., 16.]])

In [177]: M2.dtype

Out[177]: dtype('float64')

In [178]: M3 = M.astype(bool)

          M3

Out[178]: array([[ True,  True],
                 [ True,  True]])

In [1]: %reload_ext version_information

        %version_information numpy, scipy, matplot

Out[1]:

In [ ]:
```

In [1]:
```
# what is this line all about? Answer in lecture 4
%matplotlib inline
import matplotlib.pyplot as plt
from IPython.display import Image
```

In [2]:
```
from scipy import *
```

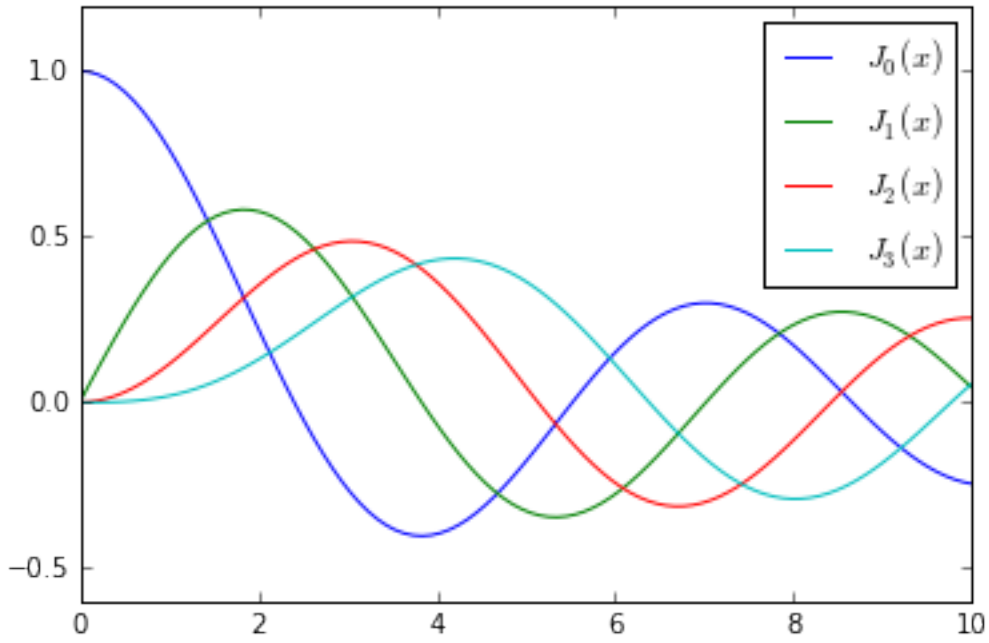
In [3]:
```
import scipy.linalg as la
```

In [4]:
```
#
# The scipy.special module includes a large number of Bessel-functions
# Here we will use the functions jn and yn, which are the Bessel functions
# of the first and second kind and real-valued order. We also include the
# function jn_zeros and yn_zeros that gives the zeroes of the functions jn
# and yn.
#
from scipy.special import jn, yn, jn_zeros, yn_zeros
```

In [5]:
```
n = 0    # order
x = 0.0

# Bessel function of first kind
print "J_%d(%f) = %f" % (n, x, jn(n, x))

x = 1.0
# Bessel function of second kind
print "Y_%d(%f) = %f" % (n, x, yn(n, x))
```

```
J_0(0.000000) = 1.000000
Y_0(1.000000) = 0.088257
```

In [6]:
```
x = linspace(0, 10, 100)

fig, ax = plt.subplots()
for n in range(4):
    ax.plot(x, jn(n, x), label=r"$J_%d(x)$" % n)
ax.legend();
```




```
In [7]: # zeros of Bessel functions
        n = 0 # order
        m = 4 # number of roots to compute
        jn_zeros(n, m)
```

```
Out[7]: array([  2.40482556,   5.52007811,   8.65372791,  11.79153444])
```

```
In [8]: from scipy.integrate import quad, dblquad, tplquad
```

```
In [9]: # define a simple function for the integrand
        def f(x):
            return x
```

```
In [10]: x_lower = 0 # the lower limit of x
         x_upper = 1 # the upper limit of x

         val, abserr = quad(f, x_lower, x_upper)

         print "integral value =", val, ", absolute error =", abserr
```

```
integral value = 0.5 , absolute error = 5.55111512313e-15
```

```
In [11]: def integrand(x, n):
             """
             Bessel function of first kind and order n.
             """
             return jn(n, x)


         x_lower = 0  # the lower limit of x
```

```
        x_upper = 10 # the upper limit of x

        val, abserr = quad(integrand, x_lower, x_upper, args=(3,))

        print val, abserr
```

```
0.736675137081 9.3891268825e-13
```

```
In [12]: val, abserr = quad(lambda x: exp(-x ** 2), -Inf, Inf)

         print "numerical  =", val, abserr

         analytical = sqrt(pi)
         print "analytical =", analytical
```

```
numerical  = 1.77245385091 1.42026367809e-08
analytical = 1.77245385091
```

```
In [13]: def integrand(x, y):
             return exp(-x**2-y**2)

         x_lower = 0
         x_upper = 10
         y_lower = 0
         y_upper = 10

         val, abserr = dblquad(integrand, x_lower, x_upper, lambda x : y_lower, lambda x: y_upper)

         print val, abserr
```

```
0.785398163397 1.63822994214e-13
```

```
In [14]: from scipy.integrate import odeint, ode

In [15]: Image(url='http://upload.wikimedia.org/wikipedia/commons/c/c9/Double-compound-pendulum-dimensi
```

```
Out[15]: <IPython.core.display.Image object>
```

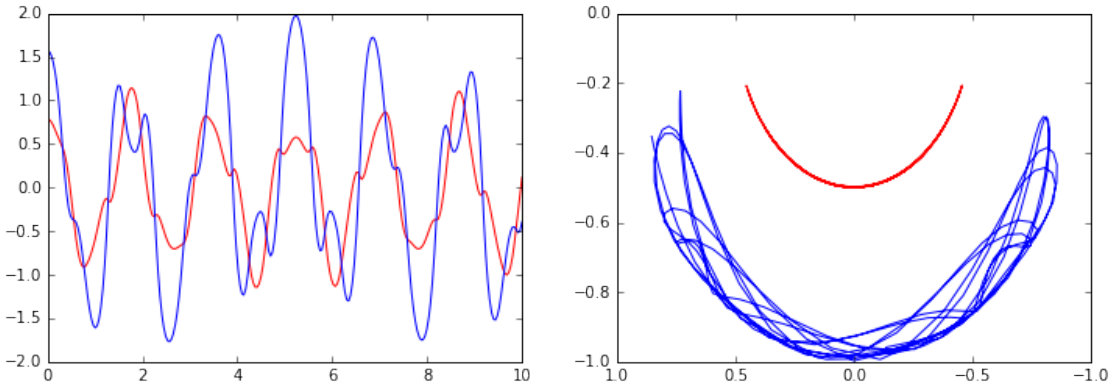
```
In [16]: g = 9.82
         L = 0.5
         m = 0.1

         def dx(x, t):
             """
             The right-hand side of the pendulum ODE
             """
             x1, x2, x3, x4 = x[0], x[1], x[2], x[3]

             dx1 = 6.0/(m*L**2) * (2 * x3 - 3 * cos(x1-x2) * x4)/(16 - 9 * cos(x1-x2)**2)
             dx2 = 6.0/(m*L**2) * (8 * x4 - 3 * cos(x1-x2) * x3)/(16 - 9 * cos(x1-x2)**2)
             dx3 = -0.5 * m * L**2 * ( dx1 * dx2 * sin(x1-x2) + 3 * (g/L) * sin(x1))
             dx4 = -0.5 * m * L**2 * (-dx1 * dx2 * sin(x1-x2) + (g/L) * sin(x2))

             return [dx1, dx2, dx3, dx4]
```

In [17]:
```
# choose an initial state
x0 = [pi/4, pi/2, 0, 0]
```

In [18]:
```
# time coodinate to solve the ODE for: from 0 to 10 seconds
t = linspace(0, 10, 250)
```

In [19]:
```
# solve the ODE problem
x = odeint(dx, x0, t)
```

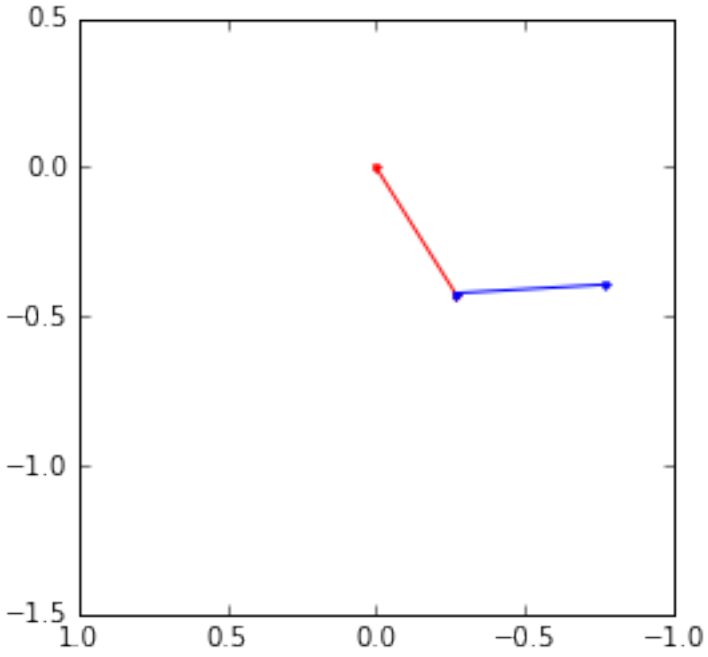
In [20]:
```
# plot the angles as a function of time

fig, axes = plt.subplots(1,2, figsize=(12,4))
axes[0].plot(t, x[:, 0], 'r', label="theta1")
axes[0].plot(t, x[:, 1], 'b', label="theta2")


x1 = + L * sin(x[:, 0])
y1 = - L * cos(x[:, 0])

x2 = x1 + L * sin(x[:, 1])
y2 = y1 - L * cos(x[:, 1])

axes[1].plot(x1, y1, 'r', label="pendulum1")
axes[1].plot(x2, y2, 'b', label="pendulum2")
axes[1].set_ylim([-1, 0])
axes[1].set_xlim([1, -1]);
```

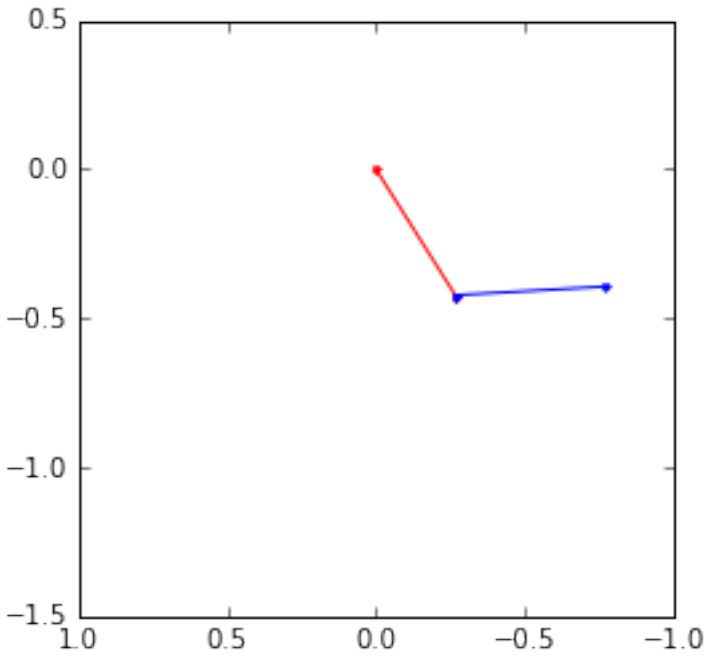


In [21]:
```
from IPython.display import display, clear_output
import time
```

In [22]:
```
fig, ax = plt.subplots(figsize=(4,4))

for t_idx, tt in enumerate(t[:200]):

    x1 = + L * sin(x[t_idx, 0])
    y1 = - L * cos(x[t_idx, 0])

    x2 = x1 + L * sin(x[t_idx, 1])
    y2 = y1 - L * cos(x[t_idx, 1])
```

```
ax.cla()
ax.plot([0, x1], [0, y1], 'r.-')
ax.plot([x1, x2], [y1, y2], 'b.-')
ax.set_ylim([-1.5, 0.5])
ax.set_xlim([1, -1])

clear_output()
display(fig)

time.sleep(0.1)
```





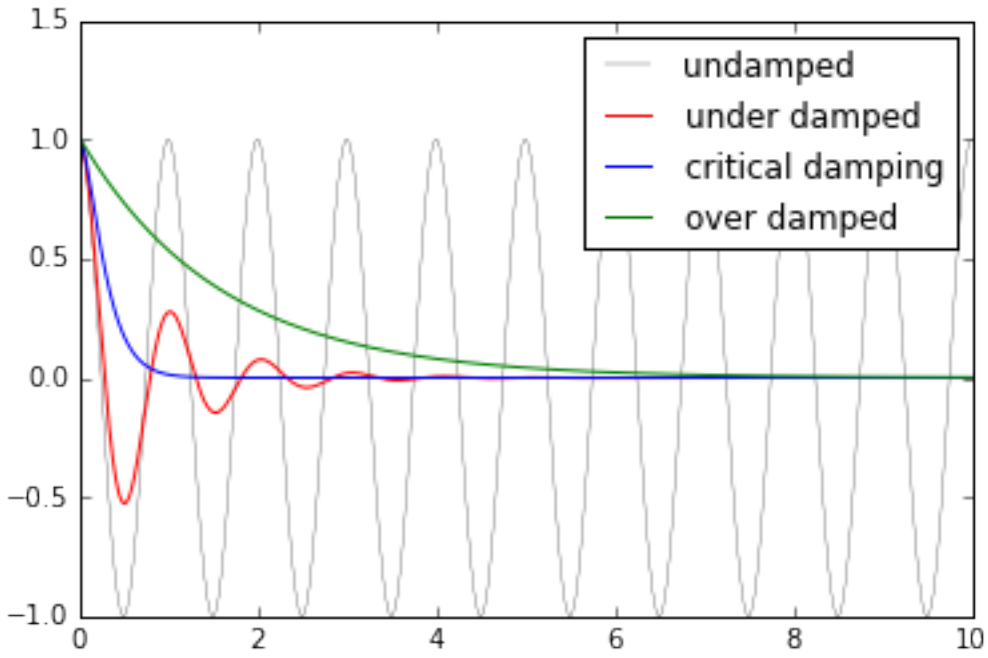

```
In [23]: def dy(y, t, zeta, w0):
             """
             The right-hand side of the damped oscillator ODE
             """
             x, p = y[0], y[1]

             dx = p
             dp = -2 * zeta * w0 * p - w0**2 * x

             return [dx, dp]
```

```
In [24]: # initial state:
         y0 = [1.0, 0.0]
```

```
In [25]: # time coodinate to solve the ODE for
         t = linspace(0, 10, 1000)
         w0 = 2*pi*1.0
```

```
In [26]: # solve the ODE problem for three different values of the damping ratio

         y1 = odeint(dy, y0, t, args=(0.0, w0)) # undamped
         y2 = odeint(dy, y0, t, args=(0.2, w0)) # under damped
         y3 = odeint(dy, y0, t, args=(1.0, w0)) # critial damping
         y4 = odeint(dy, y0, t, args=(5.0, w0)) # over damped
```

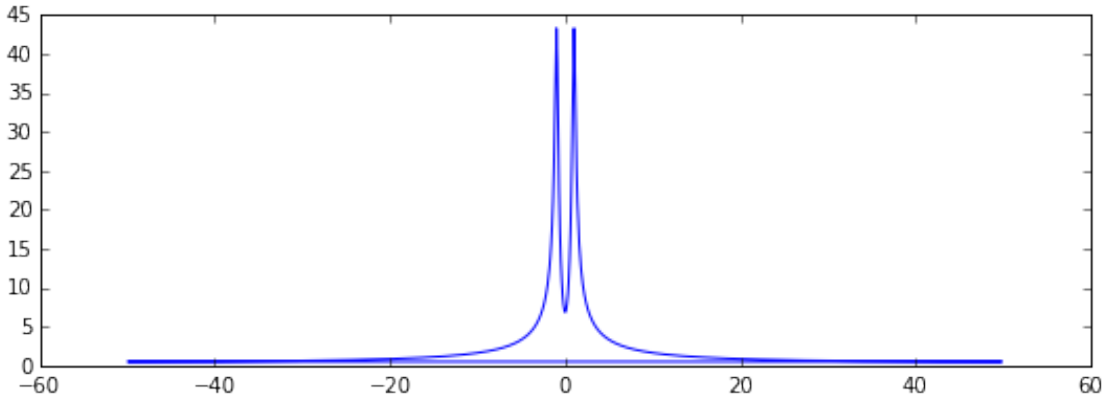
```
In [27]: fig, ax = plt.subplots()
         ax.plot(t, y1[:,0], 'k', label="undamped", linewidth=0.25)
         ax.plot(t, y2[:,0], 'r', label="under damped")
         ax.plot(t, y3[:,0], 'b', label=r"critical damping")
         ax.plot(t, y4[:,0], 'g', label="over damped")
         ax.legend();
```



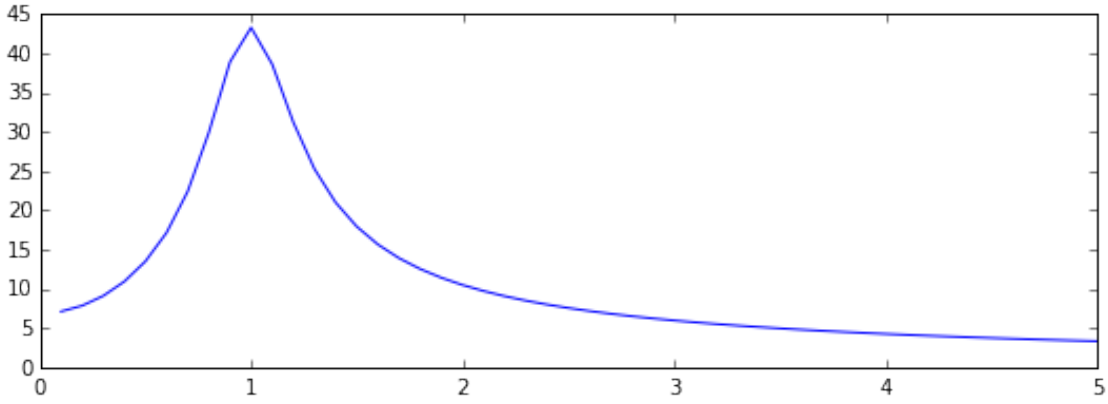
In [28]:
```
from numpy.fft import fftfreq
from scipy.fftpack import *
```

In [29]:
```
N = len(t)
dt = t[1]-t[0]

# calculate the fast fourier transform
# y2 is the solution to the under-damped oscillator from the previous section
F = fft(y2[:,0])

# calculate the frequencies for the components in F
w = fftfreq(N, dt)
```

In [30]:
```
fig, ax = plt.subplots(figsize=(9,3))
ax.plot(w, abs(F));
```



```
In [31]: indices = where(w > 0) # select only indices for elements that corresponds to positive frequen
         w_pos = w[indices]
         F_pos = F[indices]

In [32]: fig, ax = plt.subplots(figsize=(9,3))
         ax.plot(w_pos, abs(F_pos))
         ax.set_xlim(0, 5);
```




```
In [33]: from scipy.linalg import *

In [34]: A = array([[1,2,3], [4,5,6], [7,8,9]])
         b = array([1,2,3])

In [35]: x = solve(A, b)

         x

Out[35]: array([-0.33333333,  0.66666667,  0.        ])

In [36]: # check
         dot(A, x) - b

Out[36]: array([ -1.11022302e-16,   0.00000000e+00,   0.00000000e+00])

In [37]: A = rand(3,3)
         B = rand(3,3)

In [38]: X = solve(A, B)

In [39]: X

Out[39]: array([[ 1.19168749,  1.34543171,  0.38437594],
                [-0.88153715, -3.22735597,  0.66370273],
                [ 0.10044006,  1.0465058 ,  0.39801748]])

In [40]: # check
         norm(dot(A, X) - B)
```

```
Out[40]: 2.0014830212433605e-16

In [41]: evals = eigvals(A)

In [42]: evals

Out[42]: array([ 1.08466629+0.j,  0.33612878+0.j, -0.28229973+0.j])

In [43]: evals, evecs = eig(A)

In [44]: evals

Out[44]: array([ 1.08466629+0.j,  0.33612878+0.j, -0.28229973+0.j])

In [45]: evecs

Out[45]: array([[-0.20946865, -0.48428024, -0.14392087],
                [-0.79978578,  0.8616452 , -0.79527482],
                [-0.56255275,  0.15178997,  0.58891829]])

In [46]: n = 1

         norm(dot(A, evecs[:,n]) - evals[n] * evecs[:,n])

Out[46]: 3.243515426387745e-16

In [47]: # the matrix inverse
         inv(A)

Out[47]: array([[ 2.0031935 , -0.63411453,  0.49891784],
                [-4.63643938, -0.2212669 ,  3.35170585],
                [ 1.06421936,  1.37366073, -1.42726809]])

In [48]: # determinant
         det(A)

Out[48]: -0.10292296739753022

In [49]: # norms of various orders
         norm(A, ord=2), norm(A, ord=Inf)

Out[49]: (1.3060382297688262, 1.591998214728641)

In [50]: from scipy.sparse import *

In [51]: # dense matrix
         M = array([[1,0,0,0], [0,3,0,0], [0,1,1,0], [1,0,0,1]]); M

Out[51]: array([[1, 0, 0, 0],
                [0, 3, 0, 0],
                [0, 1, 1, 0],
                [1, 0, 0, 1]])

In [52]: # convert from dense to sparse
         A = csr_matrix(M); A

Out[52]: <4x4 sparse matrix of type '<type 'numpy.int64'>'
                 with 6 stored elements in Compressed Sparse Row format>
```

```
In [53]: # convert from sparse to dense
         A.todense()

Out[53]: matrix([[1, 0, 0, 0],
                 [0, 3, 0, 0],
                 [0, 1, 1, 0],
                 [1, 0, 0, 1]])

In [54]: A = lil_matrix((4,4)) # empty 4x4 sparse matrix
         A[0,0] = 1
         A[1,1] = 3
         A[2,2] = A[2,1] = 1
         A[3,3] = A[3,0] = 1
         A

Out[54]: <4x4 sparse matrix of type '<type 'numpy.float64'>'
                 with 6 stored elements in LInked List format>

In [55]: A.todense()

Out[55]: matrix([[ 1.,  0.,  0.,  0.],
                 [ 0.,  3.,  0.,  0.],
                 [ 0.,  1.,  1.,  0.],
                 [ 1.,  0.,  0.,  1.]])

In [56]: A

Out[56]: <4x4 sparse matrix of type '<type 'numpy.float64'>'
                 with 6 stored elements in LInked List format>

In [57]: A = csr_matrix(A); A

Out[57]: <4x4 sparse matrix of type '<type 'numpy.float64'>'
                 with 6 stored elements in Compressed Sparse Row format>

In [58]: A = csc_matrix(A); A

Out[58]: <4x4 sparse matrix of type '<type 'numpy.float64'>'
                 with 6 stored elements in Compressed Sparse Column format>

In [59]: A.todense()

Out[59]: matrix([[ 1.,  0.,  0.,  0.],
                 [ 0.,  3.,  0.,  0.],
                 [ 0.,  1.,  1.,  0.],
                 [ 1.,  0.,  0.,  1.]])

In [60]: (A * A).todense()

Out[60]: matrix([[ 1.,  0.,  0.,  0.],
                 [ 0.,  9.,  0.,  0.],
                 [ 0.,  4.,  1.,  0.],
                 [ 2.,  0.,  0.,  1.]])

In [61]: A.todense()

Out[61]: matrix([[ 1.,  0.,  0.,  0.],
                 [ 0.,  3.,  0.,  0.],
                 [ 0.,  1.,  1.,  0.],
                 [ 1.,  0.,  0.,  1.]])
```

```
In [62]: A.dot(A).todense()
```

```
Out[62]: matrix([[ 1.,  0.,  0.,  0.],
                 [ 0.,  9.,  0.,  0.],
                 [ 0.,  4.,  1.,  0.],
                 [ 2.,  0.,  0.,  1.]])
```

```
In [63]: v = array([1,2,3,4])[:,newaxis]; v
```

```
Out[63]: array([[1],
                [2],
                [3],
                [4]])
```

```
In [64]: # sparse matrix - dense vector multiplication
         A * v
```

```
Out[64]: array([[ 1.],
                [ 6.],
                [ 5.],
                [ 5.]])
```

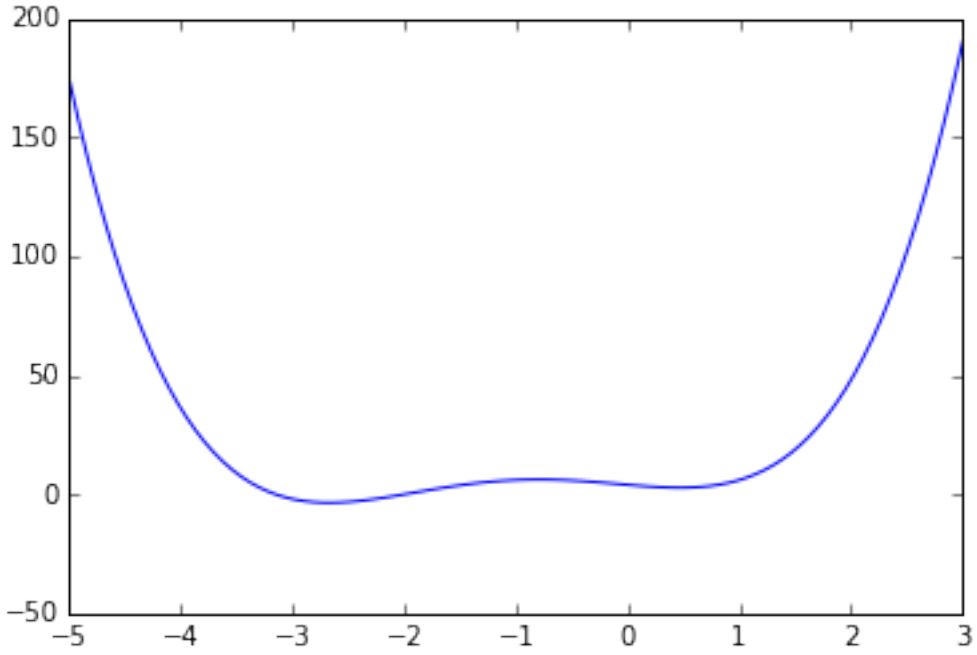
```
In [65]: # same result with dense matrix - dense vector multiplcation
         A.todense() * v
```

```
Out[65]: matrix([[ 1.],
                 [ 6.],
                 [ 5.],
                 [ 5.]])
```

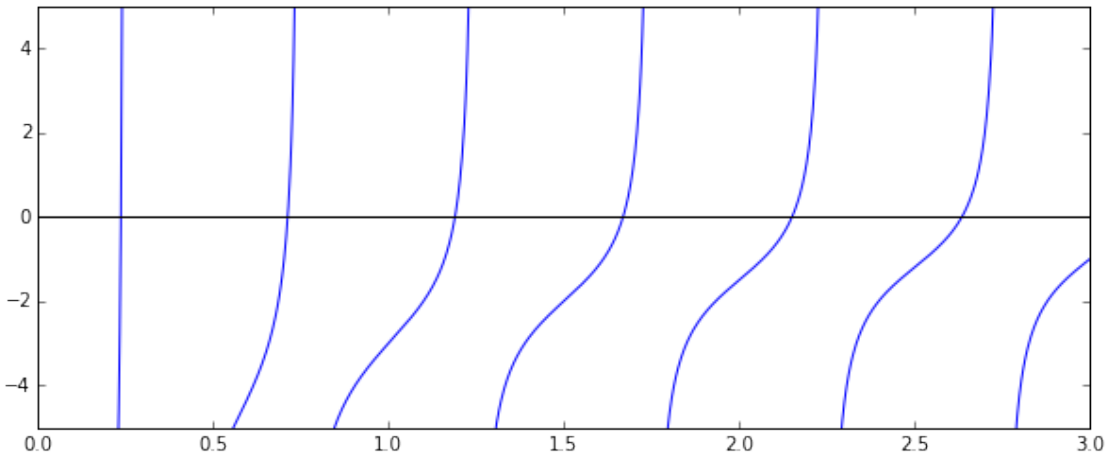
```
In [66]: from scipy import optimize
```

```
In [67]: def f(x):
             return 4*x**3 + (x-2)**2 + x**4
```

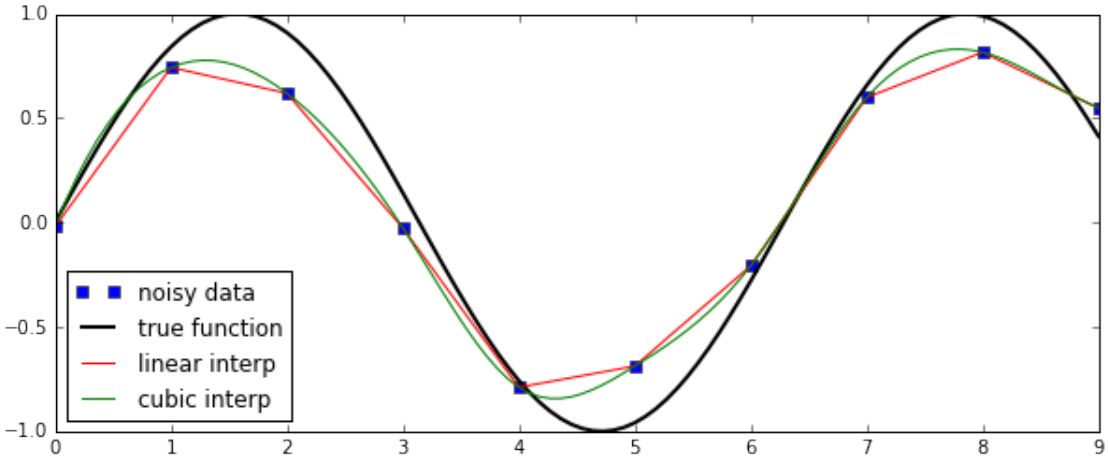
```
In [68]: fig, ax  = plt.subplots()
         x = linspace(-5, 3, 100)
         ax.plot(x, f(x));
```



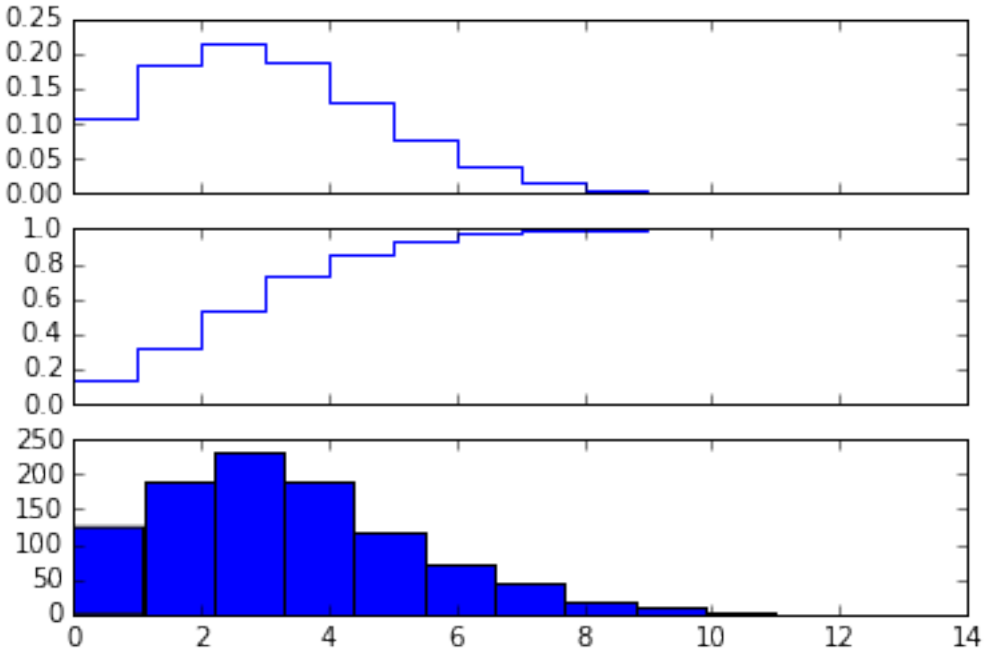
```
In [69]: x_min = optimize.fmin_bfgs(f, -2)
         x_min

Optimization terminated successfully.
         Current function value: -3.506641
         Iterations: 6
         Function evaluations: 30
         Gradient evaluations: 10


Out[69]: array([-2.67298164])

In [70]: optimize.fmin_bfgs(f, 0.5)

Optimization terminated successfully.
         Current function value: 2.804988
         Iterations: 3
         Function evaluations: 15
         Gradient evaluations: 5


Out[70]: array([ 0.46961745])

In [71]: optimize.brent(f)

Out[71]: 0.46961743402759754

In [72]: optimize.fminbound(f, -4, 2)

Out[72]: -2.6729822917513886

In [73]: omega_c = 3.0
         def f(omega):
             # a transcendental equation: resonance frequencies of a low-Q SQUID terminated microwave r
             return tan(2*pi*omega) - omega_c/omega

In [74]: fig, ax  = plt.subplots(figsize=(10,4))
         x = linspace(0, 3, 1000)
         y = f(x)
         mask = where(abs(y) > 50)
         x[mask] = y[mask] = NaN # get rid of vertical line when the function flip sign
         ax.plot(x, y)
         ax.plot([0, 3], [0, 0], 'k')
         ax.set_ylim(-5,5);

/Users/rob/miniconda/envs/py27-spl/lib/python2.7/site-packages/IPython/kernel/__main__.py:4: RuntimeWarni
```



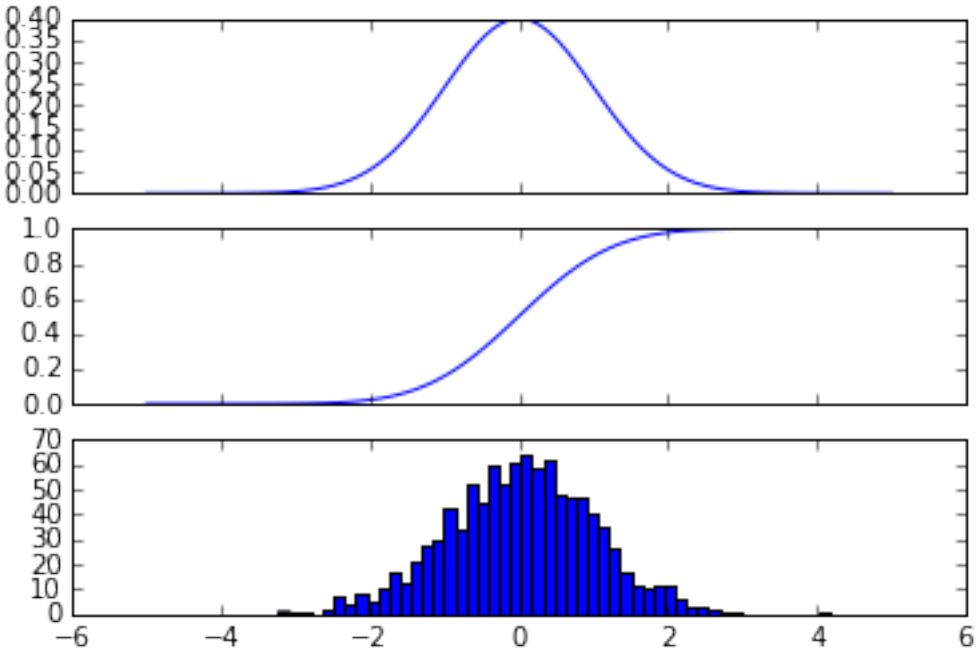

```
In [75]: optimize.fsolve(f, 0.1)

Out[75]: array([ 0.23743014])

In [76]: optimize.fsolve(f, 0.6)

Out[76]: array([ 0.71286972])

In [77]: optimize.fsolve(f, 1.1)

Out[77]: array([ 1.18990285])

In [78]: from scipy.interpolate import *

In [79]: def f(x):
             return sin(x)

In [80]: n = arange(0, 10)
         x = linspace(0, 9, 100)

         y_meas = f(n) + 0.1 * randn(len(n)) # simulate measurement with noise
         y_real = f(x)

         linear_interpolation = interp1d(n, y_meas)
         y_interp1 = linear_interpolation(x)

         cubic_interpolation = interp1d(n, y_meas, kind='cubic')
         y_interp2 = cubic_interpolation(x)

In [81]: fig, ax = plt.subplots(figsize=(10,4))
         ax.plot(n, y_meas, 'bs', label='noisy data')
         ax.plot(x, y_real, 'k', lw=2, label='true function')
         ax.plot(x, y_interp1, 'r', label='linear interp')
         ax.plot(x, y_interp2, 'g', label='cubic interp')
         ax.legend(loc=3);
```




```
In [82]: from scipy import stats

In [83]: # create a (discreet) random variable with poissionian distribution

         X = stats.poisson(3.5) # photon distribution for a coherent state with n=3.5 photons

In [84]: n = arange(0,15)

         fig, axes = plt.subplots(3,1, sharex=True)

         # plot the probability mass function (PMF)
         axes[0].step(n, X.pmf(n))

         # plot the commulative distribution function (CDF)
         axes[1].step(n, X.cdf(n))

         # plot histogram of 1000 random realizations of the stochastic variable X
         axes[2].hist(X.rvs(size=1000));
```




```
In [85]: # create a (continous) random variable with normal distribution
         Y = stats.norm()

In [86]: x = linspace(-5,5,100)

         fig, axes = plt.subplots(3,1, sharex=True)

         # plot the probability distribution function (PDF)
         axes[0].plot(x, Y.pdf(x))

         # plot the commulative distributin function (CDF)
         axes[1].plot(x, Y.cdf(x));

         # plot histogram of 1000 random realizations of the stochastic variable Y
         axes[2].hist(Y.rvs(size=1000), bins=50);
```




```
In [87]: X.mean(), X.std(), X.var() # poission distribution

Out[87]: (3.5, 1.8708286933869707, 3.5)

In [88]: Y.mean(), Y.std(), Y.var() # normal distribution

Out[88]: (0.0, 1.0, 1.0)

In [89]: t_statistic, p_value = stats.ttest_ind(X.rvs(size=1000), X.rvs(size=1000))

         print "t-statistic =", t_statistic
         print "p-value =", p_value

t-statistic = -0.901953297251
p-value = 0.367190391714


In [90]: stats.ttest_1samp(Y.rvs(size=1000), 0.1)

Out[90]: Ttest_1sampResult(statistic=-3.1644288210071765, pvalue=0.0016008455559249511)

In [91]: Y.mean()

Out[91]: 0.0

In [92]: stats.ttest_1samp(Y.rvs(size=1000), Y.mean())

Out[92]: Ttest_1sampResult(statistic=2.2098772438652992, pvalue=0.027339807364469011)

In [93]: %reload_ext version_information

         %version_information numpy, matplotlib, scipy

Out[93]:
```

In [1]:
```
# This line configures matplotlib to show figures embedded in the notebook,
# instead of opening a new window for each figure. More about that later.
# If you are using an old version of IPython, try using '%pylab inline' instead.
%matplotlib inline
```

In [2]:
```
from pylab import *
```

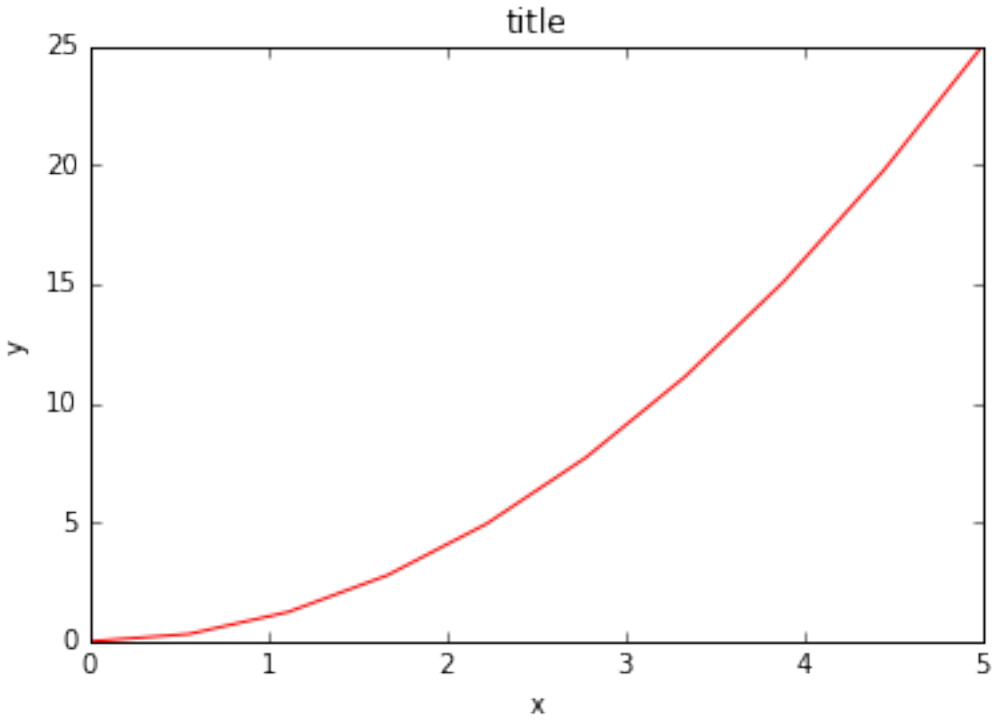
In [3]:
```
import matplotlib
import matplotlib.pyplot as plt
```

In [4]:
```
import numpy as np
```

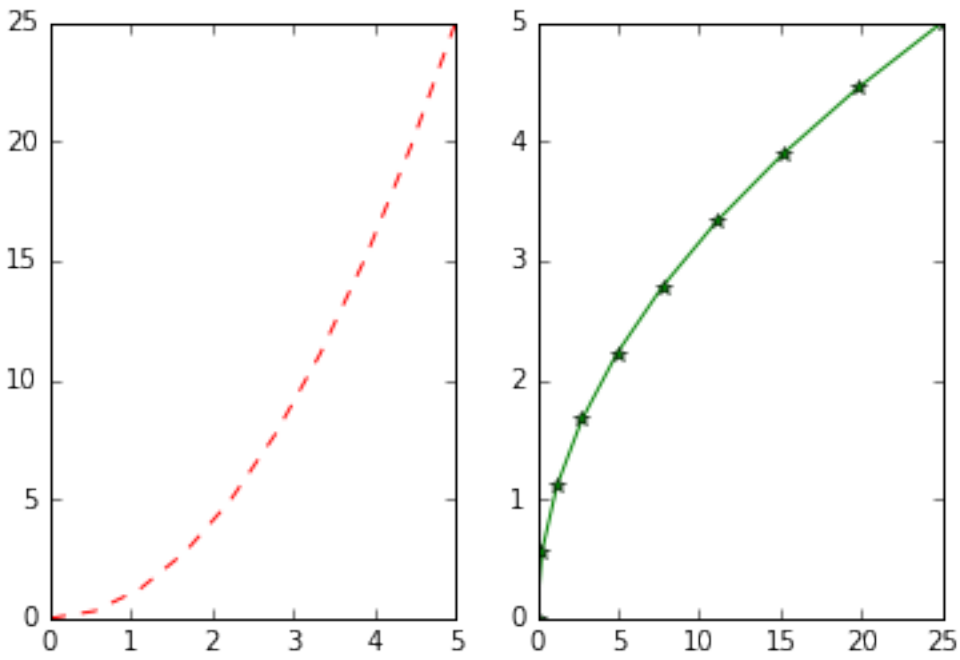
In [5]:
```
from pylab import *
```

In [6]:
```
x = np.linspace(0, 5, 10)
y = x ** 2
```

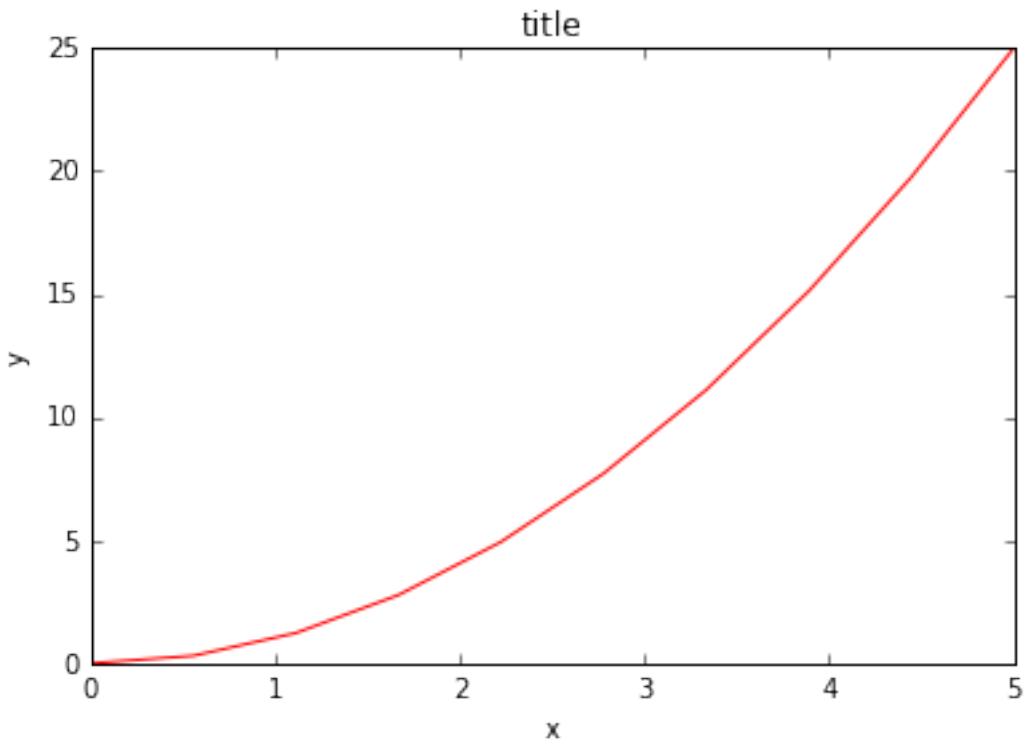
In [7]:
```
figure()
plot(x, y, 'r')
xlabel('x')
ylabel('y')
title('title')
show()
```



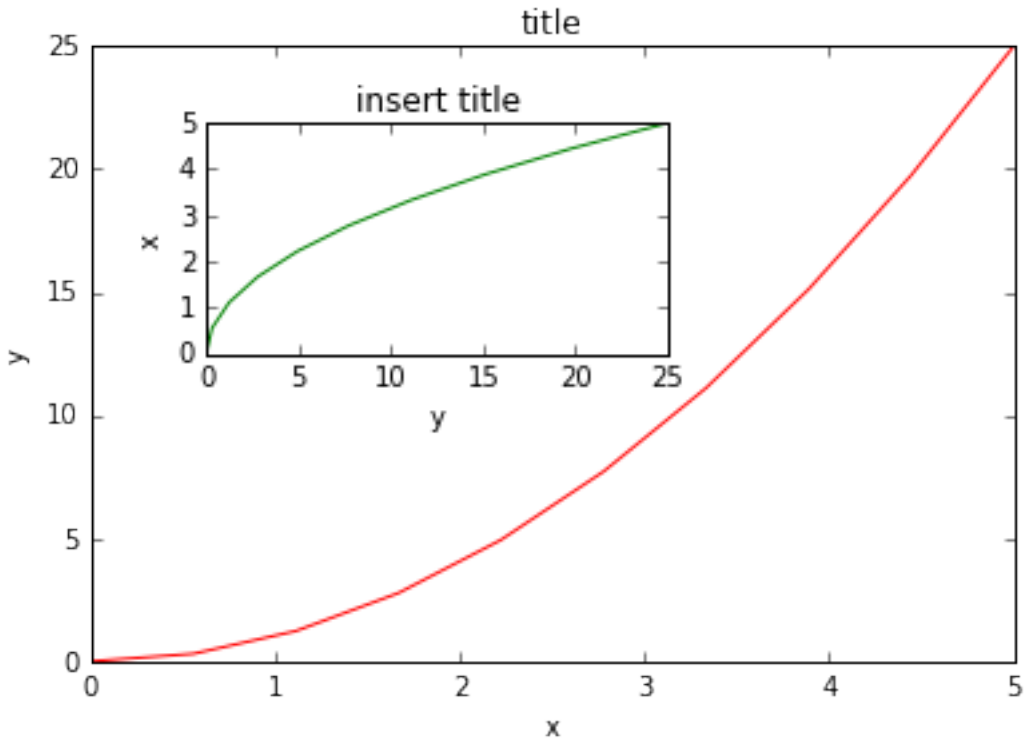
In [8]:
```
subplot(1,2,1)
plot(x, y, 'r--')
subplot(1,2,2)
plot(y, x, 'g*-');
```



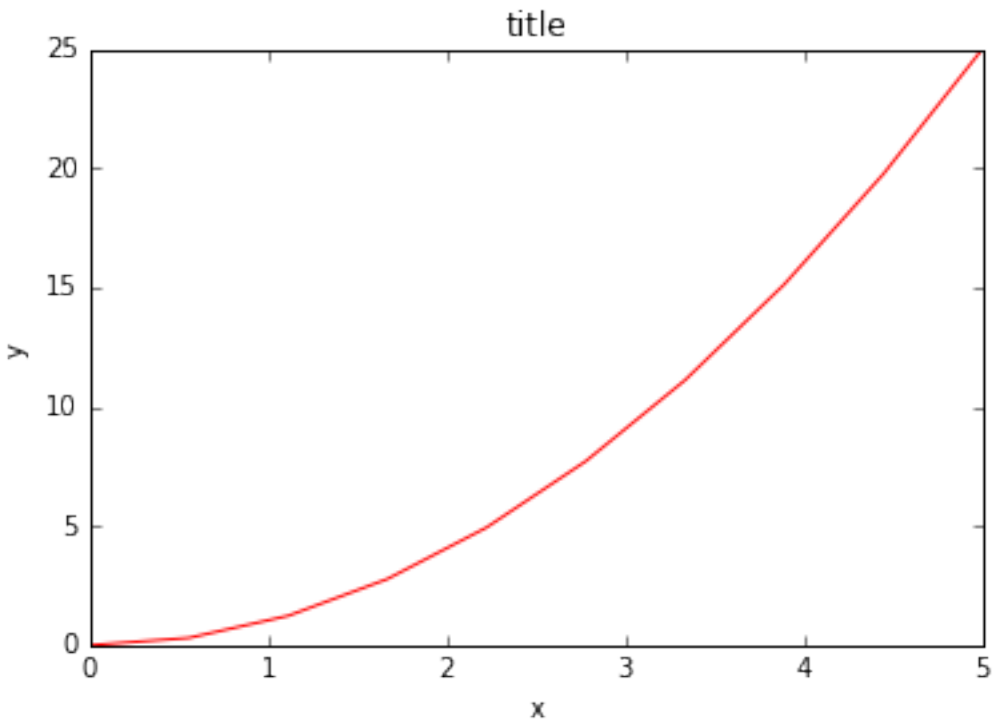
In [9]:
```
fig = plt.figure()

axes = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # left, bottom, width, height (range 0 to 1)

axes.plot(x, y, 'r')

axes.set_xlabel('x')
axes.set_ylabel('y')
axes.set_title('title');
```



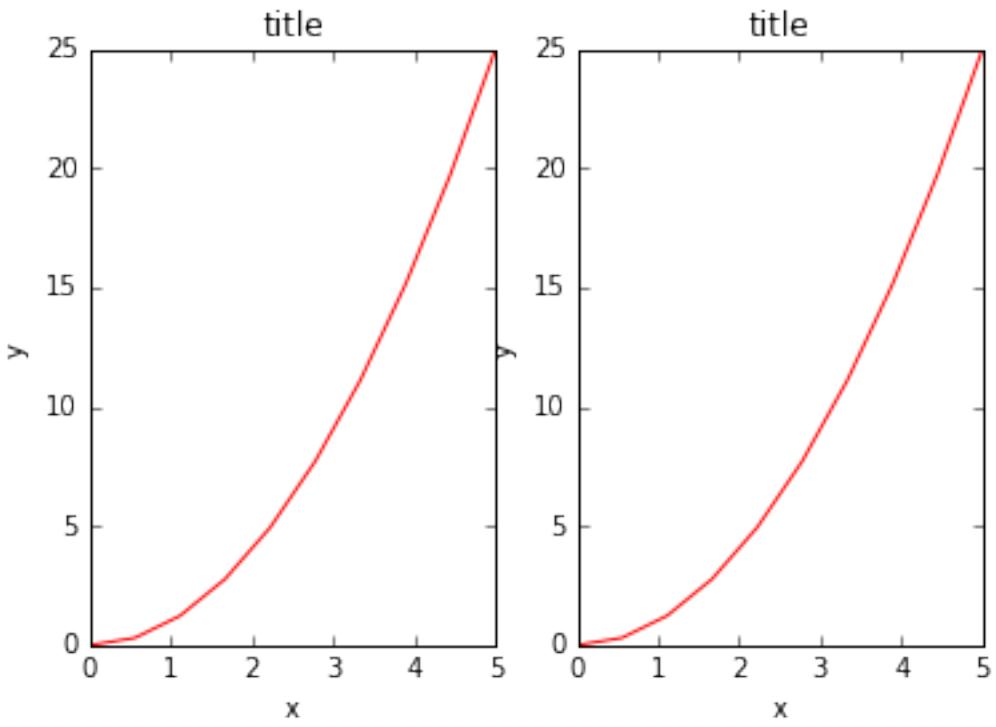

```
In [10]: fig = plt.figure()

         axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
         axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3]) # inset axes

         # main figure
         axes1.plot(x, y, 'r')
         axes1.set_xlabel('x')
         axes1.set_ylabel('y')
         axes1.set_title('title')

         # insert
         axes2.plot(y, x, 'g')
         axes2.set_xlabel('y')
         axes2.set_ylabel('x')
         axes2.set_title('insert title');
```



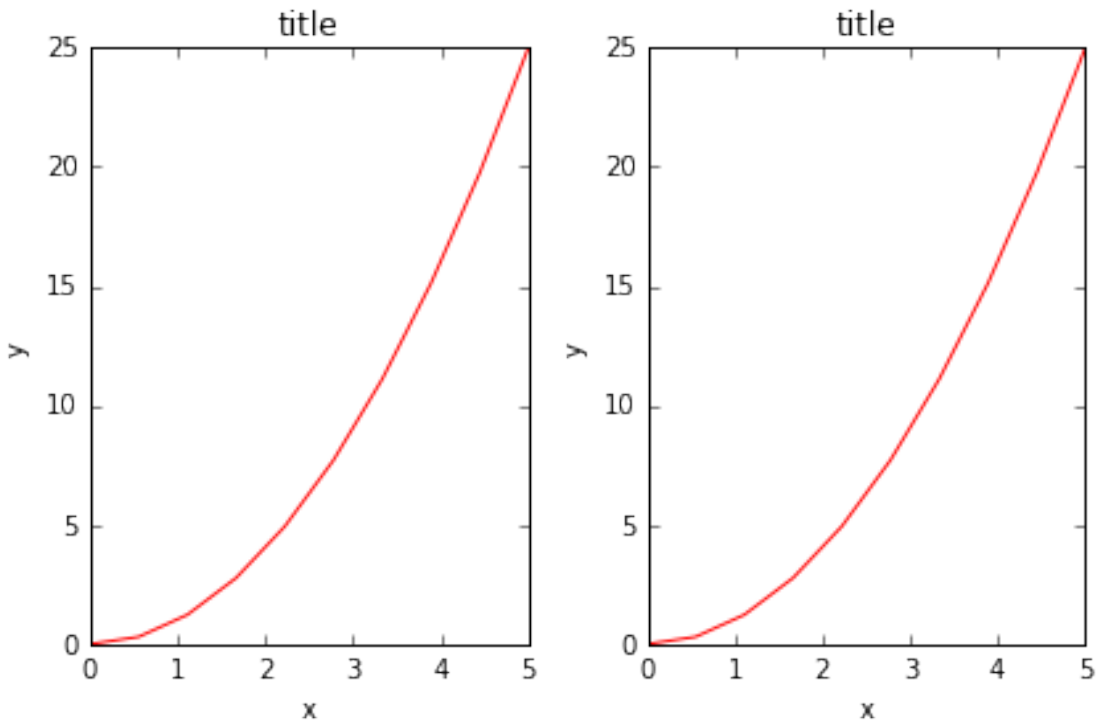

```
In [11]: fig, axes = plt.subplots()

         axes.plot(x, y, 'r')
         axes.set_xlabel('x')
         axes.set_ylabel('y')
         axes.set_title('title');
```




```
In [12]: fig, axes = plt.subplots(nrows=1, ncols=2)

         for ax in axes:
             ax.plot(x, y, 'r')
             ax.set_xlabel('x')
             ax.set_ylabel('y')
             ax.set_title('title')
```



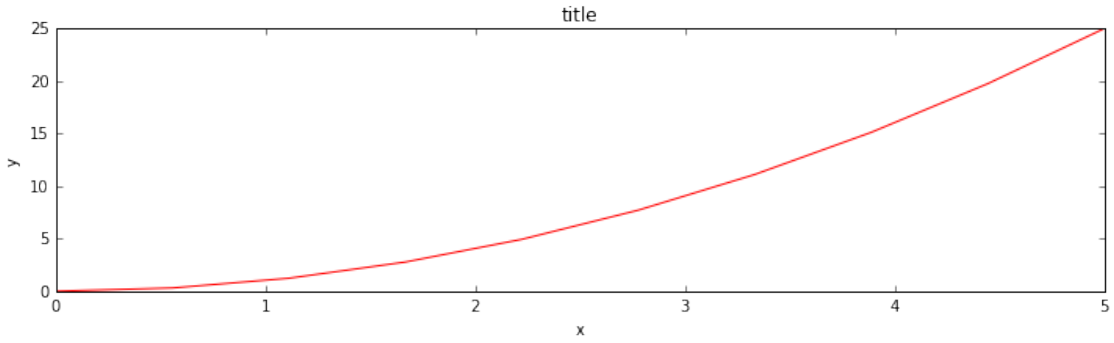

```
In [13]: fig, axes = plt.subplots(nrows=1, ncols=2)

         for ax in axes:
             ax.plot(x, y, 'r')
             ax.set_xlabel('x')
             ax.set_ylabel('y')
             ax.set_title('title')

         fig.tight_layout()
```




```
In [14]: fig = plt.figure(figsize=(8,4), dpi=100)
```

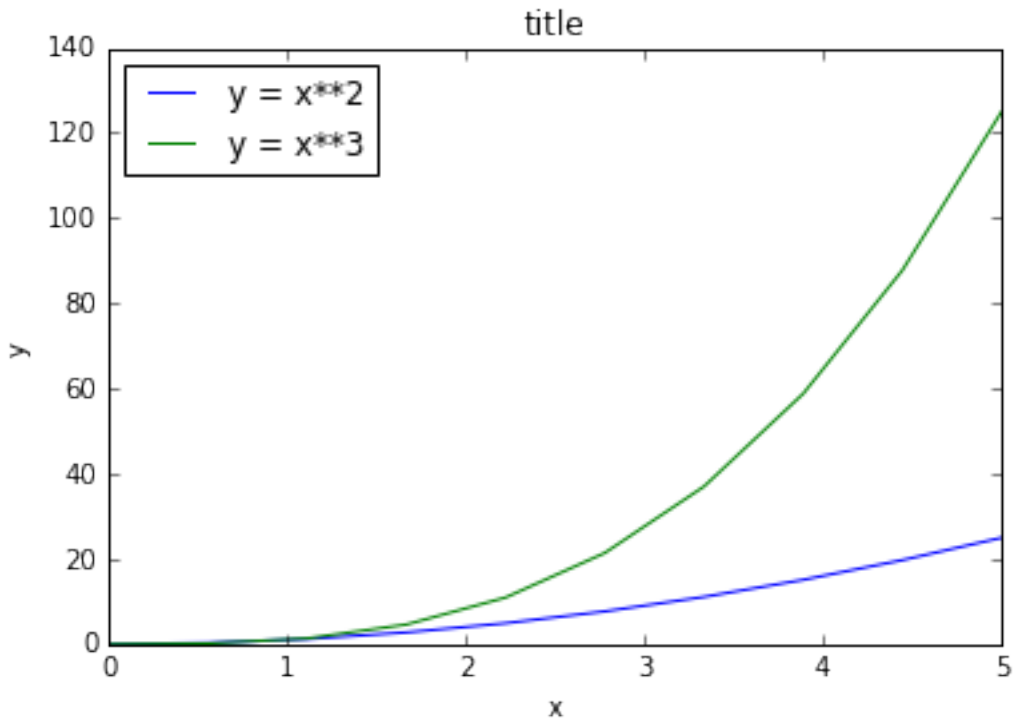
```
<matplotlib.figure.Figure at 0x8065320>
```

```
In [15]: fig, axes = plt.subplots(figsize=(12,3))

         axes.plot(x, y, 'r')
         axes.set_xlabel('x')
         axes.set_ylabel('y')
         axes.set_title('title');
```

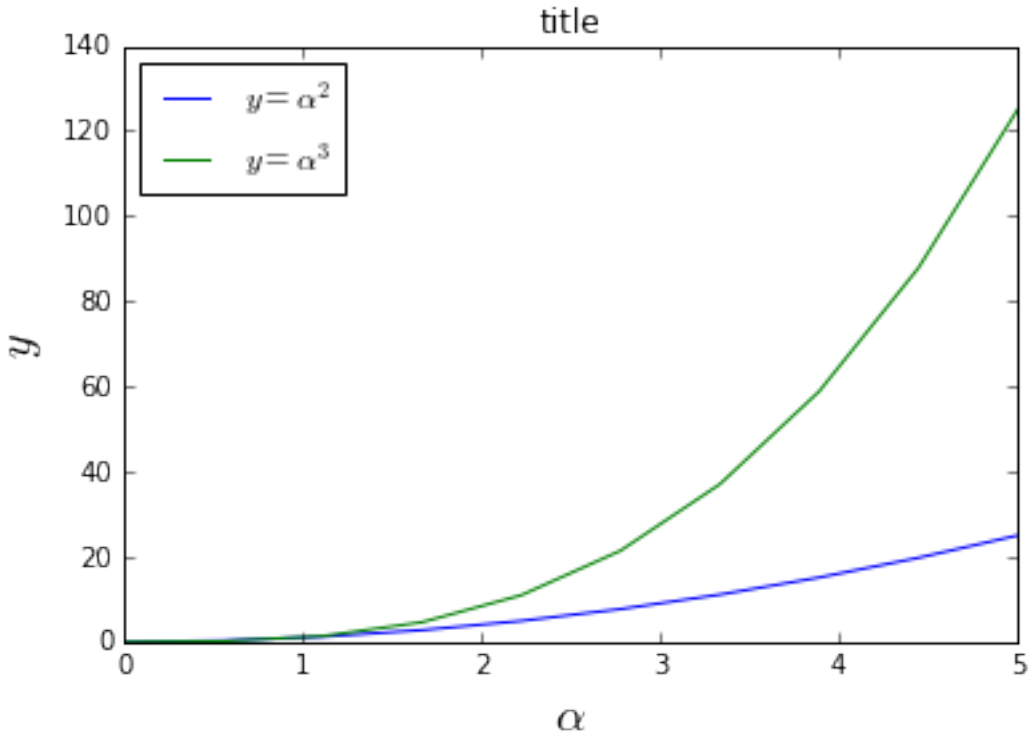



```
In [16]: fig.savefig("filename.png")
```

```
In [17]: fig.savefig("filename.png", dpi=200)

In [18]: ax.set_title("title");

In [19]: ax.set_xlabel("x")
         ax.set_ylabel("y");

In [20]: ax.legend(["curve1", "curve2", "curve3"]);

In [21]: ax.plot(x, x**2, label="curve1")
         ax.plot(x, x**3, label="curve2")
         ax.legend();

In [22]: ax.legend(loc=0) # let matplotlib decide the optimal location
         ax.legend(loc=1) # upper right corner
         ax.legend(loc=2) # upper left corner
         ax.legend(loc=3) # lower left corner
         ax.legend(loc=4) # lower right corner
         # .. many more options are available

Out[22]: <matplotlib.legend.Legend at 0x3dfc1d0>

In [23]: fig, ax = plt.subplots()

         ax.plot(x, x**2, label="y = x**2")
         ax.plot(x, x**3, label="y = x**3")
         ax.legend(loc=2); # upper left corner
         ax.set_xlabel('x')
         ax.set_ylabel('y')
         ax.set_title('title');
```

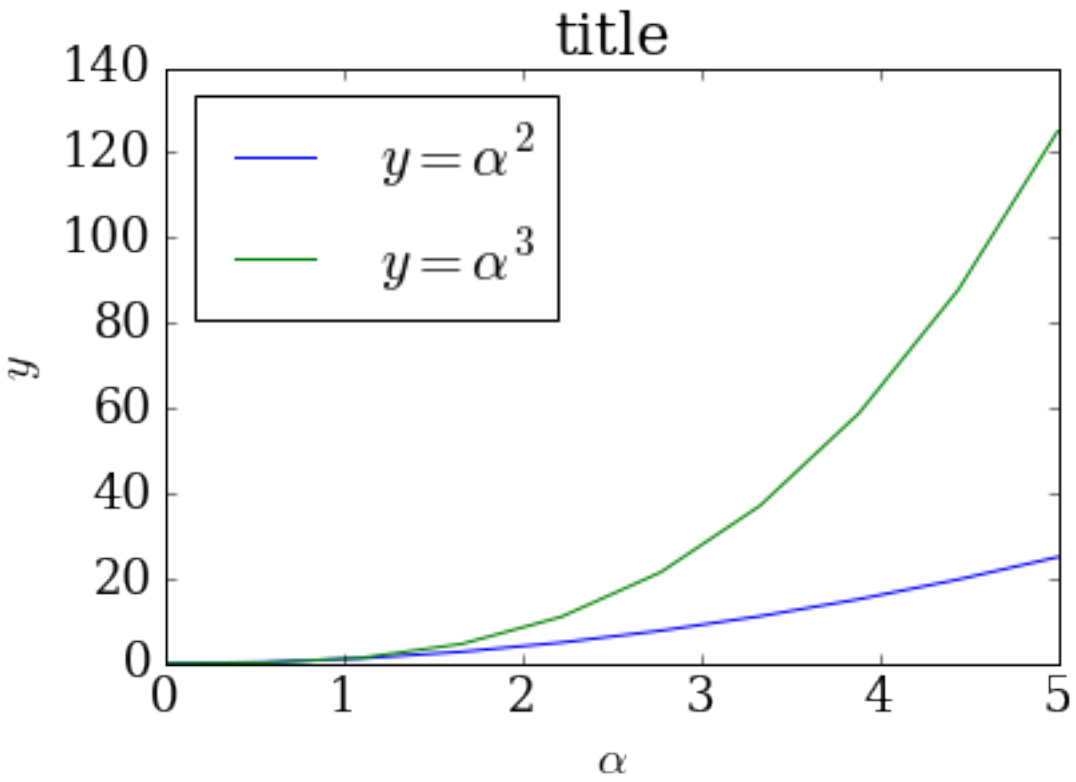


In [24]:
```
fig, ax = plt.subplots()

ax.plot(x, x**2, label=r"$y = \alpha^2$")
ax.plot(x, x**3, label=r"$y = \alpha^3$")
ax.legend(loc=2) # upper left corner
ax.set_xlabel(r'$\alpha$', fontsize=18)
ax.set_ylabel(r'$y$', fontsize=18)
ax.set_title('title');
```



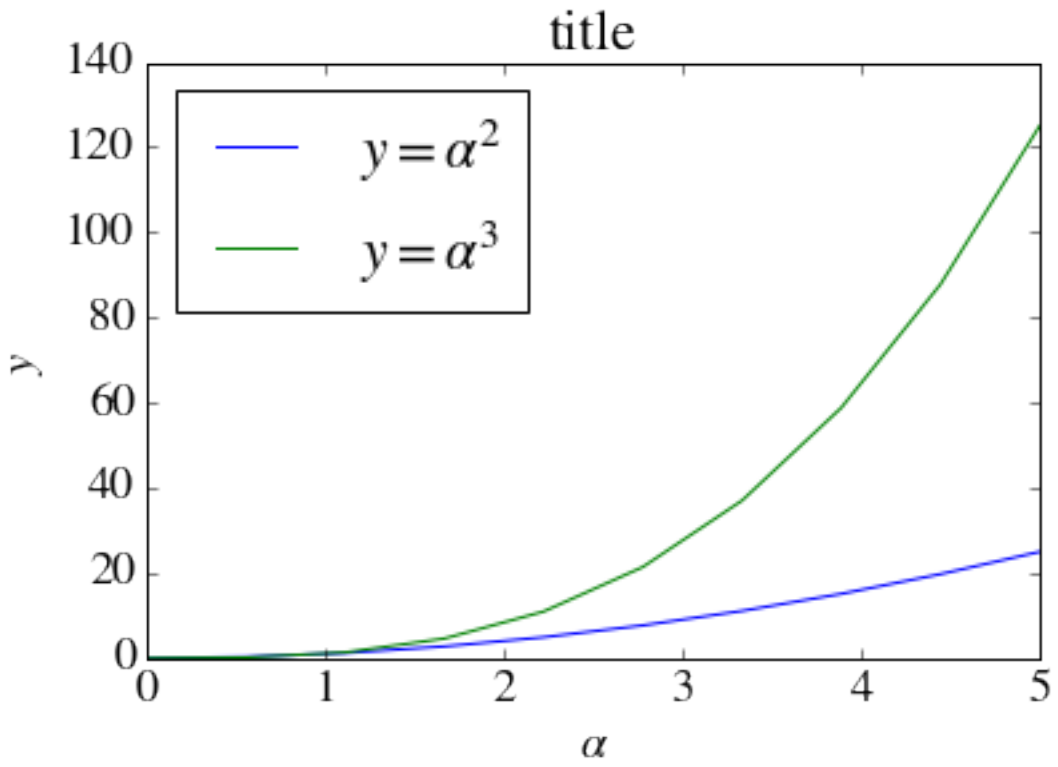
In [25]:
```
# Update the matplotlib configuration parameters:
matplotlib.rcParams.update({'font.size': 18, 'font.family': 'serif'})
```

In [26]:
```
fig, ax = plt.subplots()

ax.plot(x, x**2, label=r"$y = \alpha^2$")
ax.plot(x, x**3, label=r"$y = \alpha^3$")
ax.legend(loc=2) # upper left corner
ax.set_xlabel(r'$\alpha$')
ax.set_ylabel(r'$y$')
ax.set_title('title');
```



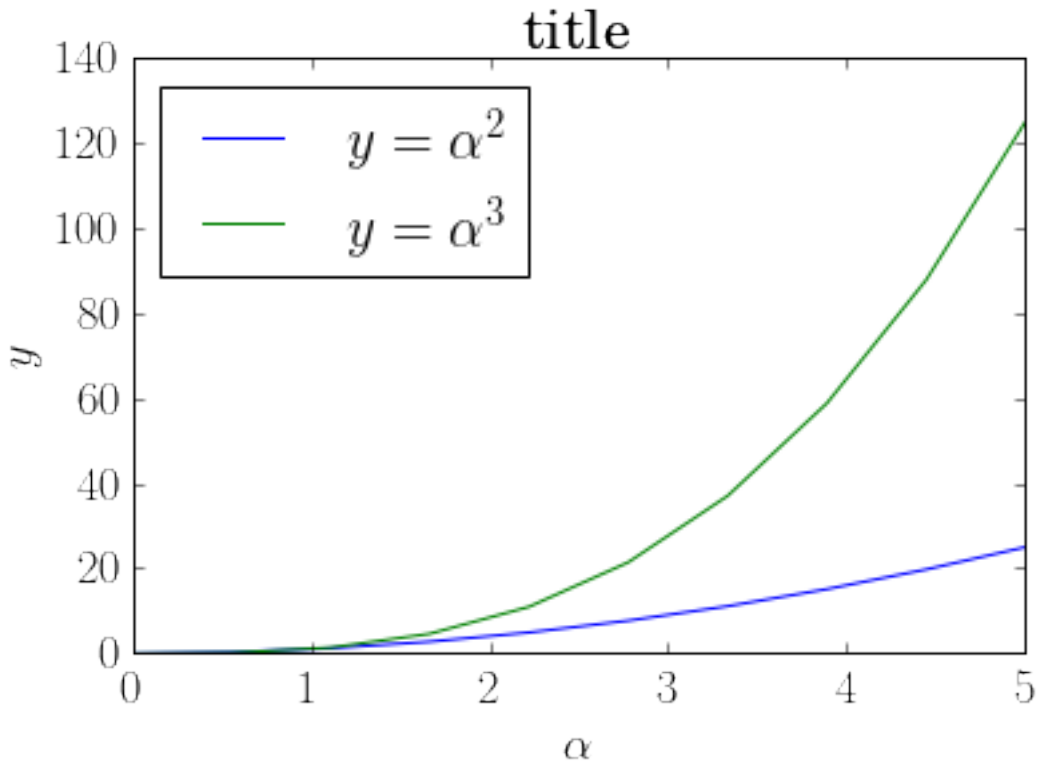
In [27]:
```
# Update the matplotlib configuration parameters:
matplotlib.rcParams.update({'font.size': 18, 'font.family': 'STIXGeneral', 'mathtext.fontset':
```

In [28]:
```
fig, ax = plt.subplots()

ax.plot(x, x**2, label=r"$y = \alpha^2$")
ax.plot(x, x**3, label=r"$y = \alpha^3$")
ax.legend(loc=2) # upper left corner
ax.set_xlabel(r'$\alpha$')
ax.set_ylabel(r'$y$')
ax.set_title('title');
```

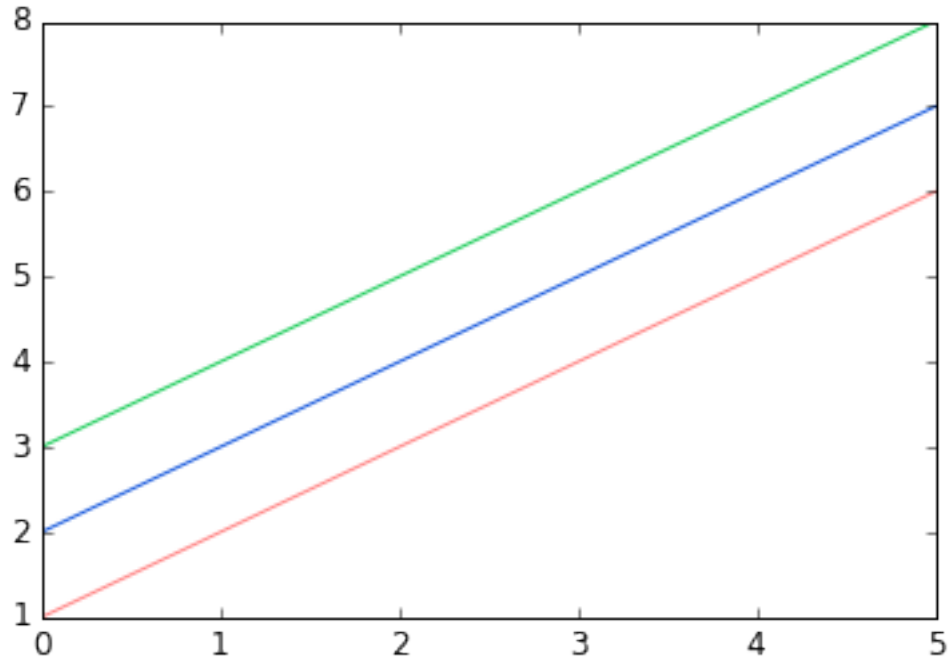


In [29]:
```
matplotlib.rcParams.update({'font.size': 18, 'text.usetex': True})
```

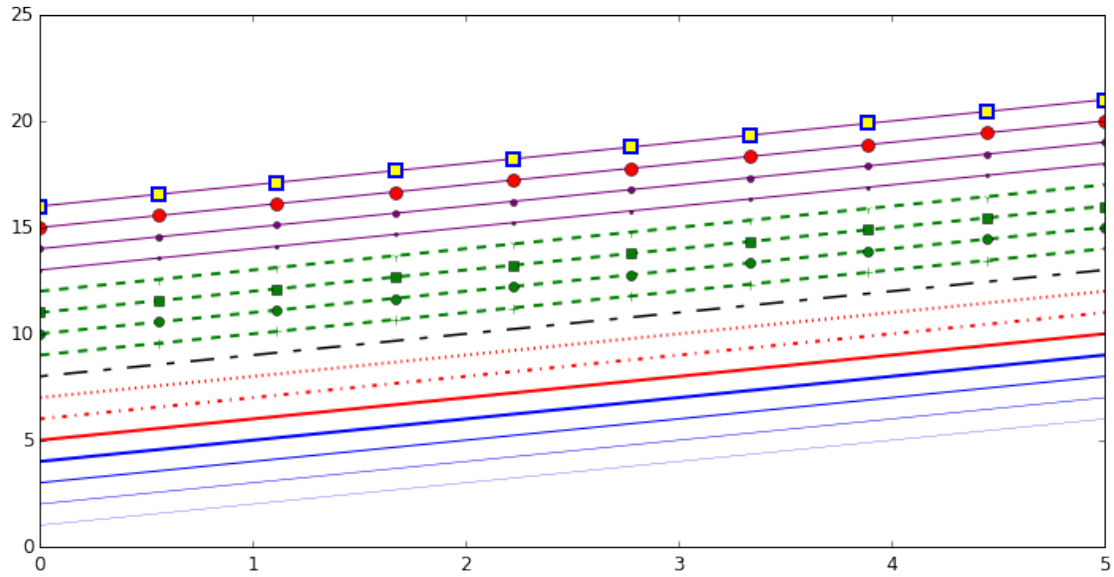
In [30]:
```
fig, ax = plt.subplots()

ax.plot(x, x**2, label=r"$y = \alpha^2$")
ax.plot(x, x**3, label=r"$y = \alpha^3$")
ax.legend(loc=2) # upper left corner
ax.set_xlabel(r'$\alpha$')
ax.set_ylabel(r'$y$')
ax.set_title('title');
```



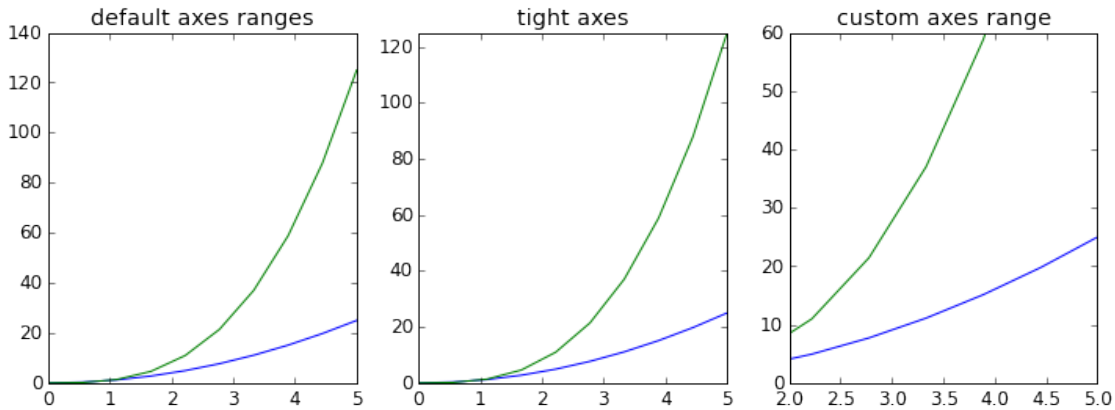

```
In [31]: # restore
         matplotlib.rcParams.update({'font.size': 12, 'font.family': 'sans', 'text.usetex': False})

In [32]: # MATLAB style line color and style
         ax.plot(x, x**2, 'b.-') # blue line with dots
         ax.plot(x, x**3, 'g--') # green dashed line

Out[32]: [<matplotlib.lines.Line2D at 0x96df0b8>]

In [33]: fig, ax = plt.subplots()

         ax.plot(x, x+1, color="red", alpha=0.5) # half-transparant red
         ax.plot(x, x+2, color="#1155dd")        # RGB hex code for a bluish color
         ax.plot(x, x+3, color="#15cc55")        # RGB hex code for a greenish color

Out[33]: [<matplotlib.lines.Line2D at 0x6fbc048>]
```



In [34]:
```
fig, ax = plt.subplots(figsize=(12,6))

ax.plot(x, x+1, color="blue", linewidth=0.25)
ax.plot(x, x+2, color="blue", linewidth=0.50)
ax.plot(x, x+3, color="blue", linewidth=1.00)
ax.plot(x, x+4, color="blue", linewidth=2.00)

# possible linestype options ‘-‘, ‘--’, ‘-.’, ‘:’, ‘steps’
ax.plot(x, x+5, color="red", lw=2, linestyle='-')
ax.plot(x, x+6, color="red", lw=2, ls='-.')
ax.plot(x, x+7, color="red", lw=2, ls=':')

# custom dash
line, = ax.plot(x, x+8, color="black", lw=1.50)
line.set_dashes([5, 10, 15, 10]) # format: line length, space length, ...

# possible marker symbols: marker = '+', 'o', '*', 's', ',', '.', '1', '2', '3', '4', ...
ax.plot(x, x+ 9, color="green", lw=2, ls='--', marker='+')
ax.plot(x, x+10, color="green", lw=2, ls='--', marker='o')
ax.plot(x, x+11, color="green", lw=2, ls='--', marker='s')
ax.plot(x, x+12, color="green", lw=2, ls='--', marker='1')

# marker size and color
ax.plot(x, x+13, color="purple", lw=1, ls='-', marker='o', markersize=2)
ax.plot(x, x+14, color="purple", lw=1, ls='-', marker='o', markersize=4)
ax.plot(x, x+15, color="purple", lw=1, ls='-', marker='o', markersize=8, markerfacecolor="red"
ax.plot(x, x+16, color="purple", lw=1, ls='-', marker='s', markersize=8,
        markerfacecolor="yellow", markeredgewidth=2, markeredgecolor="blue");
```

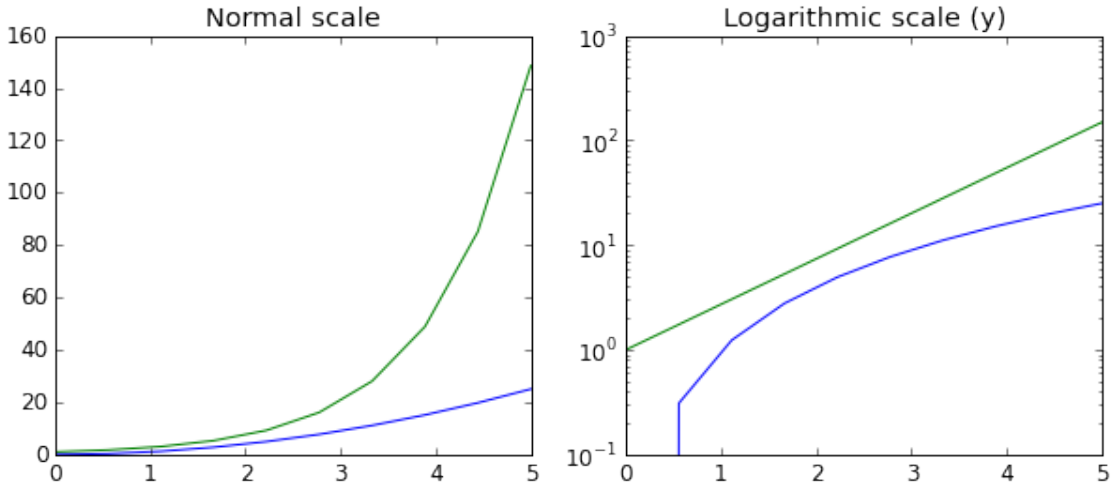


In [35]:
```
fig, axes = plt.subplots(1, 3, figsize=(12, 4))

axes[0].plot(x, x**2, x, x**3)
axes[0].set_title("default axes ranges")

axes[1].plot(x, x**2, x, x**3)
axes[1].axis('tight')
axes[1].set_title("tight axes")

axes[2].plot(x, x**2, x, x**3)
axes[2].set_ylim([0, 60])
axes[2].set_xlim([2, 5])
axes[2].set_title("custom axes range");
```



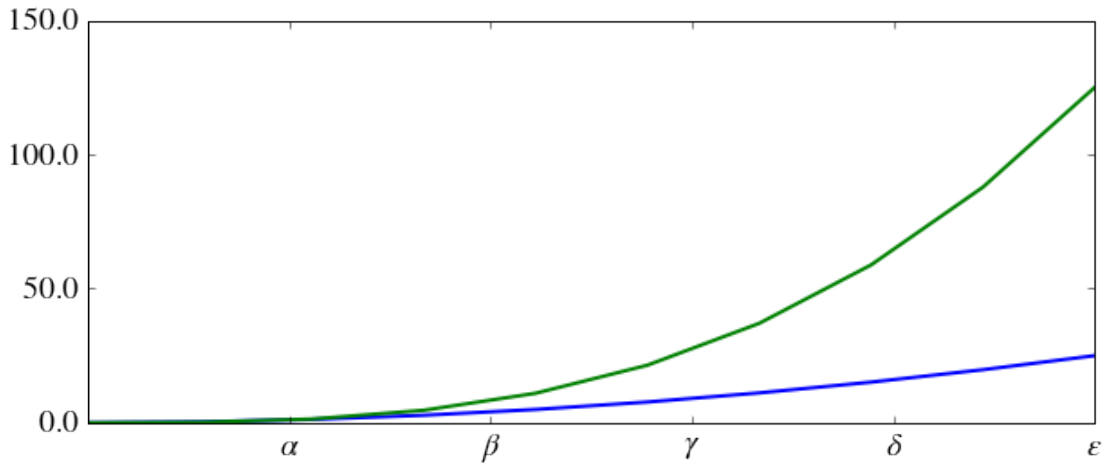
In [36]:
```
fig, axes = plt.subplots(1, 2, figsize=(10,4))
```

```
axes[0].plot(x, x**2, x, np.exp(x))
axes[0].set_title("Normal scale")

axes[1].plot(x, x**2, x, np.exp(x))
axes[1].set_yscale("log")
axes[1].set_title("Logarithmic scale (y)");
```

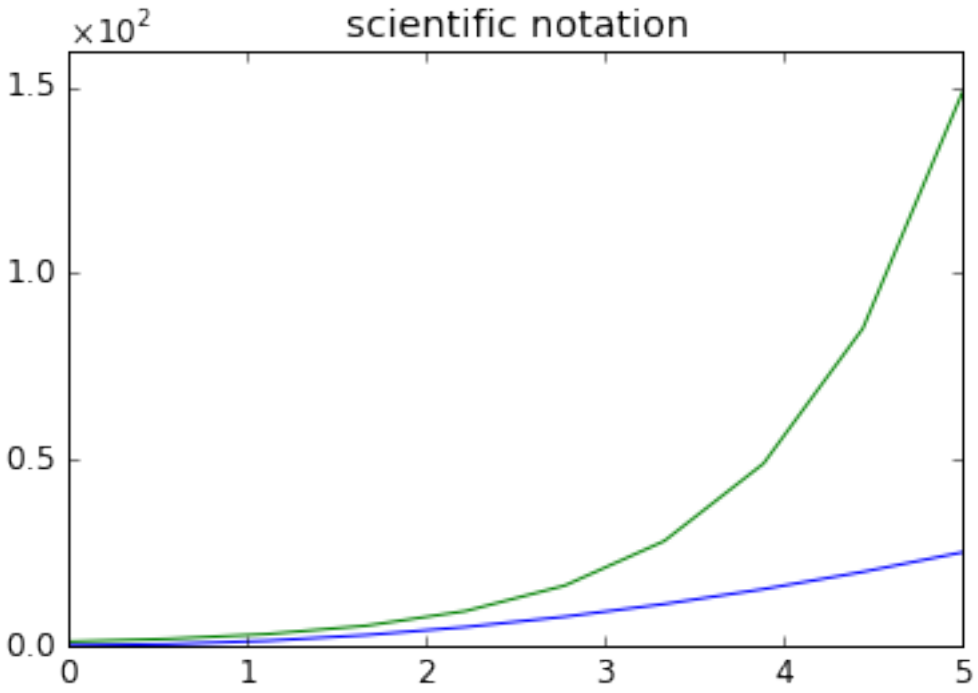


In [37]:
```
fig, ax = plt.subplots(figsize=(10, 4))

ax.plot(x, x**2, x, x**3, lw=2)

ax.set_xticks([1, 2, 3, 4, 5])
ax.set_xticklabels([r'$\alpha$', r'$\beta$', r'$\gamma$', r'$\delta$', r'$\epsilon$'], fontsiz

yticks = [0, 50, 100, 150]
ax.set_yticks(yticks)
ax.set_yticklabels(["$%.1f$" % y for y in yticks], fontsize=18); # use LaTeX formatted labels
```

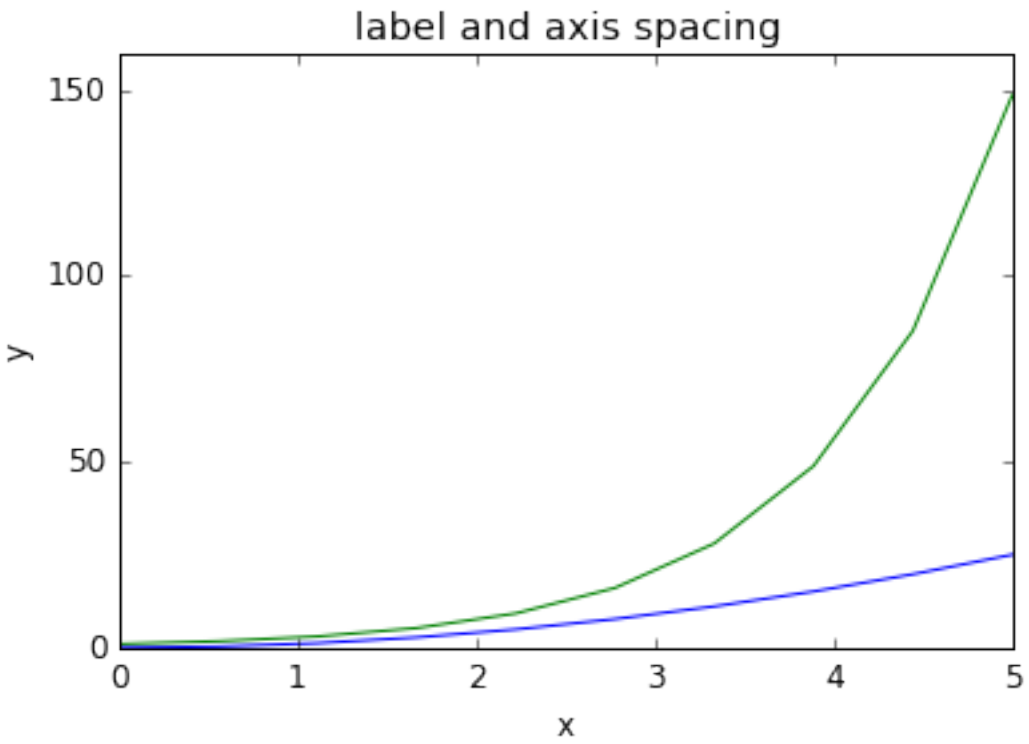
Out[37]:
```
[<matplotlib.text.Text at 0x10a3ae610>,
 <matplotlib.text.Text at 0x10a3aedd0>,
 <matplotlib.text.Text at 0x10a3fe110>,
 <matplotlib.text.Text at 0x10a3fe750>]
```



In [38]:
```
fig, ax = plt.subplots(1, 1)

ax.plot(x, x**2, x, np.exp(x))
ax.set_title("scientific notation")

ax.set_yticks([0, 50, 100, 150])

from matplotlib import ticker
formatter = ticker.ScalarFormatter(useMathText=True)
formatter.set_scientific(True)
formatter.set_powerlimits((-1,1))
ax.yaxis.set_major_formatter(formatter)
```



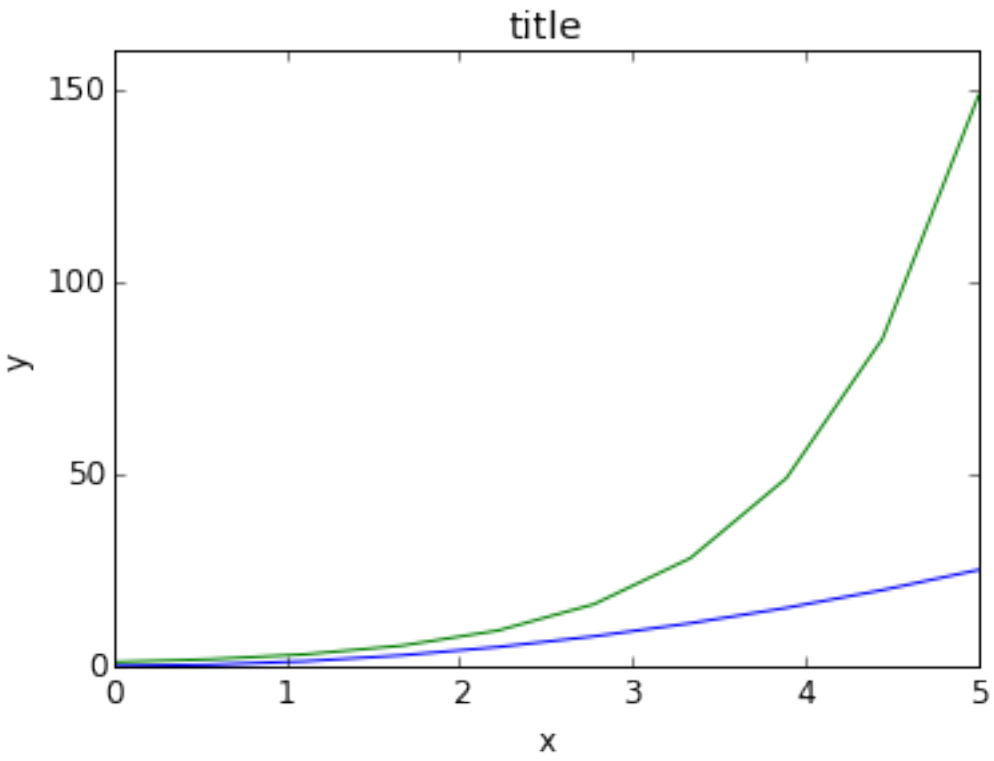
In [39]:
```
# distance between x and y axis and the numbers on the axes
matplotlib.rcParams['xtick.major.pad'] = 5
matplotlib.rcParams['ytick.major.pad'] = 5

fig, ax = plt.subplots(1, 1)

ax.plot(x, x**2, x, np.exp(x))
ax.set_yticks([0, 50, 100, 150])

ax.set_title("label and axis spacing")

# padding between axis label and axis numbers
ax.xaxis.labelpad = 5
ax.yaxis.labelpad = 5

ax.set_xlabel("x")
ax.set_ylabel("y");
```



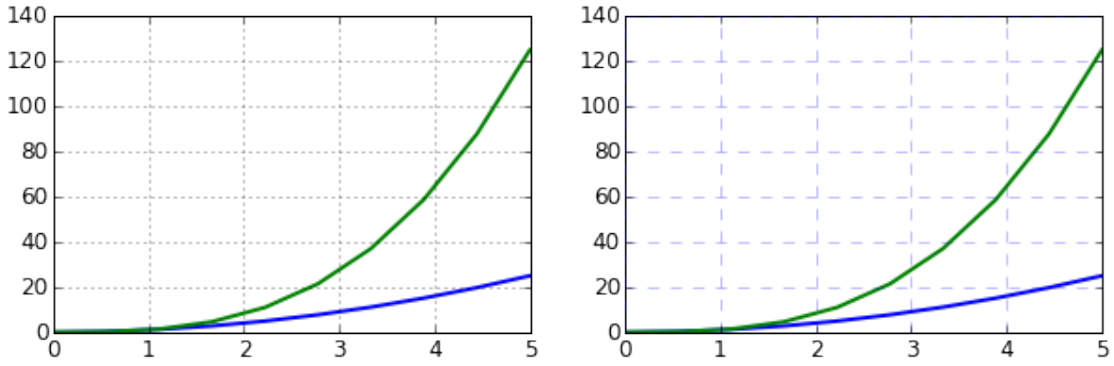
In [40]:
```
# restore defaults
matplotlib.rcParams['xtick.major.pad'] = 3
matplotlib.rcParams['ytick.major.pad'] = 3
```

In [41]:
```
fig, ax = plt.subplots(1, 1)
```

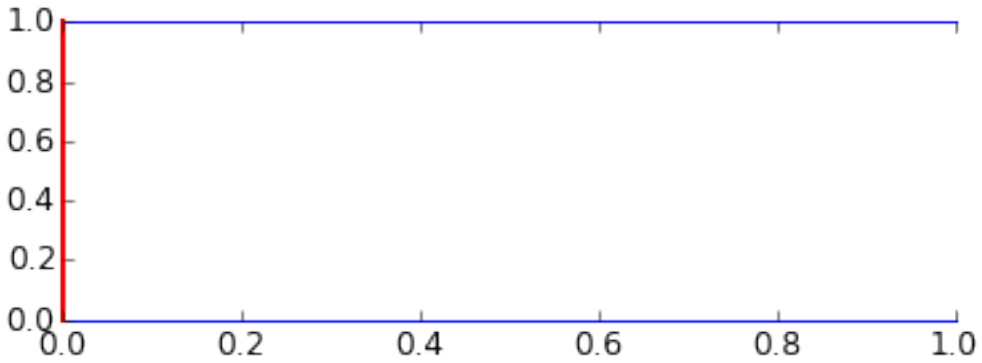
```
ax.plot(x, x**2, x, np.exp(x))
ax.set_yticks([0, 50, 100, 150])

ax.set_title("title")
ax.set_xlabel("x")
ax.set_ylabel("y")

fig.subplots_adjust(left=0.15, right=.9, bottom=0.1, top=0.9);
```



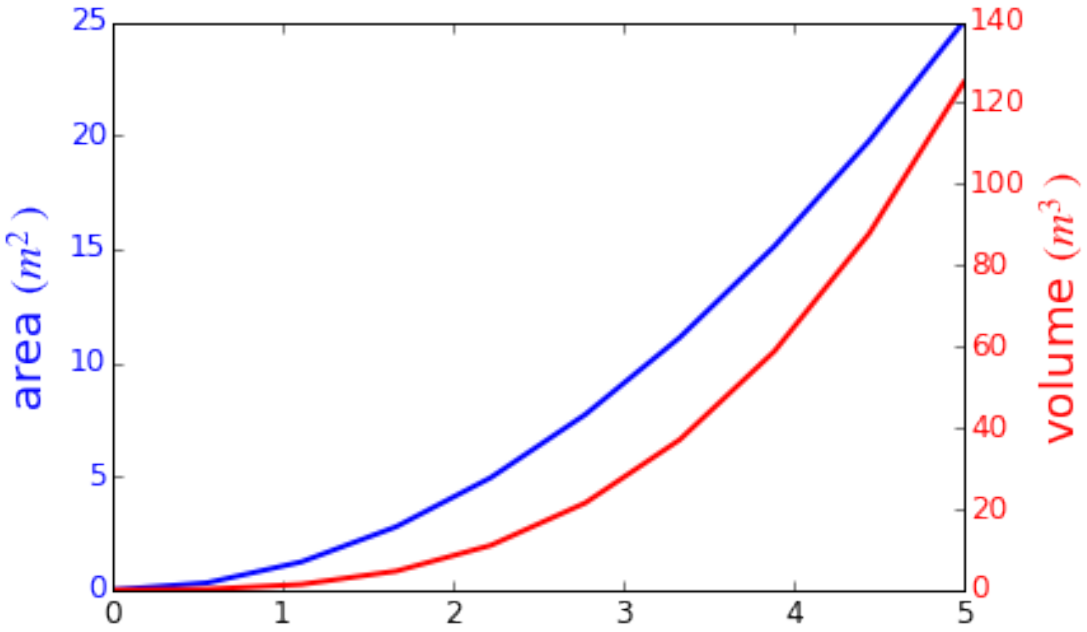
In [42]:
```
fig, axes = plt.subplots(1, 2, figsize=(10,3))

# default grid appearance
axes[0].plot(x, x**2, x, x**3, lw=2)
axes[0].grid(True)

# custom grid appearance
axes[1].plot(x, x**2, x, x**3, lw=2)
axes[1].grid(color='b', alpha=0.5, linestyle='dashed', linewidth=0.5)
```



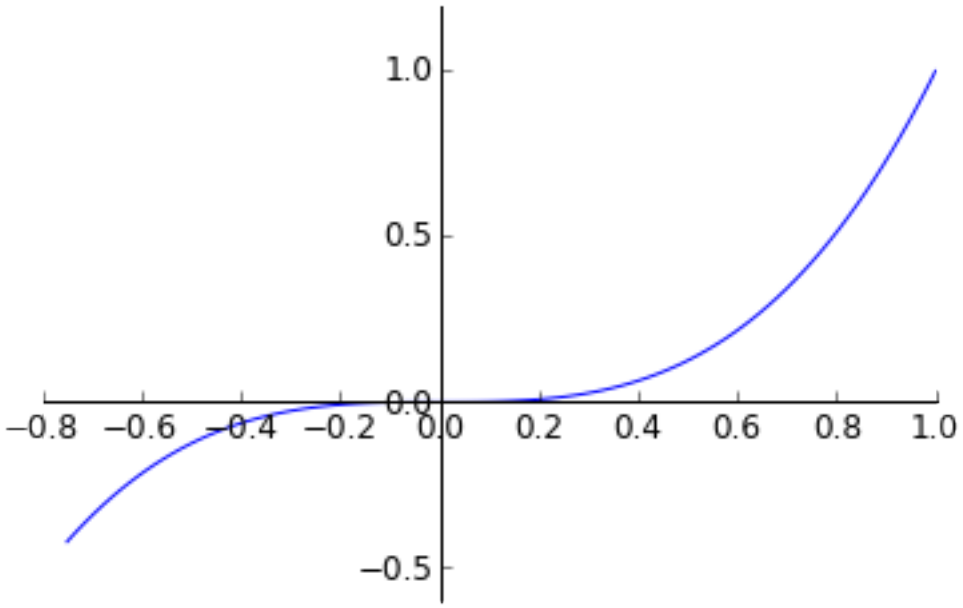
In [43]:
```
fig, ax = plt.subplots(figsize=(6,2))

ax.spines['bottom'].set_color('blue')
ax.spines['top'].set_color('blue')

ax.spines['left'].set_color('red')
ax.spines['left'].set_linewidth(2)

# turn off axis spine to the right
ax.spines['right'].set_color("none")
ax.yaxis.tick_left() # only ticks on the left side
```



In [44]:
```
fig, ax1 = plt.subplots()

ax1.plot(x, x**2, lw=2, color="blue")
ax1.set_ylabel(r"area $(m^2)$", fontsize=18, color="blue")
for label in ax1.get_yticklabels():
    label.set_color("blue")

ax2 = ax1.twinx()
ax2.plot(x, x**3, lw=2, color="red")
ax2.set_ylabel(r"volume $(m^3)$", fontsize=18, color="red")
for label in ax2.get_yticklabels():
    label.set_color("red")
```

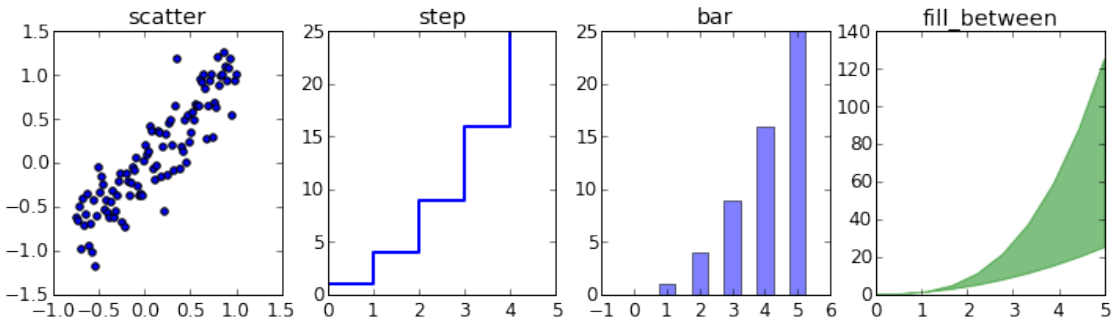



```
In [45]: fig, ax = plt.subplots()

         ax.spines['right'].set_color('none')
         ax.spines['top'].set_color('none')

         ax.xaxis.set_ticks_position('bottom')
         ax.spines['bottom'].set_position(('data',0)) # set position of x spine to x=0

         ax.yaxis.set_ticks_position('left')
         ax.spines['left'].set_position(('data',0))   # set position of y spine to y=0

         xx = np.linspace(-0.75, 1., 100)
         ax.plot(xx, xx**3);
```



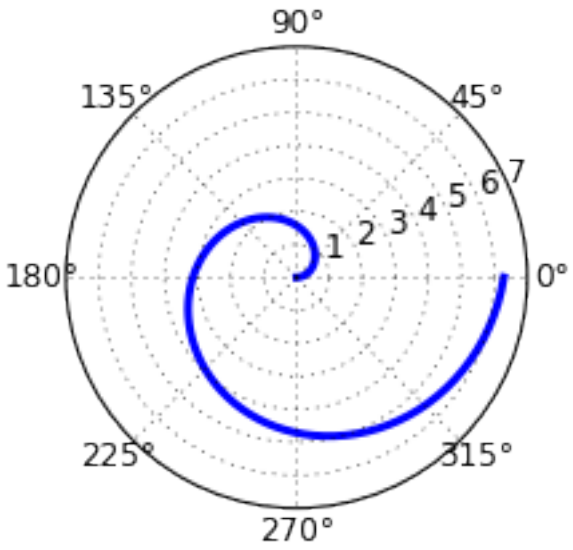

```
In [46]: n = np.array([0,1,2,3,4,5])
```

```
In [47]: fig, axes = plt.subplots(1, 4, figsize=(12,3))

         axes[0].scatter(xx, xx + 0.25*np.random.randn(len(xx)))
         axes[0].set_title("scatter")

         axes[1].step(n, n**2, lw=2)
         axes[1].set_title("step")

         axes[2].bar(n, n**2, align="center", width=0.5, alpha=0.5)
         axes[2].set_title("bar")

         axes[3].fill_between(x, x**2, x**3, color="green", alpha=0.5);
         axes[3].set_title("fill_between");
```

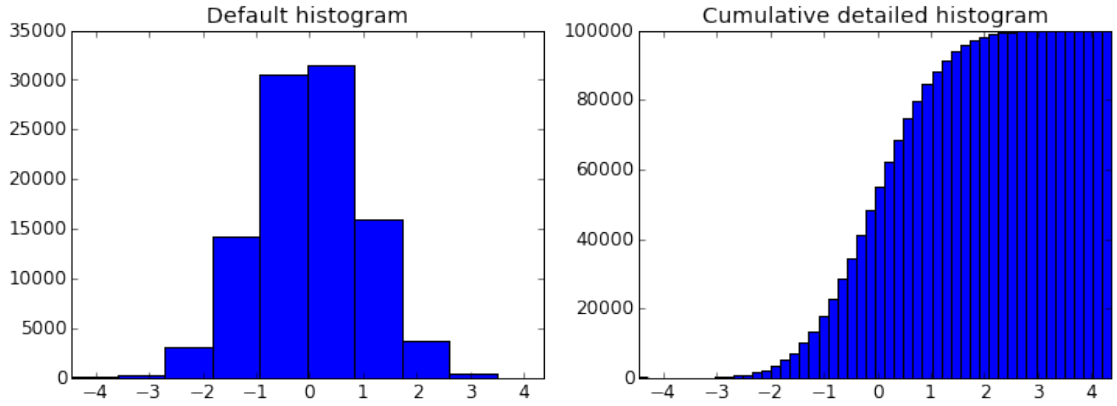



```
In [48]: # polar plot using add_axes and polar projection
         fig = plt.figure()
```

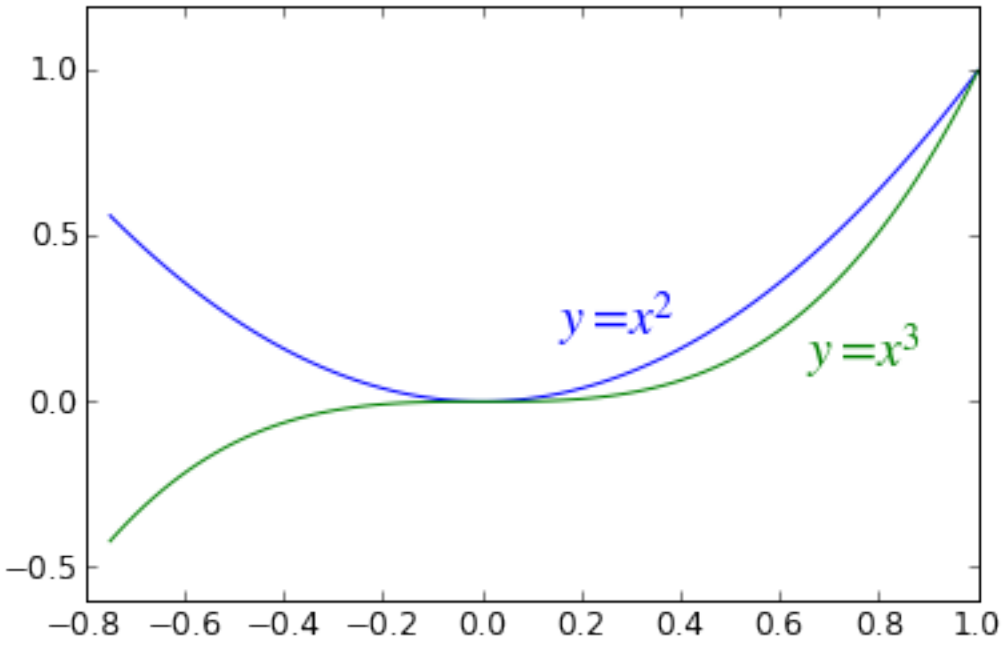
```
ax = fig.add_axes([0.0, 0.0, .6, .6], polar=True)
t = np.linspace(0, 2 * np.pi, 100)
ax.plot(t, t, color='blue', lw=3);
```



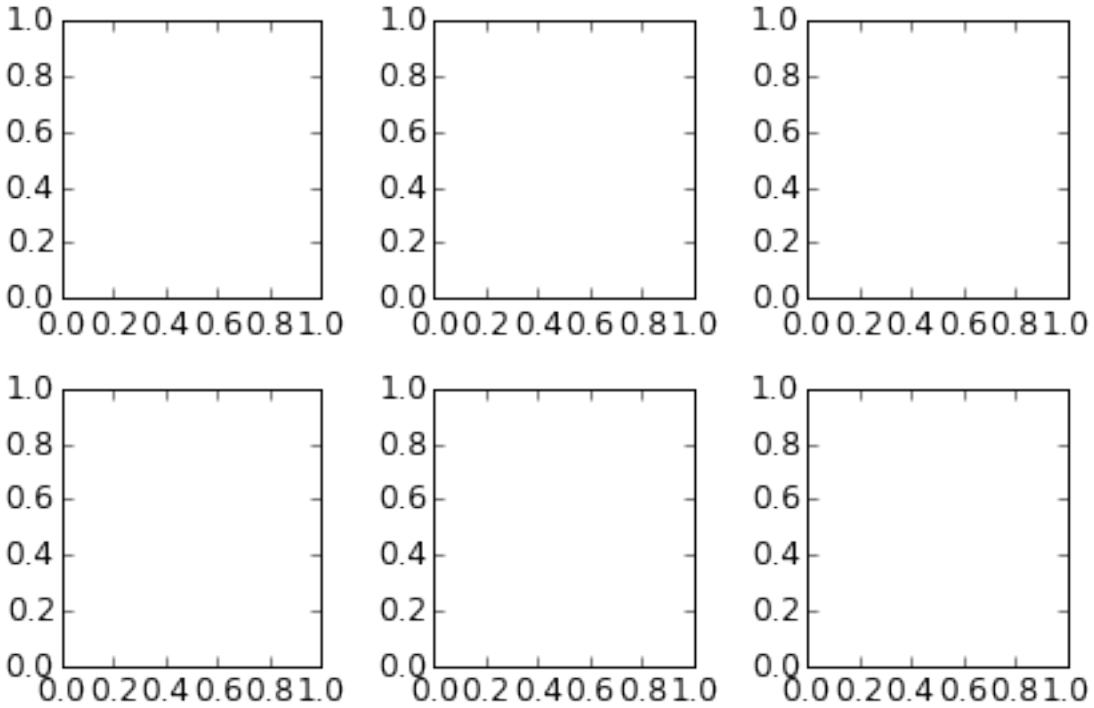
In [49]:
```
# A histogram
n = np.random.randn(100000)
fig, axes = plt.subplots(1, 2, figsize=(12,4))

axes[0].hist(n)
axes[0].set_title("Default histogram")
axes[0].set_xlim((min(n), max(n)))

axes[1].hist(n, cumulative=True, bins=50)
axes[1].set_title("Cumulative detailed histogram")
axes[1].set_xlim((min(n), max(n)));
```



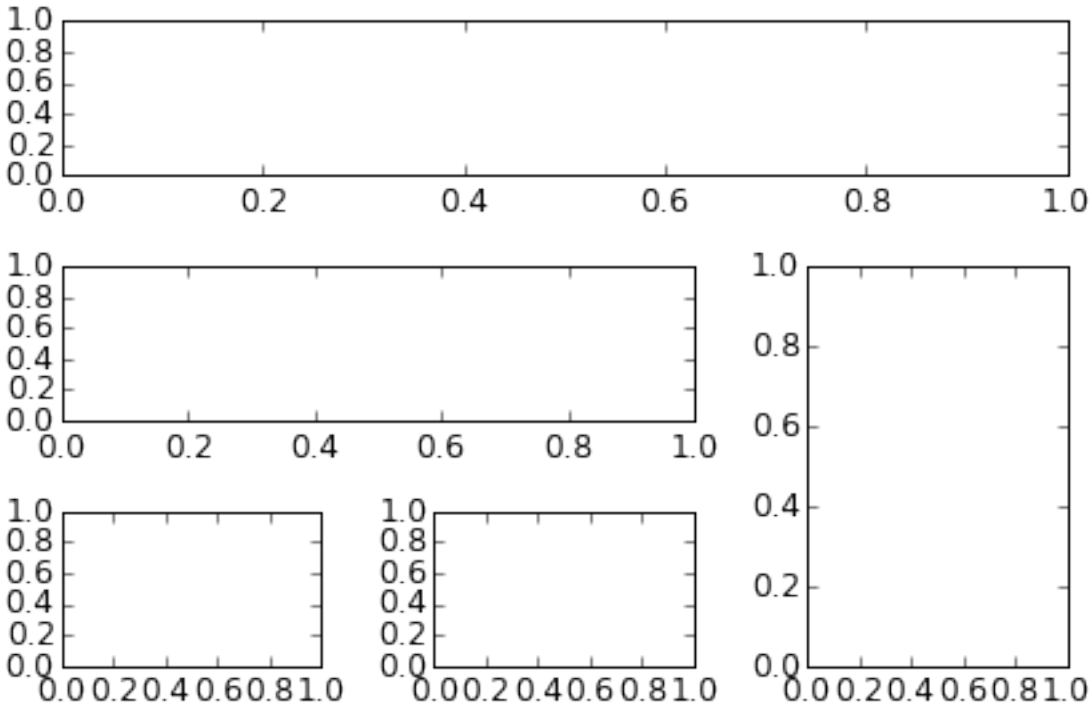
In [50]:
```
fig, ax = plt.subplots()

ax.plot(xx, xx**2, xx, xx**3)

ax.text(0.15, 0.2, r"$y=x^2$", fontsize=20, color="blue")
ax.text(0.65, 0.1, r"$y=x^3$", fontsize=20, color="green");
```



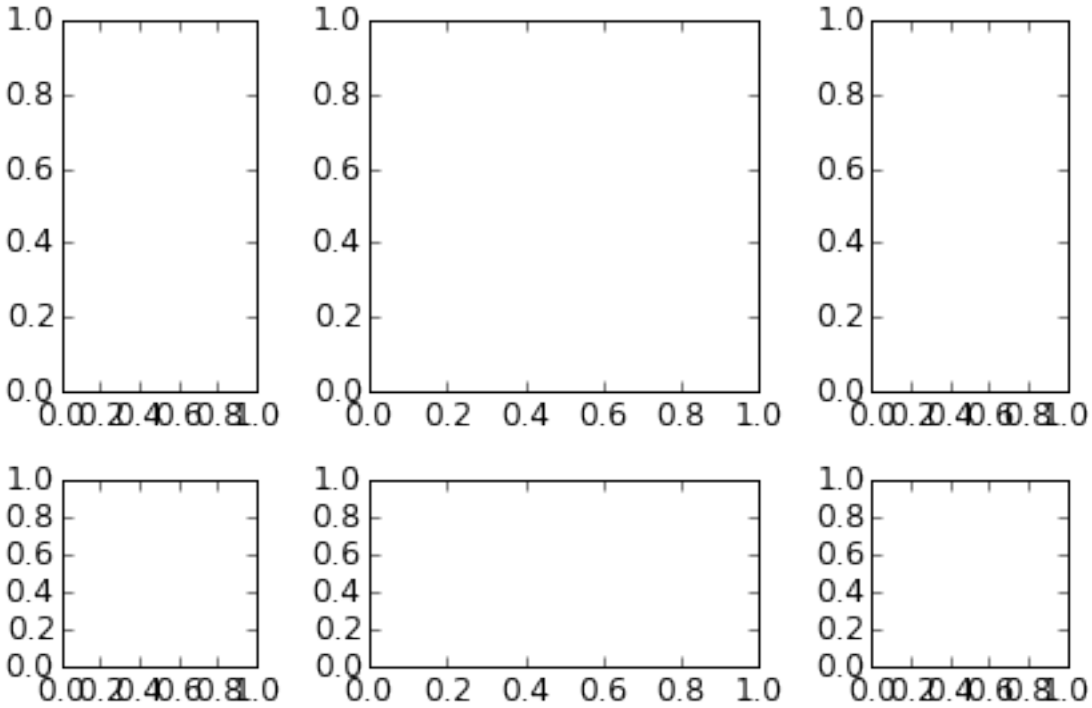
In [51]:
```
fig, ax = plt.subplots(2, 3)
fig.tight_layout()
```



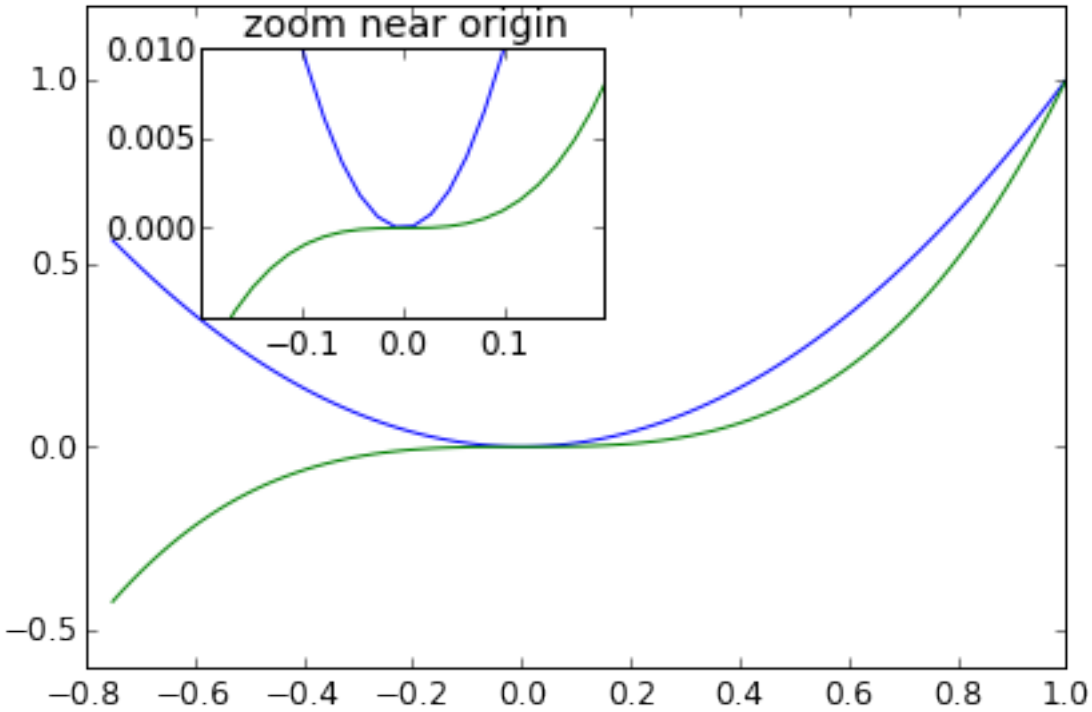

```
In [52]: fig = plt.figure()
         ax1 = plt.subplot2grid((3,3), (0,0), colspan=3)
         ax2 = plt.subplot2grid((3,3), (1,0), colspan=2)
         ax3 = plt.subplot2grid((3,3), (1,2), rowspan=2)
         ax4 = plt.subplot2grid((3,3), (2,0))
         ax5 = plt.subplot2grid((3,3), (2,1))
         fig.tight_layout()
```



```
In [53]: import matplotlib.gridspec as gridspec

In [54]: fig = plt.figure()

         gs = gridspec.GridSpec(2, 3, height_ratios=[2,1], width_ratios=[1,2,1])
         for g in gs:
             ax = fig.add_subplot(g)

         fig.tight_layout()
```



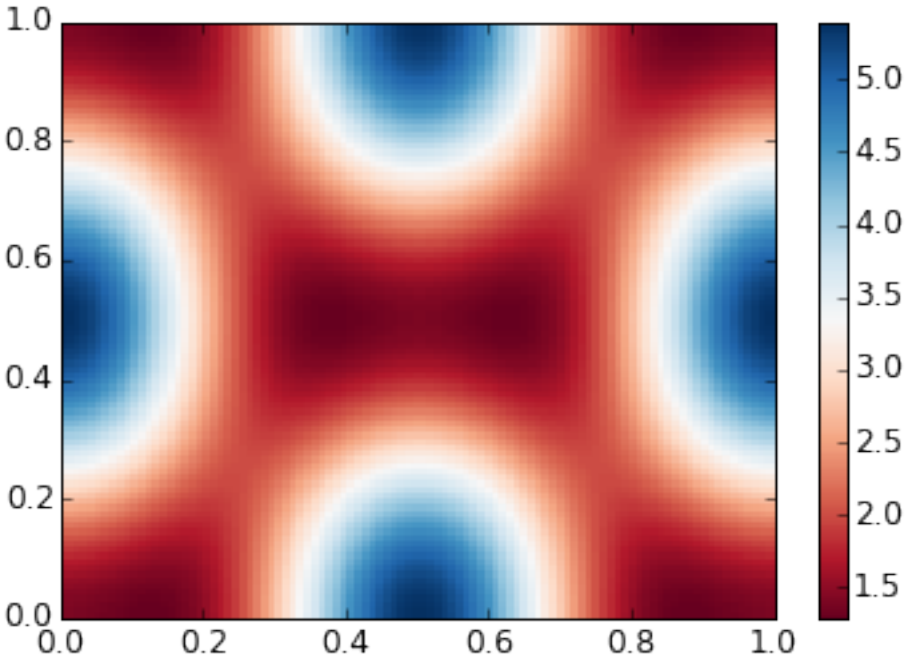
In [55]:
```
fig, ax = plt.subplots()

ax.plot(xx, xx**2, xx, xx**3)
fig.tight_layout()

# inset
inset_ax = fig.add_axes([0.2, 0.55, 0.35, 0.35]) # X, Y, width, height

inset_ax.plot(xx, xx**2, xx, xx**3)
inset_ax.set_title('zoom near origin')

# set axis range
inset_ax.set_xlim(-.2, .2)
inset_ax.set_ylim(-.005, .01)

# set axis tick locations
inset_ax.set_yticks([0, 0.005, 0.01])
inset_ax.set_xticks([-0.1,0,.1]);
```



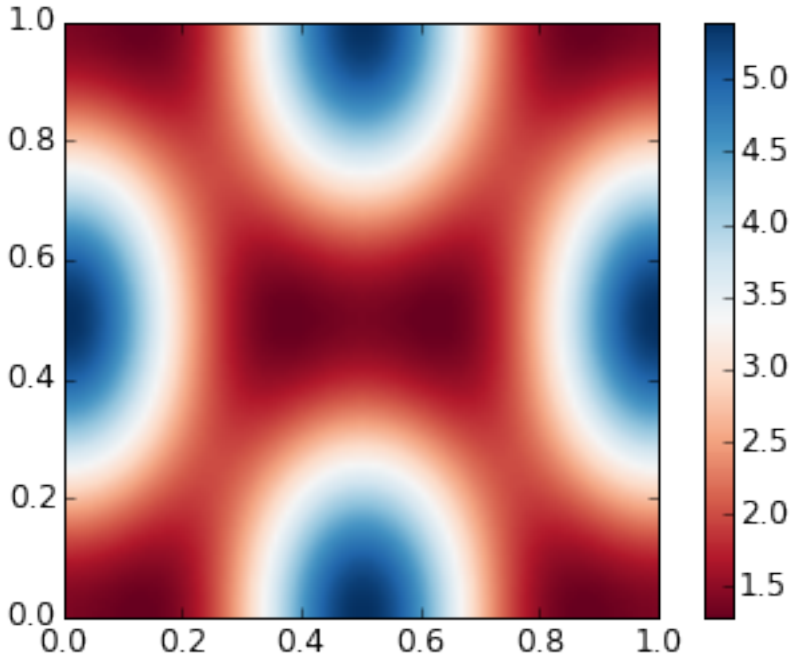
In [56]:
```
alpha = 0.7
phi_ext = 2 * np.pi * 0.5

def flux_qubit_potential(phi_m, phi_p):
    return 2 + alpha - 2 * np.cos(phi_p) * np.cos(phi_m) - alpha * np.cos(phi_ext - 2*phi_p)
```

In [57]:
```
phi_m = np.linspace(0, 2*np.pi, 100)
phi_p = np.linspace(0, 2*np.pi, 100)
X,Y = np.meshgrid(phi_p, phi_m)
Z = flux_qubit_potential(X, Y).T
```

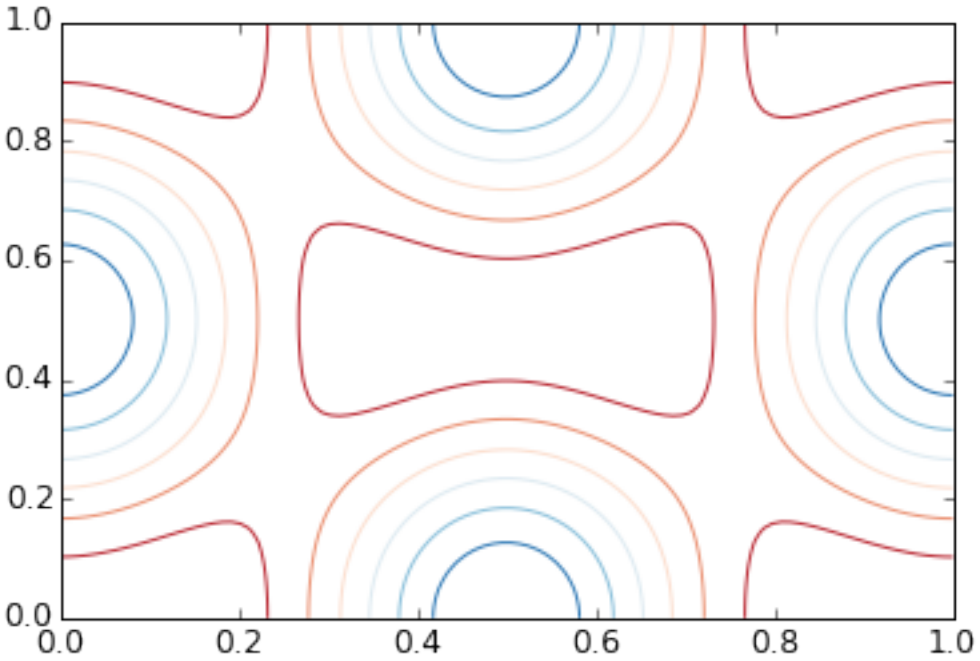
In [58]:
```
fig, ax = plt.subplots()

p = ax.pcolor(X/(2*np.pi), Y/(2*np.pi), Z, cmap=matplotlib.cm.RdBu, vmin=abs(Z).min(), vmax=abs
cb = fig.colorbar(p, ax=ax)
```



In [59]:
```
fig, ax = plt.subplots()

im = ax.imshow(Z, cmap=matplotlib.cm.RdBu, vmin=abs(Z).min(), vmax=abs(Z).max(), extent=[0, 1,
im.set_interpolation('bilinear')

cb = fig.colorbar(im, ax=ax)
```

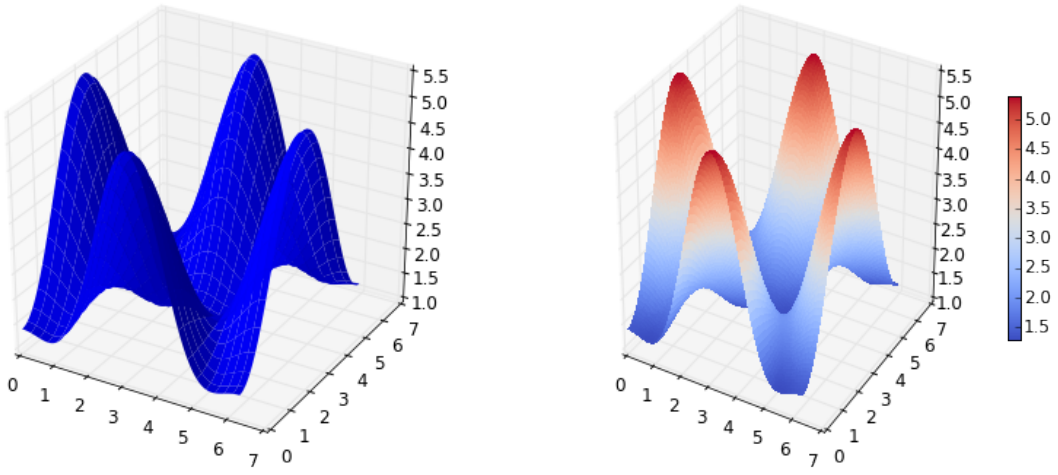


In [60]:
```
fig, ax = plt.subplots()

cnt = ax.contour(Z, cmap=matplotlib.cm.RdBu, vmin=abs(Z).min(), vmax=abs(Z).max(), extent=[0,
```

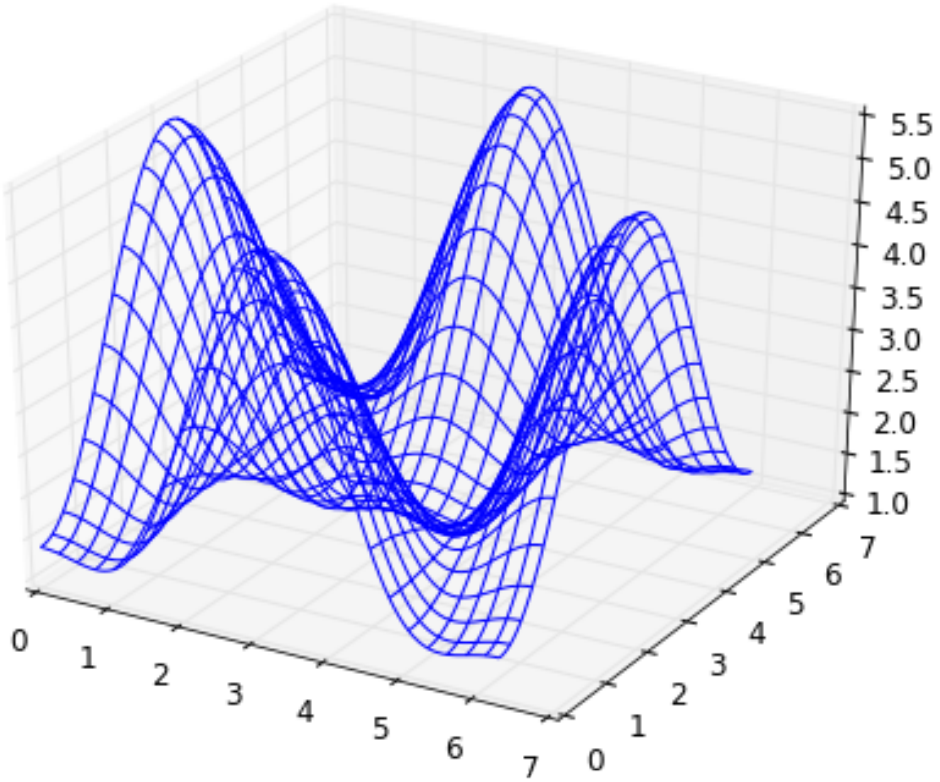


In [61]:
```
from mpl_toolkits.mplot3d.axes3d import Axes3D
```

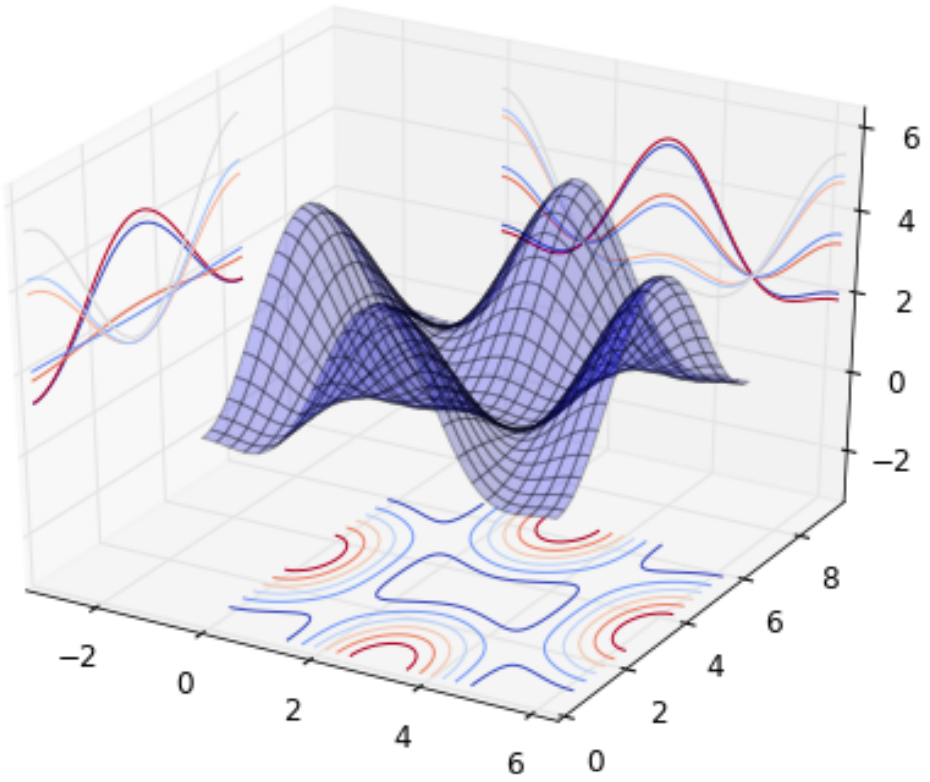
In [62]:
```
fig = plt.figure(figsize=(14,6))

# `ax` is a 3D-aware axis instance because of the projection='3d' keyword argument to add_subp
ax = fig.add_subplot(1, 2, 1, projection='3d')

p = ax.plot_surface(X, Y, Z, rstride=4, cstride=4, linewidth=0)

# surface_plot with color grading and color bar
ax = fig.add_subplot(1, 2, 2, projection='3d')
p = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=matplotlib.cm.coolwarm, linewidth=0, a
cb = fig.colorbar(p, shrink=0.5)
```



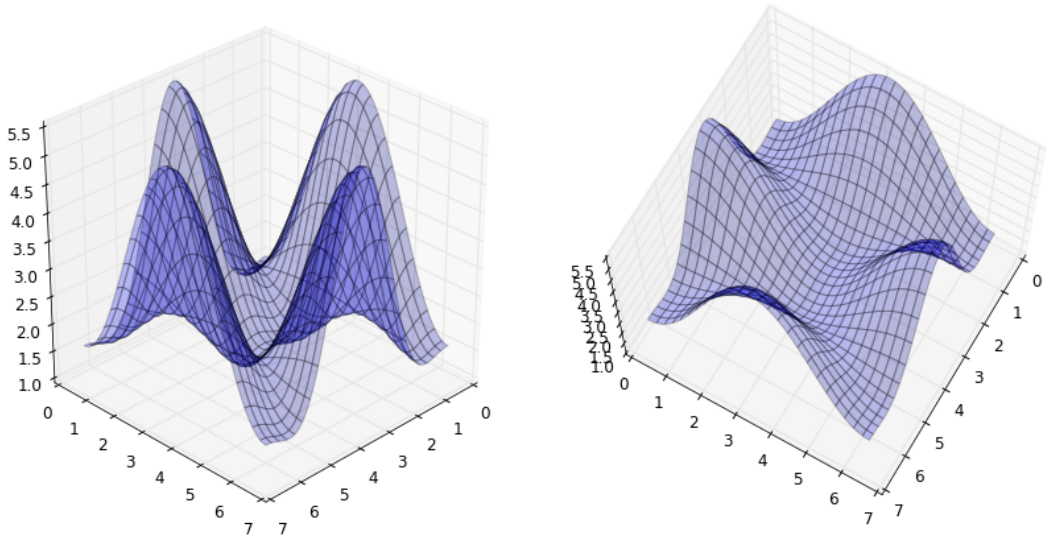
In [63]:
```
fig = plt.figure(figsize=(8,6))

ax = fig.add_subplot(1, 1, 1, projection='3d')

p = ax.plot_wireframe(X, Y, Z, rstride=4, cstride=4)
```



In [64]:
```
fig = plt.figure(figsize=(8,6))

ax = fig.add_subplot(1,1,1, projection='3d')

ax.plot_surface(X, Y, Z, rstride=4, cstride=4, alpha=0.25)
cset = ax.contour(X, Y, Z, zdir='z', offset=-np.pi, cmap=matplotlib.cm.coolwarm)
cset = ax.contour(X, Y, Z, zdir='x', offset=-np.pi, cmap=matplotlib.cm.coolwarm)
cset = ax.contour(X, Y, Z, zdir='y', offset=3*np.pi, cmap=matplotlib.cm.coolwarm)

ax.set_xlim3d(-np.pi, 2*np.pi);
ax.set_ylim3d(0, 3*np.pi);
ax.set_zlim3d(-np.pi, 2*np.pi);
```



In [65]:
```
fig = plt.figure(figsize=(12,6))

ax = fig.add_subplot(1,2,1, projection='3d')
ax.plot_surface(X, Y, Z, rstride=4, cstride=4, alpha=0.25)
ax.view_init(30, 45)

ax = fig.add_subplot(1,2,2, projection='3d')
ax.plot_surface(X, Y, Z, rstride=4, cstride=4, alpha=0.25)
ax.view_init(70, 30)

fig.tight_layout()
```



In [66]:
```
from matplotlib import animation
```

In [67]:
```
# solve the ode problem of the double compound pendulum again

from scipy.integrate import odeint
from numpy import cos, sin

g = 9.82; L = 0.5; m = 0.1

def dx(x, t):
    x1, x2, x3, x4 = x[0], x[1], x[2], x[3]

    dx1 = 6.0/(m*L**2) * (2 * x3 - 3 * cos(x1-x2) * x4)/(16 - 9 * cos(x1-x2)**2)
    dx2 = 6.0/(m*L**2) * (8 * x4 - 3 * cos(x1-x2) * x3)/(16 - 9 * cos(x1-x2)**2)
    dx3 = -0.5 * m * L**2 * ( dx1 * dx2 * sin(x1-x2) + 3 * (g/L) * sin(x1))
    dx4 = -0.5 * m * L**2 * (-dx1 * dx2 * sin(x1-x2) + (g/L) * sin(x2))
    return [dx1, dx2, dx3, dx4]

x0 = [np.pi/2, np.pi/2, 0, 0]  # initial state
t = np.linspace(0, 10, 250) # time coordinates
x = odeint(dx, x0, t)    # solve the ODE problem
```

In [68]:
```
fig, ax = plt.subplots(figsize=(5,5))

ax.set_ylim([-1.5, 0.5])
ax.set_xlim([1, -1])

pendulum1, = ax.plot([], [], color="red", lw=2)
pendulum2, = ax.plot([], [], color="blue", lw=2)

def init():
    pendulum1.set_data([], [])
    pendulum2.set_data([], [])
```

```
        def update(n):
            # n = frame counter
            # calculate the positions of the pendulums
            x1 = + L * sin(x[n, 0])
            y1 = - L * cos(x[n, 0])
            x2 = x1 + L * sin(x[n, 1])
            y2 = y1 - L * cos(x[n, 1])

            # update the line data
            pendulum1.set_data([0 ,x1], [0 ,y1])
            pendulum2.set_data([x1,x2], [y1,y2])

        anim = animation.FuncAnimation(fig, update, init_func=init, frames=len(t), blit=True)

        # anim.save can be called in a few different ways, some which might or might not work
        # on different platforms and with different versions of matplotlib and video encoders
        #anim.save('animation.mp4', fps=20, extra_args=['-vcodec', 'libx264'], writer=animation.FFMpeg
        #anim.save('animation.mp4', fps=20, extra_args=['-vcodec', 'libx264'])
        #anim.save('animation.mp4', fps=20, writer="ffmpeg", codec="libx264")
        anim.save('animation.mp4', fps=20, writer="avconv", codec="libx264")

        plt.close(fig)
```

In [69]:
```
from IPython.display import HTML
video = open("animation.mp4", "rb").read()
video_encoded = video.encode("base64")
video_tag = '<video controls alt="test" src="data:video/x-m4v;base64,{0}">'.format(video_encod
HTML(video_tag)
```

Out[69]: <IPython.core.display.HTML object>

In [70]:
```
print(matplotlib.rcsetup.all_backends)
```

```
[u'GTK', u'GTKAgg', u'GTKCairo', u'MacOSX', u'Qt4Agg', u'Qt5Agg', u'TkAgg', u'WX', u'WXAgg', u'CocoaAgg
```

In [1]:
```
#
# RESTART THE NOTEBOOK: the matplotlib backend can only be selected before pylab is imported!
# (e.g. Kernel > Restart)
#
import matplotlib
matplotlib.use('svg')
import matplotlib.pylab as plt
import numpy
from IPython.display import Image, SVG
```

In [2]:
```
#
# Now we are using the svg backend to produce SVG vector graphics
#
fig, ax = plt.subplots()
t = numpy.linspace(0, 10, 100)
ax.plot(t, numpy.cos(t)*numpy.sin(t))
plt.savefig("test.svg")
```

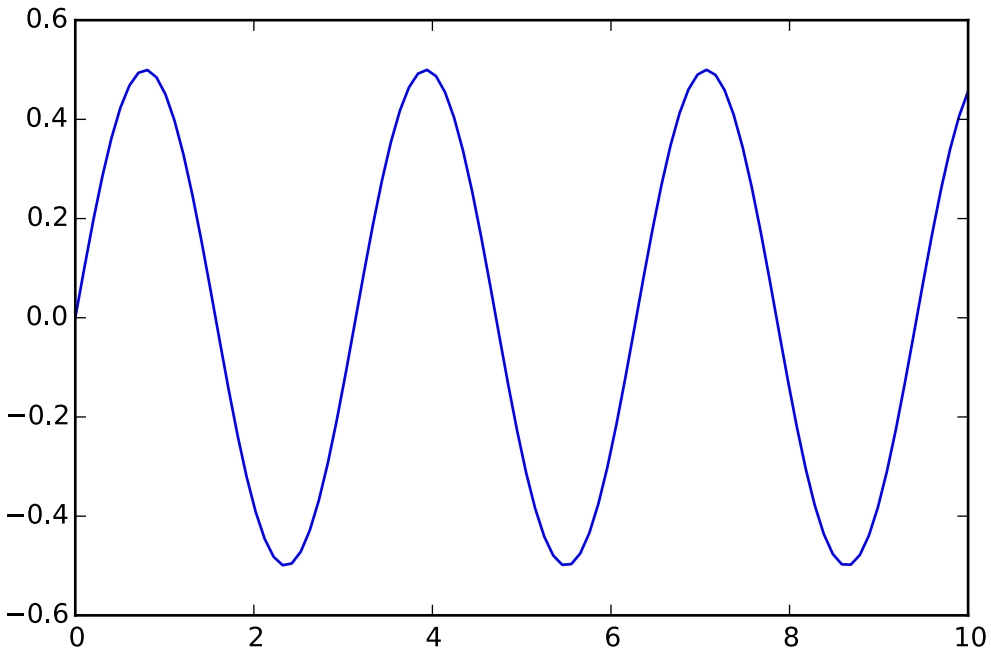
In [3]:
```
#
# Show the produced SVG file.
```

```
        #
        SVG(filename="test.svg")
```

Out[3]:

```
In [1]: %matplotlib inline
        %config InlineBackend.figure_format='svg'

        import matplotlib.pylab as plt
        import numpy
```

```
In [2]: #
        # Now we are using the SVG vector graphics displaced inline in the notebook
        #
        fig, ax = plt.subplots()
        t = numpy.linspace(0, 10, 100)
        ax.plot(t, numpy.cos(t)*numpy.sin(t))
        plt.savefig("test.svg")
```




```
In [1]: #
        # RESTART THE NOTEBOOK: the matplotlib backend can only be selected before pylab is imported!
        # (e.g. Kernel > Restart)
        #
        import matplotlib
        matplotlib.use('Qt4Agg') # or for example MacOSX
        import matplotlib.pylab as plt
        import numpy as np
```

```
In [ ]: # Now, open an interactive plot window with the Qt4Agg backend
        fig, ax = plt.subplots()
```

```
        t = np.linspace(0, 10, 100)
        ax.plot(t, np.cos(t) * np.sin(t))
        plt.show()
```

```
In [1]: %reload_ext version_information
        %version_information numpy, scipy, matplotlib
```

```
        ---------------------------------------------------------------------------

        AttributeError                            Traceback (most recent call last)

        ~\AppData\Roaming\Python\Python38\site-packages\IPython\core\formatters.py in __call__(self, obj)
        343             method = get_real_method(obj, self.print_method)
        344             if method is not None:
    --> 345                 return method()
        346             return None
        347         else:


        c:\program files\python38\lib\site-packages\version_information\version_information.py in _repr_h
        126         html += "<tr><th>Software</th><th>Version</th></tr>"
        127         for name, version in self.packages:
    --> 128             _version = cgi.escape(version)
        129             html += "<tr><td>%s</td><td>%s</td></tr>" % (name, _version)
        130


        AttributeError: module 'cgi' has no attribute 'escape'
```

```
Out[1]:
```

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt

In [2]: from sympy import *

In [3]: init_printing()

        # or with older versions of sympy/ipython, load the IPython extension
        #%load_ext sympy.interactive.ipythonprinting
        # or
        #%load_ext sympyprinting

In [4]: x = Symbol('x')

In [ ]: (pi + x)**2

In [ ]: # alternative way of defining symbols
        a, b, c = symbols("a, b, c")

In [ ]: type(a)

In [ ]: x = Symbol('x', real=True)

In [ ]: x.is_imaginary

In [ ]: x = Symbol('x', positive=True)

In [ ]: x > 0

In [ ]: 1+1*I

In [ ]: I**2

In [ ]: (x * I + 1)**2

In [ ]: r1 = Rational(4,5)
        r2 = Rational(5,4)

In [ ]: r1

In [ ]: r1+r2

In [ ]: r1/r2

In [ ]: pi.evalf(n=50)

In [ ]: y = (x + pi)**2

In [ ]: N(y, 5) # same as evalf

In [ ]: y.subs(x, 1.5)

In [ ]: N(y.subs(x, 1.5))

In [ ]: y.subs(x, a+pi)

In [ ]: import numpy

In [ ]: x_vec = numpy.arange(0, 10, 0.1)
```

```
In [ ]: y_vec = numpy.array([N(((x + pi)**2).subs(x, xx)) for xx in x_vec])

In [ ]: fig, ax = plt.subplots()
        ax.plot(x_vec, y_vec);

In [ ]: f = lambdify([x], (x + pi)**2, 'numpy')  # the first argument is a list of variables that
                                                 # f will be a function of: in this case only x -> f(x)

In [ ]: y_vec = f(x_vec)  # now we can directly pass a numpy array and f(x) is efficiently evaluated

In [ ]: %%timeit

        y_vec = numpy.array([N(((x + pi)**2).subs(x, xx)) for xx in x_vec])

In [ ]: %%timeit

        y_vec = f(x_vec)

In [ ]: (x+1)*(x+2)*(x+3)

In [ ]: expand((x+1)*(x+2)*(x+3))

In [ ]: sin(a+b)

In [ ]: expand(sin(a+b), trig=True)

In [ ]: factor(x**3 + 6 * x**2 + 11*x + 6)

In [ ]: # simplify expands a product
        simplify((x+1)*(x+2)*(x+3))

In [ ]: # simplify uses trigonometric identities
        simplify(sin(a)**2 + cos(a)**2)

In [ ]: simplify(cos(x)/sin(x))

In [ ]: f1 = 1/((a+1)*(a+2))

In [ ]: f1

In [ ]: apart(f1)

In [ ]: f2 = 1/(a+2) + 1/(a+3)

In [ ]: f2

In [ ]: together(f2)

In [ ]: simplify(f2)

In [ ]: y

In [ ]: diff(y**2, x)

In [ ]: diff(y**2, x, x)

In [ ]: diff(y**2, x, 2) # same as above

In [ ]: x, y, z = symbols("x,y,z")
```

```
In [ ]: f = sin(x*y) + cos(y*z)

In [ ]: diff(f, x, 1, y, 2)

In [ ]: f

In [ ]: integrate(f, x)

In [ ]: integrate(f, (x, -1, 1))

In [ ]: integrate(exp(-x**2), (x, -oo, oo))

In [ ]: n = Symbol("n")

In [ ]: Sum(1/n**2, (n, 1, 10))

In [ ]: Sum(1/n**2, (n,1, 10)).evalf()

In [ ]: Sum(1/n**2, (n, 1, oo)).evalf()

In [ ]: Product(n, (n, 1, 10)) # 10!

In [ ]: limit(sin(x)/x, x, 0)

In [ ]: f

In [ ]: diff(f, x)

In [ ]: h = Symbol("h")

In [ ]: limit((f.subs(x, x+h) - f)/h, h, 0)

In [ ]: limit(1/x, x, 0, dir="+")

In [ ]: limit(1/x, x, 0, dir="-")

In [ ]: series(exp(x), x)

In [ ]: series(exp(x), x, 1)

In [ ]: series(exp(x), x, 1, 10)

In [ ]: s1 = cos(x).series(x, 0, 5)
        s1

In [ ]: s2 = sin(x).series(x, 0, 2)
        s2

In [ ]: expand(s1 * s2)

In [ ]: expand(s1.removeO() * s2.removeO())

In [ ]: (cos(x)*sin(x)).series(x, 0, 6)

In [ ]: m11, m12, m21, m22 = symbols("m11, m12, m21, m22")
        b1, b2 = symbols("b1, b2")

In [ ]: A = Matrix([[m11, m12],[m21, m22]])
        A
```

```
In [ ]: b = Matrix([[b1], [b2]])
        b

In [ ]: A**2

In [ ]: A * b

In [ ]: A.det()

In [ ]: A.inv()

In [ ]: solve(x**2 - 1, x)

In [ ]: solve(x**4 - x**2 - 1, x)

In [ ]: solve([x + y - 1, x - y - 1], [x,y])

In [ ]: solve([x + y - a, x - y - c], [x,y])

In [ ]: %reload_ext version_information

        %version_information numpy, matplotlib, sympy
```

```
In [1]: %pylab inline
        from IPython.display import Image

Populating the interactive namespace from numpy and matplotlib


In [2]: Image(filename='images/optimizing-what.png')

Out[2]:

In [3]: %%file hellofortran.f
        C File  hellofortran.f
                subroutine hellofortran (n)
                integer n

                do 100 i=0, n
                    print *, "Fortran says hello"
        100     continue
                end

Overwriting hellofortran.f


In [4]: !f2py -c -m hellofortran hellofortran.f

running build
running config_cc
unifing config_cc, config, build_clib, build_ext, build commands --compiler options
running config_fc
unifing config_fc, config, build_clib, build_ext, build commands --fcompiler options
running build_src
build_src
building extension "hellofortran" sources
f2py options: []
f2py:> /tmp/tmpz2IPjB/src.linux-x86_64-2.7/hellofortranmodule.c
creating /tmp/tmpz2IPjB/src.linux-x86_64-2.7
Reading fortran codes...
        Reading file 'hellofortran.f' (format:fix,strict)
Post-processing...
        Block: hellofortran
                        Block: hellofortran
Post-processing (stage 2)...
Building modules...
        Building module "hellofortran"...
                Constructing wrapper function "hellofortran"...
                  hellofortran(n)
        Wrote C/API module "hellofortran" to file "/tmp/tmpz2IPjB/src.linux-x86_64-2.7/hellofortranmodul
  adding '/tmp/tmpz2IPjB/src.linux-x86_64-2.7/fortranobject.c' to sources.
  adding '/tmp/tmpz2IPjB/src.linux-x86_64-2.7' to include_dirs.
copying /usr/lib/python2.7/dist-packages/numpy/f2py/src/fortranobject.c -> /tmp/tmpz2IPjB/src.linux-x86
copying /usr/lib/python2.7/dist-packages/numpy/f2py/src/fortranobject.h -> /tmp/tmpz2IPjB/src.linux-x86
build_src: building npy-pkg config files
running build_ext
customize UnixCCompiler
customize UnixCCompiler using build_ext
customize Gnu95FCompiler
```

```
Found executable /usr/bin/gfortran
customize Gnu95FCompiler
customize Gnu95FCompiler using build_ext
building 'hellofortran' extension
compiling C sources
C compiler: x86_64-linux-gnu-gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-pr

creating /tmp/tmpz2IPjB/tmp
creating /tmp/tmpz2IPjB/tmp/tmpz2IPjB
creating /tmp/tmpz2IPjB/tmp/tmpz2IPjB/src.linux-x86_64-2.7
compile options: '-I/tmp/tmpz2IPjB/src.linux-x86_64-2.7 -I/usr/lib/python2.7/dist-packages/numpy/core/in
x86_64-linux-gnu-gcc: /tmp/tmpz2IPjB/src.linux-x86_64-2.7/hellofortranmodule.c
In file included from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarraytypes.h:1761:0,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarrayobject.h:17,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/arrayobject.h:4,
                 from /tmp/tmpz2IPjB/src.linux-x86_64-2.7/fortranobject.h:13,
                 from /tmp/tmpz2IPjB/src.linux-x86_64-2.7/hellofortranmodule.c:17:
/usr/lib/python2.7/dist-packages/numpy/core/include/numpy/npy_1_7_deprecated_api.h:15:2: warning: #warnin
 #warning "Using deprecated NumPy API, disable it by " \
  ^
x86_64-linux-gnu-gcc: /tmp/tmpz2IPjB/src.linux-x86_64-2.7/fortranobject.c
In file included from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarraytypes.h:1761:0,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarrayobject.h:17,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/arrayobject.h:4,
                 from /tmp/tmpz2IPjB/src.linux-x86_64-2.7/fortranobject.h:13,
                 from /tmp/tmpz2IPjB/src.linux-x86_64-2.7/fortranobject.c:2:
/usr/lib/python2.7/dist-packages/numpy/core/include/numpy/npy_1_7_deprecated_api.h:15:2: warning: #warnin
 #warning "Using deprecated NumPy API, disable it by " \
  ^
compiling Fortran sources
Fortran f77 compiler: /usr/bin/gfortran -Wall -ffixed-form -fno-second-underscore -fPIC -O3 -funroll-loo
Fortran f90 compiler: /usr/bin/gfortran -Wall -fno-second-underscore -fPIC -O3 -funroll-loops
Fortran fix compiler: /usr/bin/gfortran -Wall -ffixed-form -fno-second-underscore -Wall -fno-second-unde
compile options: '-I/tmp/tmpz2IPjB/src.linux-x86_64-2.7 -I/usr/lib/python2.7/dist-packages/numpy/core/in
gfortran:f77: hellofortran.f
/usr/bin/gfortran -Wall -Wall -shared /tmp/tmpz2IPjB/tmp/tmpz2IPjB/src.linux-x86_64-2.7/hellofortranmodu
Removing build directory /tmp/tmpz2IPjB
```

```
In [5]: %%file hello.py
        import hellofortran

        hellofortran.hellofortran(5)

Overwriting hello.py
```

```
In [6]: # run the script
        !python hello.py

 Fortran says hello

 Fortran says hello

 Fortran says hello
```

```
 Fortran says hello

 Fortran says hello

 Fortran says hello
```

```
In [7]: %%file dprod.f

                subroutine dprod(x, y, n)

                double precision x(n), y
                y = 1.0

                do 100 i=1, n
                    y = y * x(i)
        100     continue
                end
```

```
Overwriting dprod.f
```

```
In [8]: !rm -f dprod.pyf
        !f2py -m dprod -h dprod.pyf dprod.f
```

```
Reading fortran codes...

        Reading file 'dprod.f' (format:fix,strict)

Post-processing...

        Block: dprod

{}

In: :dprod:dprod.f:dprod

vars2fortran: No typespec for argument "n".

                        Block: dprod

Post-processing (stage 2)...

Saving signatures to file "./dprod.pyf"
```

```
In [9]: !cat dprod.pyf
```

```
!    -*- f90 -*-

! Note: the context of this file is case sensitive.
```

```
python module dprod ! in

    interface  ! in :dprod

        subroutine dprod(x,y,n) ! in :dprod:dprod.f

            double precision dimension(n) :: x

            double precision :: y

            integer, optional,check(len(x)>=n),depend(x) :: n=len(x)

        end subroutine dprod

    end interface

end python module dprod



! This file was auto-generated with f2py (version:2).

! See http://cens.ioc.ee/projects/f2py2e/
```

```
In [10]: %%file dprod.pyf
         python module dprod ! in
             interface  ! in :dprod
                 subroutine dprod(x,y,n) ! in :dprod:dprod.f
                     double precision dimension(n), intent(in) :: x
                     double precision, intent(out) :: y
                     integer, optional,check(len(x)>=n),depend(x),intent(in) :: n=len(x)
                 end subroutine dprod
             end interface
         end python module dprod

Overwriting dprod.pyf


In [11]: !f2py -c dprod.pyf dprod.f

running build
running config_cc
unifing config_cc, config, build_clib, build_ext, build commands --compiler options
running config_fc
unifing config_fc, config, build_clib, build_ext, build commands --fcompiler options
running build_src
build_src
building extension "dprod" sources
creating /tmp/tmpWyCvx1/src.linux-x86_64-2.7
f2py options: []
f2py: dprod.pyf
```

```
Reading fortran codes...
        Reading file 'dprod.pyf' (format:free)
Post-processing...
        Block: dprod
                        Block: dprod
Post-processing (stage 2)...
Building modules...
        Building module "dprod"...
                Constructing wrapper function "dprod"...
                  y = dprod(x,[n])
        Wrote C/API module "dprod" to file "/tmp/tmpWyCvx1/src.linux-x86_64-2.7/dprodmodule.c"
  adding '/tmp/tmpWyCvx1/src.linux-x86_64-2.7/fortranobject.c' to sources.
  adding '/tmp/tmpWyCvx1/src.linux-x86_64-2.7' to include_dirs.
copying /usr/lib/python2.7/dist-packages/numpy/f2py/src/fortranobject.c -> /tmp/tmpWyCvx1/src.linux-x86
copying /usr/lib/python2.7/dist-packages/numpy/f2py/src/fortranobject.h -> /tmp/tmpWyCvx1/src.linux-x86
build_src: building npy-pkg config files
running build_ext
customize UnixCCompiler
customize UnixCCompiler using build_ext
customize Gnu95FCompiler
Found executable /usr/bin/gfortran
customize Gnu95FCompiler
customize Gnu95FCompiler using build_ext
building 'dprod' extension
compiling C sources
C compiler: x86_64-linux-gnu-gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-pr

creating /tmp/tmpWyCvx1/tmp
creating /tmp/tmpWyCvx1/tmp/tmpWyCvx1
creating /tmp/tmpWyCvx1/tmp/tmpWyCvx1/src.linux-x86_64-2.7
compile options: '-I/tmp/tmpWyCvx1/src.linux-x86_64-2.7 -I/usr/lib/python2.7/dist-packages/numpy/core/in
x86_64-linux-gnu-gcc: /tmp/tmpWyCvx1/src.linux-x86_64-2.7/dprodmodule.c
In file included from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarraytypes.h:1761:0,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarrayobject.h:17,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/arrayobject.h:4,
                 from /tmp/tmpWyCvx1/src.linux-x86_64-2.7/fortranobject.h:13,
                 from /tmp/tmpWyCvx1/src.linux-x86_64-2.7/dprodmodule.c:18:
/usr/lib/python2.7/dist-packages/numpy/core/include/numpy/npy_1_7_deprecated_api.h:15:2: warning: #warnin
 #warning "Using deprecated NumPy API, disable it by " \
  ^
/tmp/tmpWyCvx1/src.linux-x86_64-2.7/dprodmodule.c:111:12: warning: ‘f2py_size’ defined but not used [-Wu
 static int f2py_size(PyArrayObject* var, ...)
            ^
x86_64-linux-gnu-gcc: /tmp/tmpWyCvx1/src.linux-x86_64-2.7/fortranobject.c
In file included from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarraytypes.h:1761:0,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarrayobject.h:17,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/arrayobject.h:4,
                 from /tmp/tmpWyCvx1/src.linux-x86_64-2.7/fortranobject.h:13,
                 from /tmp/tmpWyCvx1/src.linux-x86_64-2.7/fortranobject.c:2:
/usr/lib/python2.7/dist-packages/numpy/core/include/numpy/npy_1_7_deprecated_api.h:15:2: warning: #warnin
 #warning "Using deprecated NumPy API, disable it by " \
  ^
compiling Fortran sources
Fortran f77 compiler: /usr/bin/gfortran -Wall -ffixed-form -fno-second-underscore -fPIC -O3 -funroll-loo
```

```
Fortran f90 compiler: /usr/bin/gfortran -Wall -fno-second-underscore -fPIC -O3 -funroll-loops
Fortran fix compiler: /usr/bin/gfortran -Wall -ffixed-form -fno-second-underscore -Wall -fno-second-unde
compile options: '-I/tmp/tmpWyCvx1/src.linux-x86_64-2.7 -I/usr/lib/python2.7/dist-packages/numpy/core/in
gfortran:f77: dprod.f
/usr/bin/gfortran -Wall -Wall -shared /tmp/tmpWyCvx1/tmp/tmpWyCvx1/src.linux-x86_64-2.7/dprodmodule.o /t
Removing build directory /tmp/tmpWyCvx1
```

```
In [12]: import dprod

In [13]: help(dprod)

Help on module dprod:

NAME
    dprod

FILE
    /home/rob/Desktop/scientific-python-lectures/dprod.so

DESCRIPTION
    This module 'dprod' is auto-generated with f2py (version:2).
    Functions:
      y = dprod(x,n=len(x))
    .

DATA
    __version__ = '$Revision: $'
    dprod = <fortran object>

VERSION
```

```
In [14]: dprod.dprod(arange(1,50))

Out[14]: 6.082818640342675e+62

In [15]: # compare to numpy
         prod(arange(1.0,50.0))

Out[15]: 6.0828186403426752e+62

In [16]: dprod.dprod(arange(1,10), 5) # only the 5 first elements

Out[16]: 120.0

In [17]: xvec = rand(500)

In [18]: timeit dprod.dprod(xvec)

1000000 loops, best of 3: 882 ns per loop

In [19]: timeit xvec.prod()
```

```
100000 loops, best of 3: 4.45 µs per loop
```

```
In [20]: # simple python algorithm: example of a SLOW implementation
         # Why? Because the loop is implemented in python.
         def py_dcumsum(a):
             b = empty_like(a)
             b[0] = a[0]
             for n in range(1,len(a)):
                 b[n] = b[n-1]+a[n]
             return b
```

```
In [21]: %%file dcumsum.f
         c File dcumsum.f
                subroutine dcumsum(a, b, n)
                double precision a(n)
                double precision b(n)
                integer n
         cf2py  intent(in) :: a
         cf2py  intent(out) :: b
         cf2py  intent(hide) :: n

                b(1) = a(1)
                do 100 i=2, n
                    b(i) = b(i-1) + a(i)
         100    continue
                end
```

```
Overwriting dcumsum.f
```

```
In [22]: !f2py -c dcumsum.f -m dcumsum
```

```
running build
running config_cc
unifing config_cc, config, build_clib, build_ext, build commands --compiler options
running config_fc
unifing config_fc, config, build_clib, build_ext, build commands --fcompiler options
running build_src
build_src
building extension "dcumsum" sources
f2py options: []
f2py:> /tmp/tmpfvrMl6/src.linux-x86_64-2.7/dcumsummodule.c
creating /tmp/tmpfvrMl6/src.linux-x86_64-2.7
Reading fortran codes...
        Reading file 'dcumsum.f' (format:fix,strict)
Post-processing...
        Block: dcumsum
                        Block: dcumsum
Post-processing (stage 2)...
Building modules...
        Building module "dcumsum"...
                Constructing wrapper function "dcumsum"...
                  b = dcumsum(a)
        Wrote C/API module "dcumsum" to file "/tmp/tmpfvrMl6/src.linux-x86_64-2.7/dcumsummodule.c"
```

```
  adding '/tmp/tmpfvrMl6/src.linux-x86_64-2.7/fortranobject.c' to sources.
  adding '/tmp/tmpfvrMl6/src.linux-x86_64-2.7' to include_dirs.
copying /usr/lib/python2.7/dist-packages/numpy/f2py/src/fortranobject.c -> /tmp/tmpfvrMl6/src.linux-x86
copying /usr/lib/python2.7/dist-packages/numpy/f2py/src/fortranobject.h -> /tmp/tmpfvrMl6/src.linux-x86
build_src: building npy-pkg config files
running build_ext
customize UnixCCompiler
customize UnixCCompiler using build_ext
customize Gnu95FCompiler
Found executable /usr/bin/gfortran
customize Gnu95FCompiler
customize Gnu95FCompiler using build_ext
building 'dcumsum' extension
compiling C sources
C compiler: x86_64-linux-gnu-gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-pr

creating /tmp/tmpfvrMl6/tmp
creating /tmp/tmpfvrMl6/tmp/tmpfvrMl6
creating /tmp/tmpfvrMl6/tmp/tmpfvrMl6/src.linux-x86_64-2.7
compile options: '-I/tmp/tmpfvrMl6/src.linux-x86_64-2.7 -I/usr/lib/python2.7/dist-packages/numpy/core/in
x86_64-linux-gnu-gcc: /tmp/tmpfvrMl6/src.linux-x86_64-2.7/dcumsummodule.c
In file included from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarraytypes.h:1761:0,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarrayobject.h:17,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/arrayobject.h:4,
                 from /tmp/tmpfvrMl6/src.linux-x86_64-2.7/fortranobject.h:13,
                 from /tmp/tmpfvrMl6/src.linux-x86_64-2.7/dcumsummodule.c:18:
/usr/lib/python2.7/dist-packages/numpy/core/include/numpy/npy_1_7_deprecated_api.h:15:2: warning: #warnin
 #warning "Using deprecated NumPy API, disable it by " \
  ^
/tmp/tmpfvrMl6/src.linux-x86_64-2.7/dcumsummodule.c:111:12: warning: ‘f2py_size’ defined but not used [-
 static int f2py_size(PyArrayObject* var, ...)
            ^
x86_64-linux-gnu-gcc: /tmp/tmpfvrMl6/src.linux-x86_64-2.7/fortranobject.c
In file included from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarraytypes.h:1761:0,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/ndarrayobject.h:17,
                 from /usr/lib/python2.7/dist-packages/numpy/core/include/numpy/arrayobject.h:4,
                 from /tmp/tmpfvrMl6/src.linux-x86_64-2.7/fortranobject.h:13,
                 from /tmp/tmpfvrMl6/src.linux-x86_64-2.7/fortranobject.c:2:
/usr/lib/python2.7/dist-packages/numpy/core/include/numpy/npy_1_7_deprecated_api.h:15:2: warning: #warnin
 #warning "Using deprecated NumPy API, disable it by " \
  ^
compiling Fortran sources
Fortran f77 compiler: /usr/bin/gfortran -Wall -ffixed-form -fno-second-underscore -fPIC -O3 -funroll-loo
Fortran f90 compiler: /usr/bin/gfortran -Wall -fno-second-underscore -fPIC -O3 -funroll-loops
Fortran fix compiler: /usr/bin/gfortran -Wall -ffixed-form -fno-second-underscore -Wall -fno-second-unde
compile options: '-I/tmp/tmpfvrMl6/src.linux-x86_64-2.7 -I/usr/lib/python2.7/dist-packages/numpy/core/in
gfortran:f77: dcumsum.f
/usr/bin/gfortran -Wall -Wall -shared /tmp/tmpfvrMl6/tmp/tmpfvrMl6/src.linux-x86_64-2.7/dcumsummodule.o
Removing build directory /tmp/tmpfvrMl6
```

```
In [23]: import dcumsum
```

```
In [24]: a = array([1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0])
```

```
In [25]: py_dcumsum(a)
```

```
Out[25]: array([  1.,   3.,   6.,  10.,  15.,  21.,  28.,  36.])

In [26]: dcumsum.dcumsum(a)

Out[26]: array([  1.,   3.,   6.,  10.,  15.,  21.,  28.,  36.])

In [27]: cumsum(a)

Out[27]: array([  1.,   3.,   6.,  10.,  15.,  21.,  28.,  36.])

In [28]: a = rand(10000)

In [29]: timeit py_dcumsum(a)

100 loops, best of 3: 4.83 ms per loop


In [30]: timeit dcumsum.dcumsum(a)

100000 loops, best of 3: 12.2 µs per loop


In [31]: timeit a.cumsum()

10000 loops, best of 3: 27.4 µs per loop


In [32]: %%file functions.c

         #include <stdio.h>

         void hello(int n);

         double dprod(double *x, int n);

         void dcumsum(double *a, double *b, int n);

         void
         hello(int n)
         {
             int i;

             for (i = 0; i < n; i++)
             {
                 printf("C says hello\n");
             }
         }


         double
         dprod(double *x, int n)
         {
             int i;
             double y = 1.0;

             for (i = 0; i < n; i++)
             {
```

```
            y *= x[i];
        }

        return y;
    }

    void
    dcumsum(double *a, double *b, int n)
    {
        int i;

        b[0] = a[0];
        for (i = 1; i < n; i++)
        {
            b[i] = a[i] + b[i-1];
        }
    }
```

```
Overwriting functions.c
```

```
In [33]: !gcc -c -Wall -O2 -Wall -ansi -pedantic -fPIC -o functions.o functions.c
         !gcc -o libfunctions.so -shared functions.o
```

```
In [34]: !file libfunctions.so
```

```
libfunctions.so: ELF 64-bit LSB  shared object, x86-64, version 1 (SYSV), dynamically linked, BuildID[s
```

```
In [35]: %%file functions.py

         import numpy
         import ctypes

         _libfunctions = numpy.ctypeslib.load_library('libfunctions', '.')

         _libfunctions.hello.argtypes = [ctypes.c_int]
         _libfunctions.hello.restype  =  ctypes.c_void_p

         _libfunctions.dprod.argtypes = [numpy.ctypeslib.ndpointer(dtype=numpy.float), ctypes.c_int]
         _libfunctions.dprod.restype  = ctypes.c_double

         _libfunctions.dcumsum.argtypes = [numpy.ctypeslib.ndpointer(dtype=numpy.float), numpy.ctypeslib
         _libfunctions.dcumsum.restype  = ctypes.c_void_p

         def hello(n):
             return _libfunctions.hello(int(n))

         def dprod(x, n=None):
             if n is None:
                 n = len(x)
             x = numpy.asarray(x, dtype=numpy.float)
             return _libfunctions.dprod(x, int(n))
```

```
        def dcumsum(a, n):
            a = numpy.asarray(a, dtype=numpy.float)
            b = numpy.empty(len(a), dtype=numpy.float)
            _libfunctions.dcumsum(a, b, int(n))
            return b
```

```
Overwriting functions.py
```

```
In [36]: %%file run_hello_c.py

         import functions

         functions.hello(3)
```

```
Overwriting run_hello_c.py
```

```
In [37]: !python run_hello_c.py
```

```
C says hello

C says hello

C says hello
```

```
In [38]: import functions

In [39]: functions.dprod([1,2,3,4,5])

Out[39]: 120.0

In [40]: a = rand(100000)

In [41]: res_c = functions.dcumsum(a, len(a))

In [42]: res_fortran = dcumsum.dcumsum(a)

In [43]: res_c - res_fortran

Out[43]: array([ 0.,  0.,  0., ...,  0.,  0.,  0.])

In [44]: timeit functions.dcumsum(a, len(a))

1000 loops, best of 3: 286 µs per loop

In [45]: timeit dcumsum.dcumsum(a)

10000 loops, best of 3: 119 µs per loop

In [46]: timeit a.cumsum()

1000 loops, best of 3: 261 µs per loop
```

In [47]: %%file cy_dcumsum.pyx

```
cimport numpy

def dcumsum(numpy.ndarray[numpy.float64_t, ndim=1] a, numpy.ndarray[numpy.float64_t, ndim=1] b
    cdef int i, n = len(a)
    b[0] = a[0]
    for i from 1 <= i < n:
        b[i] = b[i-1] + a[i]
    return b
```

```
Overwriting cy_dcumsum.pyx
```

In [48]: %%file setup.py

```
from distutils.core import setup
from distutils.extension import Extension
from Cython.Distutils import build_ext

setup(
    cmdclass = {'build_ext': build_ext},
    ext_modules = [Extension("cy_dcumsum", ["cy_dcumsum.pyx"])]
)
```

```
Overwriting setup.py
```

In [49]: !python setup.py build_ext --inplace

```
running build_ext
cythoning cy_dcumsum.pyx to cy_dcumsum.c
warning: /usr/local/lib/python2.7/dist-packages/Cython/Includes/numpy.pxd:869:17: Non-trivial type decla
warning: /usr/local/lib/python2.7/dist-packages/Cython/Includes/numpy.pxd:869:24: Non-trivial type decla
building 'cy_dcumsum' extension
x86_64-linux-gnu-gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-prototypes -fP
In file included from /usr/include/python2.7/numpy/ndarraytypes.h:1761:0,
                 from /usr/include/python2.7/numpy/ndarrayobject.h:17,
                 from /usr/include/python2.7/numpy/arrayobject.h:4,
                 from cy_dcumsum.c:352:
/usr/include/python2.7/numpy/npy_1_7_deprecated_api.h:15:2: warning: #warning "Using deprecated NumPy API
 #warning "Using deprecated NumPy API, disable it by " \
  ^
In file included from /usr/include/python2.7/numpy/ndarrayobject.h:26:0,
                 from /usr/include/python2.7/numpy/arrayobject.h:4,
                 from cy_dcumsum.c:352:
/usr/include/python2.7/numpy/__multiarray_api.h:1629:1: warning: '_import_array' defined but not used [-W
 _import_array(void)
 ^
In file included from /usr/include/python2.7/numpy/ufuncobject.h:327:0,
                 from cy_dcumsum.c:353:
/usr/include/python2.7/numpy/__ufunc_api.h:241:1: warning: '_import_umath' defined but not used [-Wunused
 _import_umath(void)
 ^
x86_64-linux-gnu-gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-Bsymbolic-functions -Wl,-z,r
```

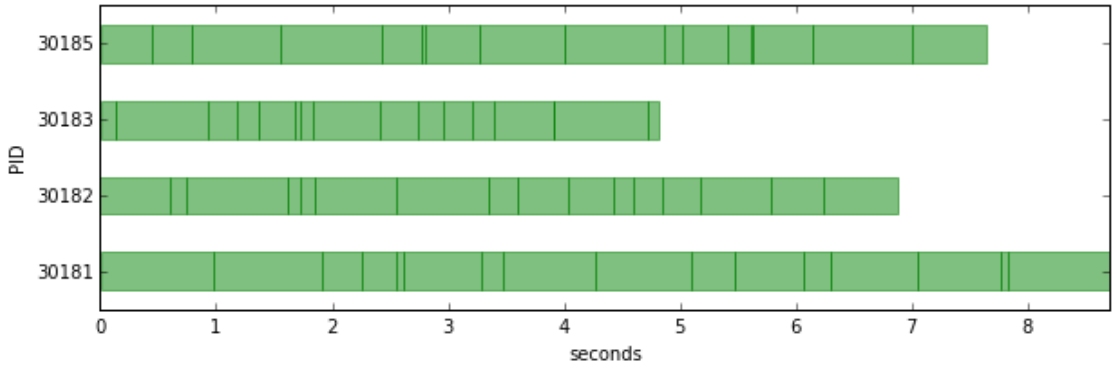
```
In [50]: import cy_dcumsum

In [51]: a = array([1,2,3,4], dtype=float)
         b = empty_like(a)
         cy_dcumsum.dcumsum(a,b)
         b

Out[51]: array([  1.,   3.,   6.,  10.])

In [52]: a = array([1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0])

In [53]: b = empty_like(a)
         cy_dcumsum.dcumsum(a, b)
         b

Out[53]: array([  1.,   3.,   6.,  10.,  15.,  21.,  28.,  36.])

In [54]: py_dcumsum(a)

Out[54]: array([  1.,   3.,   6.,  10.,  15.,  21.,  28.,  36.])

In [55]: a = rand(100000)
         b = empty_like(a)

In [56]: timeit py_dcumsum(a)

10 loops, best of 3: 50.1 ms per loop


In [57]: timeit cy_dcumsum.dcumsum(a,b)

1000 loops, best of 3: 263 µs per loop


In [58]: %load_ext cythonmagic

In [62]: %%cython

         cimport numpy

         def cy_dcumsum2(numpy.ndarray[numpy.float64_t, ndim=1] a, numpy.ndarray[numpy.float64_t, ndim=
             cdef int i, n = len(a)
             b[0] = a[0]
             for i from 1 <= i < n:
                 b[i] = b[i-1] + a[i]
             return b

In [63]: timeit cy_dcumsum2(a,b)

1000 loops, best of 3: 265 µs per loop


In [64]: %reload_ext version_information

         %version_information ctypes, Cython

Out[64]:
```

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt

In [2]: import multiprocessing
        import os
        import time
        import numpy

In [3]: def task(args):
            print("PID =", os.getpid(), ", args =", args)

            return os.getpid(), args

In [4]: task("test")

PID = 28995 , args = test


Out[4]: (28995, 'test')

In [5]: pool = multiprocessing.Pool(processes=4)

In [6]: result = pool.map(task, [1,2,3,4,5,6,7,8])

PID = 29006 , args = 1
PID = 29009 , args = 4
PID = 29007 , args = 2
PID = 29008 , args = 3
PID = 29006 , args = 6
PID = 29009 , args = 5
PID = 29007 , args = 8
PID = 29008 , args = 7


In [7]: result

Out[7]: [(29006, 1),
         (29007, 2),
         (29008, 3),
         (29009, 4),
         (29009, 5),
         (29006, 6),
         (29008, 7),
         (29007, 8)]

In [8]: from IPython.parallel import Client

In [9]: cli = Client()

In [10]: cli.ids

Out[10]: [0, 1, 2, 3]

In [11]: def getpid():
             """ return the unique ID of the current process """
             import os
             return os.getpid()
```

```
In [12]: # first try it on the notebook process
         getpid()

Out[12]: 28995

In [13]: # run it on one of the engines
         cli[0].apply_sync(getpid)

Out[13]: 30181

In [14]: # run it on ALL of the engines at the same time
         cli[:].apply_sync(getpid)

Out[14]: [30181, 30182, 30183, 30185]

In [15]: dview = cli[:]

In [16]: @dview.parallel(block=True)
         def dummy_task(delay):
             """ a dummy task that takes 'delay' seconds to finish """
             import os, time

             t0 = time.time()
             pid = os.getpid()
             time.sleep(delay)
             t1 = time.time()

             return [pid, t0, t1]

In [17]: # generate random delay times for dummy tasks
         delay_times = numpy.random.rand(4)

In [18]: dummy_task.map(delay_times)

Out[18]: [[30181, 1395044753.2096598, 1395044753.9150908],
          [30182, 1395044753.2084103, 1395044753.4959202],
          [30183, 1395044753.2113762, 1395044753.6453338],
          [30185, 1395044753.2130392, 1395044754.1905618]]

In [19]: def visualize_tasks(results):
             res = numpy.array(results)
             fig, ax = plt.subplots(figsize=(10, res.shape[1]))

             yticks = []
             yticklabels = []
             tmin = min(res[:,1])
             for n, pid in enumerate(numpy.unique(res[:,0])):
                 yticks.append(n)
                 yticklabels.append("%d" % pid)
                 for m in numpy.where(res[:,0] == pid)[0]:
                     ax.add_patch(plt.Rectangle((res[m,1] - tmin, n-0.25),
                                  res[m,2] - res[m,1], 0.5, color="green", alpha=0.5))

             ax.set_ylim(-.5, n+.5)
             ax.set_xlim(0, max(res[:,2]) - tmin + 0.)
             ax.set_yticks(yticks)
             ax.set_yticklabels(yticklabels)
             ax.set_ylabel("PID")
             ax.set_xlabel("seconds")
```

```
In [20]: delay_times = numpy.random.rand(64)
```

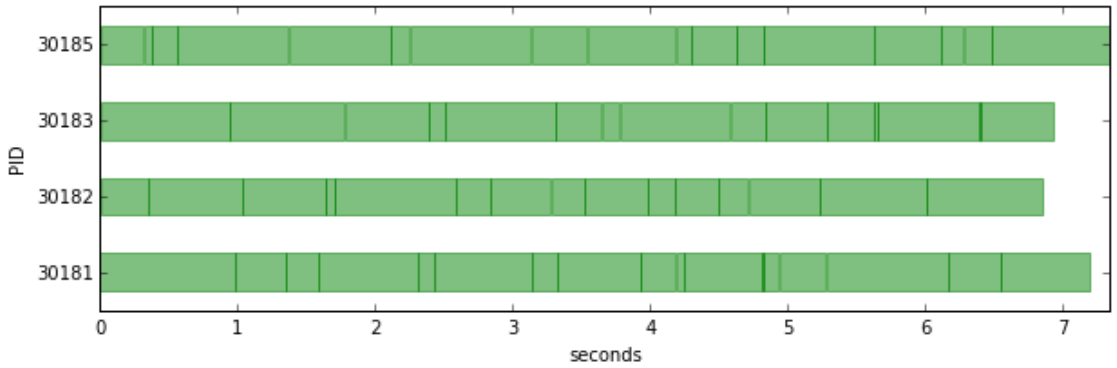
```
In [21]: result = dummy_task.map(delay_times)
         visualize_tasks(result)
```




```
In [22]: lbview = cli.load_balanced_view()
```

```
In [23]: @lbview.parallel(block=True)
         def dummy_task_load_balanced(delay):
             """ a dummy task that takes 'delay' seconds to finish """
             import os, time

             t0 = time.time()
             pid = os.getpid()
             time.sleep(delay)
             t1 = time.time()

             return [pid, t0, t1]
```

```
In [24]: result = dummy_task_load_balanced.map(delay_times)
         visualize_tasks(result)
```




```
In [25]: %%file mpitest.py
```

```
          from mpi4py import MPI

          comm = MPI.COMM_WORLD
          rank = comm.Get_rank()

          if rank == 0:
             data = [1.0, 2.0, 3.0, 4.0]
             comm.send(data, dest=1, tag=11)
          elif rank == 1:
             data = comm.recv(source=0, tag=11)

          print "rank =", rank, ", data =", data

Overwriting mpitest.py


In [26]: !mpirun -n 2 python mpitest.py

rank = 0 , data = [1.0, 2.0, 3.0, 4.0]

rank = 1 , data = [1.0, 2.0, 3.0, 4.0]



In [27]: %%file mpi-numpy-array.py

          from mpi4py import MPI
          import numpy

          comm = MPI.COMM_WORLD
          rank = comm.Get_rank()

          if rank == 0:
             data = numpy.random.rand(10)
             comm.Send(data, dest=1, tag=13)
          elif rank == 1:
             data = numpy.empty(10, dtype=numpy.float64)
             comm.Recv(data, source=0, tag=13)

          print "rank =", rank, ", data =", data

Overwriting mpi-numpy-array.py


In [28]: !mpirun -n 2 python mpi-numpy-array.py

rank = 0 , data = [ 0.71397658  0.37182268  0.25863587  0.08007216  0.50832534  0.80038331

  0.90613024  0.99535428  0.11717776  0.48353805]

rank = 1 , data = [ 0.71397658  0.37182268  0.25863587  0.08007216  0.50832534  0.80038331

  0.90613024  0.99535428  0.11717776  0.48353805]
```

```
In [29]: # prepare some random data
         N = 16
         A = numpy.random.rand(N, N)
         numpy.save("random-matrix.npy", A)
         x = numpy.random.rand(N)
         numpy.save("random-vector.npy", x)

In [30]: %%file mpi-matrix-vector.py

         from mpi4py import MPI
         import numpy

         comm = MPI.COMM_WORLD
         rank = comm.Get_rank()
         p = comm.Get_size()

         def matvec(comm, A, x):
             m = A.shape[0] / p
             y_part = numpy.dot(A[rank * m:(rank+1)*m], x)
             y = numpy.zeros_like(x)
             comm.Allgather([y_part,  MPI.DOUBLE], [y, MPI.DOUBLE])
             return y

         A = numpy.load("random-matrix.npy")
         x = numpy.load("random-vector.npy")
         y_mpi = matvec(comm, A, x)

         if rank == 0:
             y = numpy.dot(A, x)
             print(y_mpi)
             print "sum(y - y_mpi) =", (y - y_mpi).sum()

Overwriting mpi-matrix-vector.py


In [31]: !mpirun -n 4 python mpi-matrix-vector.py

[ 6.40342716  3.62421625  3.42334637  3.99854639  4.95852419  6.13378754

  5.33319708  5.42803442  5.12403754  4.87891654  2.38660728  6.72030412

  4.05218475  3.37415974  3.90903001  5.82330226]

sum(y - y_mpi) = 0.0



In [32]: # prepare some random data
         N = 128
         a = numpy.random.rand(N)
         numpy.save("random-vector.npy", a)

In [33]: %%file mpi-psum.py

         from mpi4py import MPI
```

```
        import numpy as np

        def psum(a):
            r = MPI.COMM_WORLD.Get_rank()
            size = MPI.COMM_WORLD.Get_size()
            m = len(a) / size
            locsum = np.sum(a[r*m:(r+1)*m])
            rcvBuf = np.array(0.0, 'd')
            MPI.COMM_WORLD.Allreduce([locsum, MPI.DOUBLE], [rcvBuf, MPI.DOUBLE], op=MPI.SUM)
            return rcvBuf

        a = np.load("random-vector.npy")
        s = psum(a)

        if MPI.COMM_WORLD.Get_rank() == 0:
            print "sum =", s, ", numpy sum =", a.sum()
```

```
Overwriting mpi-psum.py
```

```
In [34]: !mpirun -n 4 python mpi-psum.py
```

```
sum = 64.948311241 , numpy sum = 64.948311241
```

```
In [35]: N_core = multiprocessing.cpu_count()

         print("This system has %d cores" % N_core)
```

```
This system has 12 cores
```

```
In [36]: %load_ext cythonmagic
```

```
In [37]: %%cython -f -c-fopenmp --link-args=-fopenmp -c-g

         cimport cython
         cimport numpy
         from cython.parallel import prange, parallel
         cimport openmp

         def cy_openmp_test():

             cdef int n, N

             # release GIL so that we can use OpenMP
             with nogil, parallel():
                 N = openmp.omp_get_num_threads()
                 n = openmp.omp_get_thread_num()
                 with gil:
                     print("Number of threads %d: thread number %d" % (N, n))
```

```
In [38]: cy_openmp_test()
```

```
Number of threads 12: thread number 0
Number of threads 12: thread number 10
```

```
Number of threads 12: thread number 8
Number of threads 12: thread number 4
Number of threads 12: thread number 7
Number of threads 12: thread number 3
Number of threads 12: thread number 2
Number of threads 12: thread number 1
Number of threads 12: thread number 11
Number of threads 12: thread number 9
Number of threads 12: thread number 5
Number of threads 12: thread number 6
```

```
In [39]: # prepare some random data
         N = 4 * N_core

         M = numpy.random.rand(N, N)
         x = numpy.random.rand(N)
         y = numpy.zeros_like(x)
```

```
In [40]: %%cython

         cimport cython
         cimport numpy
         import numpy

         @cython.boundscheck(False)
         @cython.wraparound(False)
         def cy_matvec(numpy.ndarray[numpy.float64_t, ndim=2] M,
                       numpy.ndarray[numpy.float64_t, ndim=1] x,
                       numpy.ndarray[numpy.float64_t, ndim=1] y):

             cdef int i, j, n = len(x)

             for i from 0 <= i < n:
                 for j from 0 <= j < n:
                     y[i] += M[i, j] * x[j]

             return y
```

```
In [41]: # check that we get the same results
         y = numpy.zeros_like(x)
         cy_matvec(M, x, y)
         numpy.dot(M, x) - y
```

```
Out[41]: array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])
```

```
In [42]: %timeit numpy.dot(M, x)
```

```
100000 loops, best of 3: 2.93 µs per loop
```

```
In [43]: %timeit cy_matvec(M, x, y)
```

```
100000 loops, best of 3: 5.4 µs per loop
```

```
In [44]: %%cython -f -c-fopenmp --link-args=-fopenmp -c-g

         cimport cython
         cimport numpy
         from cython.parallel import parallel
         cimport openmp

         @cython.boundscheck(False)
         @cython.wraparound(False)
         def cy_matvec_omp(numpy.ndarray[numpy.float64_t, ndim=2] M,
                           numpy.ndarray[numpy.float64_t, ndim=1] x,
                           numpy.ndarray[numpy.float64_t, ndim=1] y):

             cdef int i, j, n = len(x), N, r, m

             # release GIL, so that we can use OpenMP
             with nogil, parallel():
                 N = openmp.omp_get_num_threads()
                 r = openmp.omp_get_thread_num()
                 m = n / N

                 for i from 0 <= i < m:
                     for j from 0 <= j < n:
                         y[r * m + i] += M[r * m + i, j] * x[j]

             return y
```

```
In [45]: # check that we get the same results
         y = numpy.zeros_like(x)
         cy_matvec_omp(M, x, y)
         numpy.dot(M, x) - y
```

```
Out[45]: array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])
```

```
In [46]: %timeit numpy.dot(M, x)
```

```
100000 loops, best of 3: 2.95 µs per loop
```

```
In [47]: %timeit cy_matvec_omp(M, x, y)
```

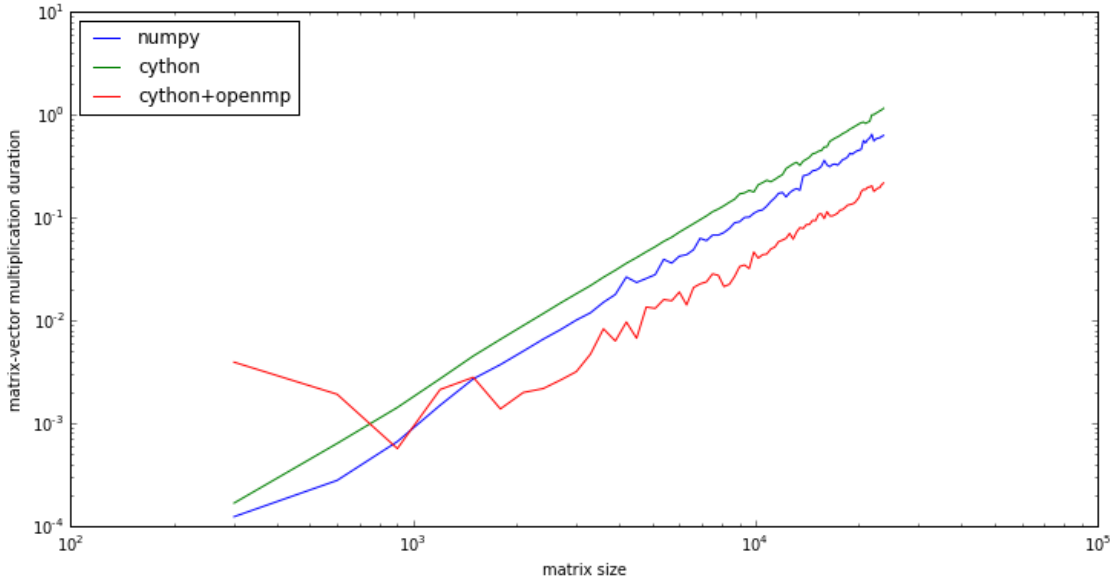
```
1000 loops, best of 3: 209 µs per loop
```

```
In [48]: N_vec  = numpy.arange(25, 2000, 25) * N_core
```

```
In [49]: duration_ref = numpy.zeros(len(N_vec))
         duration_cy = numpy.zeros(len(N_vec))
         duration_cy_omp = numpy.zeros(len(N_vec))
```

```
    for idx, N in enumerate(N_vec):

        M = numpy.random.rand(N, N)
        x = numpy.random.rand(N)
        y = numpy.zeros_like(x)

        t0 = time.time()
        numpy.dot(M, x)
        duration_ref[idx] = time.time() - t0

        t0 = time.time()
        cy_matvec(M, x, y)
        duration_cy[idx] = time.time() - t0

        t0 = time.time()
        cy_matvec_omp(M, x, y)
        duration_cy_omp[idx] = time.time() - t0
```

In [50]:
```
fig, ax = plt.subplots(figsize=(12, 6))

ax.loglog(N_vec, duration_ref, label='numpy')
ax.loglog(N_vec, duration_cy, label='cython')
ax.loglog(N_vec, duration_cy_omp, label='cython+openmp')

ax.legend(loc=2)
ax.set_yscale("log")
ax.set_ylabel("matrix-vector multiplication duration")
ax.set_xlabel("matrix size");
```



In [51]:
```
((duration_ref / duration_cy_omp)[-10:]).mean()
```

Out[51]:
```
3.0072232987815148
```

In [52]: N_core

Out[52]: 12

In [53]: %%file opencl-dense-mv.py

```
import pyopencl as cl
import numpy
import time

# problem size
n = 10000

# platform
platform_list = cl.get_platforms()
platform = platform_list[0]

# device
device_list = platform.get_devices()
device = device_list[0]

if False:
    print("Platform name:" + platform.name)
    print("Platform version:" + platform.version)
    print("Device name:" + device.name)
    print("Device type:" + cl.device_type.to_string(device.type))
    print("Device memory: " + str(device.global_mem_size//1024//1024) + ' MB')
    print("Device max clock speed:" + str(device.max_clock_frequency) + ' MHz')
    print("Device compute units:" + str(device.max_compute_units))

# context
ctx = cl.Context([device]) # or we can use cl.create_some_context()

# command queue
queue = cl.CommandQueue(ctx)

# kernel
KERNEL_CODE = """
//
// Matrix-vector multiplication: r = m * v
//
#define N %(mat_size)d
__kernel
void dmv_cl(__global float *m, __global float *v, __global float *r)
{
    int i, gid = get_global_id(0);

    r[gid] = 0;
    for (i = 0; i < N; i++)
    {
        r[gid] += m[gid * N + i] * v[i];
    }
}
"""
```

```
        kernel_params = {"mat_size": n}
        program = cl.Program(ctx, KERNEL_CODE % kernel_params).build()

        # data
        A = numpy.random.rand(n, n)
        x = numpy.random.rand(n, 1)

        # host buffers
        h_y = numpy.empty(numpy.shape(x)).astype(numpy.float32)
        h_A = numpy.real(A).astype(numpy.float32)
        h_x = numpy.real(x).astype(numpy.float32)

        # device buffers
        mf = cl.mem_flags
        d_A_buf = cl.Buffer(ctx, mf.READ_ONLY | mf.COPY_HOST_PTR, hostbuf=h_A)
        d_x_buf = cl.Buffer(ctx, mf.READ_ONLY | mf.COPY_HOST_PTR, hostbuf=h_x)
        d_y_buf = cl.Buffer(ctx, mf.WRITE_ONLY, size=h_y.nbytes)

        # execute OpenCL code
        t0 = time.time()
        event = program.dmv_cl(queue, h_y.shape, None, d_A_buf, d_x_buf, d_y_buf)
        event.wait()
        cl.enqueue_copy(queue, h_y, d_y_buf)
        t1 = time.time()

        print "opencl elapsed time =", (t1-t0)

        # Same calculation with numpy
        t0 = time.time()
        y = numpy.dot(h_A, h_x)
        t1 = time.time()

        print "numpy elapsed time =", (t1-t0)

        # see if the results are the same
        print "max deviation =", numpy.abs(y-h_y).max()

Overwriting opencl-dense-mv.py


In [54]: !python opencl-dense-mv.py

/usr/local/lib/python2.7/dist-packages/pyopencl-2012.1-py2.7-linux-x86_64.egg/pyopencl/__init__.py:36: Co
  "to see more.", CompilerWarning)
opencl elapsed time = 0.0188570022583
numpy elapsed time = 0.0755031108856
max deviation = 0.0136719


In [55]: %load_ext version_information

        %version_information numpy, mpi4py, Cython

Out[55]:
```

```
In [13]: from IPython.display import Image
```

```
In [4]: # create a new git repository called gitdemo:
        !git init gitdemo
```

```
Reinitialized existing Git repository in /home/rob/Desktop/scientific-python-lectures/gitdemo/.git/
```

```
In [5]: !git clone https://github.com/qutip/qutip
```

```
Cloning into 'qutip'...
remote: Counting objects: 7425, done.
remote: Compressing objects: 100% (2013/2013), done.
remote: Total 7425 (delta 5386), reused 7420 (delta 5381)
Receiving objects: 100% (7425/7425), 2.25 MiB | 696 KiB/s, done.
Resolving deltas: 100% (5386/5386), done.
```

```
In [6]: !git clone gitdemo gitdemo2
```

```
Cloning into 'gitdemo2'...

warning: You appear to have cloned an empty repository.

done.
```

```
In [34]: !git status
```

```
# On branch master

#

# Initial commit

#

# Untracked files:

#   (use "git add <file>..." to include in what will be committed)

#

#        Lecture-7-Revision-Control-Software.ipynb

nothing added to commit but untracked files present (use "git add" to track)
```

```
In [35]: %%file README

         A file with information about the gitdemo repository.
```

```
Writing README
```

```
In [36]: !git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#        Lecture-7-Revision-Control-Software.ipynb
#        README
nothing added to commit but untracked files present (use "git add" to track)


In [37]: !git add README
In [38]: !git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#        new file:   README
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#        Lecture-7-Revision-Control-Software.ipynb
```

```
In [39]: !git commit -m "Added a README file" README
[master (root-commit) 1f26ad6] Added a README file
 1 file changed, 2 insertions(+)
 create mode 100644 README

In [40]: !git add Lecture-7-Revision-Control-Software.ipynb
In [41]: !git commit -m "added notebook file" Lecture-7-Revision-Control-Software.ipynb
[master da8b6e9] added notebook file
 1 file changed, 2047 insertions(+)
 create mode 100644 Lecture-7-Revision-Control-Software.ipynb

In [42]: !git status
# On branch master
nothing to commit (working directory clean)

In [43]: %%file README
         A file with information about the gitdemo repository.
         A new line.
Overwriting README

In [44]: !git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   README
#
no changes added to commit (use "git add" and/or "git commit -a")
```

```
In [45]: !git commit -m "added one more line in README" README
[master b6db712] added one more line in README
 1 file changed, 3 insertions(+), 1 deletion(-)

In [46]: !git status
# On branch master
nothing to commit (working directory clean)

In [47]: %%file tmpfile
         A short-lived file.
Writing tmpfile

In [48]: !git add tmpfile
In [49]: !git commit -m "adding file tmpfile" tmpfile
[master 44ed840] adding file tmpfile
 1 file changed, 2 insertions(+)
 create mode 100644 tmpfile

In [51]: !git rm tmpfile
rm 'tmpfile'

In [52]: !git commit -m "remove file tmpfile" tmpfile
[master a9dc0a4] remove file tmpfile
 1 file changed, 2 deletions(-)
 delete mode 100644 tmpfile

In [53]: !git log
commit a9dc0a4b68e8b1b6d973be8f7e7b8f1c92393c17
Author: Robert Johansson <jrjohansson@gmail.com>
```

```
Date:   Mon Dec 10 06:54:41 2012 +0100

    remove file tmpfile

commit 44ed840422571c62db55eabd8e8768be6c7784e4
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:31 2012 +0100

    adding file tmpfile

commit b6db712506a45a68001c768a6cf6e15e11c62f89
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:26 2012 +0100

    added one more line in README

commit da8b6e92b34fe3838873bdd27a94402ecc121c43
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:20 2012 +0100

    added notebook file

commit 1f26ad648a791e266fbb951ef5c49b8d990e6461
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:19 2012 +0100

    Added a README file
```

```
In [54]: %%file README

         A file with information about the gitdemo repository.

         README files usually contains installation instructions, and information about how to get star
Overwriting README

In [55]: !git diff README

diff --git a/README b/README

index 4f51868..d3951c6 100644

--- a/README

+++ b/README

@@ -1,4 +1,4 @@


 A file with information about the gitdemo repository.


-A new line.

\ No newline at end of file

+README files usually contains installation instructions, and information about how to get started usin
\ No newline at end of file


In [24]: Image(filename='images/github-diff.png')

Out[24]:

In [58]: !git checkout -- README

In [59]: !git status

# On branch master

nothing to commit (working directory clean)


In [60]: !git log
```

```
commit a9dc0a4b68e8b1b6d973be8f7e7b8f1c92393c17
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:41 2012 +0100

    remove file tmpfile

commit 44ed840422571c62db55eabd8e8768be6c7784e4
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:31 2012 +0100

    adding file tmpfile

commit b6db712506a45a68001c768a6cf6e15e11c62f89
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:26 2012 +0100

    added one more line in README

commit da8b6e92b34fe3838873bdd27a94402ecc121c43
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:20 2012 +0100

    added notebook file

commit 1f26ad648a791e266fbb951ef5c49b8d990e6461
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:19 2012 +0100
```

```
    Added a README file
```

```
In [61]: !git checkout 1f26ad648a791e266fbb951ef5c49b8d990e6461
Note: checking out '1f26ad648a791e266fbb951ef5c49b8d990e6461'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 1f26ad6... Added a README file
```

```
In [62]: !cat README

A file with information about the gitdemo repository.
In [63]: !git checkout master
Previous HEAD position was 1f26ad6... Added a README file
Switched to branch 'master'
```

```
In [64]: !cat README

A file with information about the gitdemo repository.

A new line.
```

```
In [65]: !git status
# On branch master
nothing to commit (working directory clean)

In [66]: !git log
commit a9dc0a4b68e8b1b6d973be8f7e7b8f1c92393c17
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:41 2012 +0100

    remove file tmpfile

commit 44ed840422571c62db55eabd8e8768be6c7784e4
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:31 2012 +0100

    adding file tmpfile

commit b6db712506a45a68001c768a6cf6e15e11c62f89
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:26 2012 +0100

    added one more line in README

commit da8b6e92b34fe3838873bdd27a94402ecc121c43
Author: Robert Johansson <jrjohansson@gmail.com>
Date:   Mon Dec 10 06:54:20 2012 +0100

    added notebook file
```

```
commit 1f26ad648a791e266fbb951ef5c49b8d990e6461

Author: Robert Johansson <jrjohansson@gmail.com>

Date:   Mon Dec 10 06:54:19 2012 +0100


    Added a README file


In [67]: !git tag -a demotag1 -m "Code used for this and that purpuse"

In [68]: !git tag -l

demotag1


In [69]: !git show demotag1

tag demotag1

Tagger: Robert Johansson <jrjohansson@gmail.com>

Date:   Mon Dec 10 06:57:25 2012 +0100


Code used for this and that purpuse


commit a9dc0a4b68e8b1b6d973be8f7e7b8f1c92393c17

Author: Robert Johansson <jrjohansson@gmail.com>

Date:   Mon Dec 10 06:54:41 2012 +0100


    remove file tmpfile


diff --git a/tmpfile b/tmpfile

deleted file mode 100644

index ee4c1e7..0000000
```

```
--- a/tmpfile

+++ /dev/null

@@ -1,2 +0,0 @@

-

-A short-lived file.

\ No newline at end of file
```

```
In [70]: !git branch expr1

In [71]: !git branch

  expr1

* master
```

```
In [81]: !git checkout expr1

Switched to branch 'expr1'
```

```
In [74]: %%file README

         A file with information about the gitdemo repository.

         README files usually contains installation instructions, and information about how to get star

         Experimental addition.

Overwriting README
```

```
In [76]: !git commit -m "added a line in expr1 branch" README

[expr1 a6dc24f] added a line in expr1 branch

 1 file changed, 3 insertions(+), 1 deletion(-)
```

```
In [77]: !git branch

* expr1

  master
```

```
In [78]: !git checkout master
Switched to branch 'master'

In [79]: !git branch
  expr1
* master

In [82]: !git checkout master
Switched to branch 'master'

In [83]: !git merge expr1
Updating a9dc0a4..a6dc24f
Fast-forward
 README |    4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

In [84]: !git branch
  expr1
* master

In [85]: !git branch -d expr1
Deleted branch expr1 (was a6dc24f).

In [86]: !git branch
* master

In [88]: !cat README

A file with information about the gitdemo repository.
```

```
README files usually contains installation instructions, and information about how to get started using


Experimental addition.
In [5]: !git remote
origin


In [4]: !git remote show origin
* remote origin
  Fetch URL: git@github.com:jrjohansson/scientific-python-lectures.git
  Push  URL: git@github.com:jrjohansson/scientific-python-lectures.git
  HEAD branch: master
  Remote branch:
    master tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)


In [6]: !git pull origin
Already up-to-date.


In [7]: !git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#        Lecture-7-Revision-Control-Software.ipynb
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

```
In [8]: !git add Lecture-7-Revision-Control-Software.ipynb
```

```
In [9]: !git commit -m "added lecture notebook about RCS" Lecture-7-Revision-Control-Software.ipynb
```

```
[master d0d6a70] added lecture notebook about RCS
 1 file changed, 2114 insertions(+)
 create mode 100644 Lecture-7-Revision-Control-Software.ipynb
```

```
In [11]: !git push
```

```
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 118.94 KiB, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.com:jrjohansson/scientific-python-lectures.git
   2495af4..d0d6a70  master -> master
```

```
In [14]: Image(filename='images/github-project-page.png')
```

```
Out[14]:
```

```
In [15]: Image(filename='images/gitk.png')
```

```
Out[15]:
```