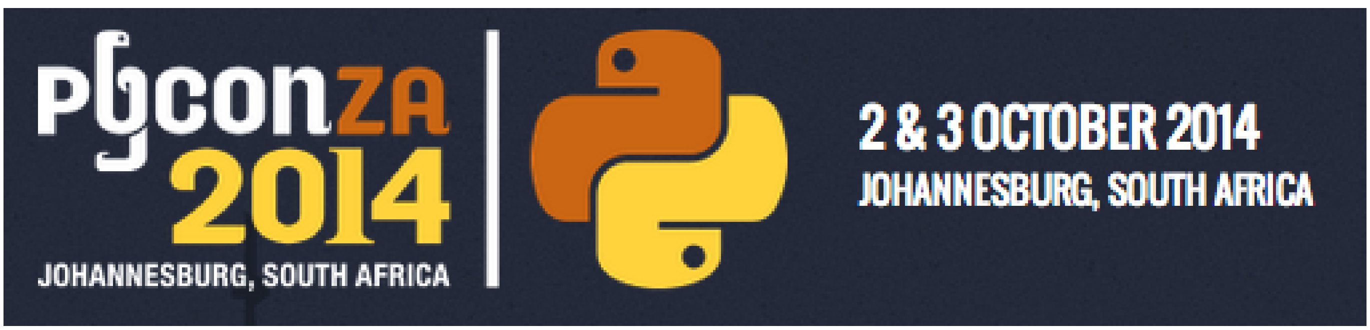
pyconza14

October 7, 2014

0.1 Introduction to IPython Notebook



Tobie Nortje, www.eon.co.za, @tooblippe
tobie.nortje@eon.co.za
http://bit.ly/pyconza-notebook

In [1]:
```
# Load a custom CSS and import plotting
# https://github.com/fperez/pycon2014-keynote

%run static/talktools/talktools.py
#makes sure inline plotting is enabled
```

```
<IPython.core.display.HTML at 0x106031150>
```

In [1]:
```
%pylab inline
#set figure size
figsize(15, 6)

import this
```

```
Populating the interactive namespace from numpy and matplotlib
The Zen of Python, by Tim Peters
```

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

1 ls

- Background and Notebook basics
- Plotting
- Symbolic Mathematics and Typesetting
- Pandas and the Pandas Dataframe
- Quick Machine Learning example
- Publishing Your Work

2 whoami

- Electrical Engineer / Hacker
- Currently busy with M.Eng
- Principal Consultant at Eon Consulting
- Management consultant focus on research, analytics
- Clients Eskom, City Power, Ekhurhuleni, Neotel
- Use Windows, Ubuntu GNU/Linux, Mac, IPhone and Android
- People that I work with uses Excel (*alot*) [**big time**]
- Hack with Python, mySQL, Arduino, Energy Loggers, RaspberryPi
- You will see Python 2.7 here. Almost time to upgrade

3 Python is Popular

4 Practical Use

5 Why Python for Analytics and Visualization?

10 years ago, Python was considered exotic in the analytics space – at best. Languages/packages like R and Matlab dominated the scene. Today, Python has become a major force in data analytics & visualization due to a number of characteristics:

- **Multi-purpose**: prototyping, development, production, systems admin – one for all
- **Libraries**: there is a library for almost any task or problem you face
- **Efficiency**: speeds up IT development tasks for analytics applications
- **Performance**: Python has evolved from a scripting language to a 'meta' language with bridges to all high performance environments (e.g. multi-core CPUs, GPUs, clusters)
- **Interoperability**: Python integrates with almost any other language/ technology
- **Interactivity**: Python allows domain experts to get closer to their business and financial data pools and to do real-time analytics
- **Collaboration**: solutions like Wakari with IPython Notebook allow the easy sharing of code, data, results, graphics, etc.

5.1 The Python Science Stack

- **Python** – the Python interpreter itself
- **NumPy** – high performance, flexible array structures and operations

- **SciPy** – scientific modules and functions (regression, optimization, integration)
- **pandas** – time series and panel data analysis and I/O
- **PyTables** – hierarchical, high performance database (e.g. for out-of-memory analytics)
- **matplotlib** – 2d and 3d visualization
- **IPython** – interactive data analytics, visualization, publishing
- The list is growing...

5.2 IPython Introduction

IPython provides a rich architecture for interactive computing with:

- Powerful interactive shells (terminal and Qt-based)
- Browser-based notebook support for code, text, math expressions, inline plots
- Support for interactive data visualization and use of GUI toolkits
- Easy to use, high performance tools for parallel computing

The main reasons I have been using it includes:

- Plotting is possible in the QT console or the Notebook
- Magic functions makes life easier (magics gets called with a %)
- I also use it as a replacement shell for Windows Shell or Terminal
- Code Completion
- And of course... **Data Analysis**

6 Notebook Basics

The IPython Notebook is a web-based interactive computational environment where you can:

- Combine code execution
- Text
- Mathematics
- Plots and rich media into a single document
- Used to teach classes (Berkley), talks, publish papers etc.

It also features:

- Code Completion
- Help and Docstrings
- Markdown cells for composing documents
- Run it locally or on any webserver with Python installed

Everything you see here is standard Python and Markdown code running in a browser on top of an IPython kernel using Python 2.7 '

6.1 Some Helpful Commands

- **We are now live in an IPython Notebook sessions!**
- ? - Overview of IPython's features
- %quickref - Quick reference.
- help - Python's own help system.
- object? - Inspect an object

7 Let's Get Started

- Each cell is populated with Markdown or Python Code
- This is a markdown cell (Double Click To Reveal)
- The notebook is currently in presentation mode
- Running a cell
- *SHIFT+ENTER* will run the contents of a cell and move to the next one
- *CTRL+ENTER* run the cell in place and don't move to the next cell
- Help
- *CTRL-m h* show keyboard shortcuts
- Save
- At any point, even when the Kernel is busy, you can press *CTRL-S* to save the notebook

In [3]:
```
# press shift-enter to run code

# Create and Set Variable a to the value of 4
a = 4

# Create and Set Variable b to the value of 2
b = 3

print "Hallo PyConZA 2014"
print "a + b =", a+b

print "Now this is weird a/b = ", a/b

b = 3.0 #<---- Be Carefull

print "This is even weirder a/b = ", a/b
```

```
Hallo PyConZA 2014
a + b = 7
Now this is weird a/b =  1
This is even weirder a/b =  1.33333333333
```

7.1 Using the Help System

- The `%quickref` command can be used to obtain a bit more information

In [4]:
```
#IPython -- An enhanced Interactive Python - Quick Reference Card
%quickref   # now press shift-enter
```

7.2 Code Completion and Introspection

- The next cell defines a function with a bit of a long name
- By using the `?` command the docstring can we viewed
- `??` will open up the source code
- The autocomplete function is also demonstrated

In [5]:
```
# lets degine a function with a long name.
def long_silly_dummy_name(a, b):
    """
    This is the docstring for this function
    It takes two arguments a and b
    It returns the sum of a+5 and b*5
    No error checking is done!
    """
    a -= 5
    b *= 5
    return a+b


def long_silly_dummy_name_2(a, b):
    """
    This is the docstring for this function
    It takes two arguments a and b
    It returns the sum of a+5 and b*5
    No error checking is done!
    """
    a += 5
    b *= 5
    return a+b

# again we press SHIFT-Enter
# this will run the function and add it to the namespace

for i in dir():
    if "silly" in i:
        print i
```

```
long_silly_dummy_name
long_silly_dummy_name_2
```

In [6]:
```
# lets get the docstring or some help
long_silly_dummy_name?

# This will open a tab at the bottom of the web page
# You can close it by clicking on the small cross on the right
```

In [7]:
```
#view the source
long_silly_dummy_name??
```

```
In [8]: #press tab to autocplete
        long_silly_dummy_name_2
```

```
Out[8]: <function __main__.long_silly_dummy_name_2>
```

```
In [9]: # press shift-enter to run
        long_silly_dummy_name(5,6)
```

```
Out[9]: 30
```

7.3 Using markdown

- You can set the contents type of a cell in the toolbar above.
- When Markdown is selected you can enter markdown in a cell and its contents will be rendered as HTML.

1. The markdown syntax can be found here

8 This is heading 1

8.1 This is heading 2

8.1.1 This is heading 5

IP[y]:
IPython

IPython
Enhanced Interactive
Console

> Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated.

8.2 IPython and Notebook Magics

- IPython has a set of predefined 'magic functions' that you can call with a command line style syntax
- Think of them as little helper macro's/funcions
- There are two kinds of magics, line-oriented and cell-oriented

- ***Line magics*** are prefixed with the `%` character and work much like OS command-line calls: they get as an argument the rest of the line, where arguments are passed without parentheses or quotes.
- ***Cell magics*** are prefixed with a double `%%`, and they are functions that get as an argument not only the rest of the line, but also the lines below it in a separate argument

8.3 timeit magic

- The `%%timeit` magic can be used to evaluate the average time your loop or piece of code is taking to complete

```
In [10]: %%timeit
         x = 0   # setup
         for i in xrange(100000):
             x = x + i**2
             x = x + i**2
```

```
10 loops, best of 3: 21 ms per loop
```

```
In [11]: %%timeit
         x = 0   # setup#
         for i in xrange(100000):
             x += i**2
```

```
100 loops, best of 3: 11.6 ms per loop
```

8.4 Running Shell Commands

- I now use IPython as my default shell scripting language
- Example - put the contents of the current directory into a list and count the file types
- The `!` before a command indicates that you want to run a system command.

```
In [13]: filelist = !ls static/img                        #read the current directory into variable


         d ={}

         for x,i in enumerate(filelist):

             key = i[ i.find(".")+1:len(i) ]

             if d.has_key(key):
                 d[ i[i.find(".")+1:len(i)] ] += 1

             else:
                 d[i[i.find(".")+1:len(i)]] = 1


         for key in d:
             print key, d[key]
```

```
pptx 1
json 1
db 1
jpg 1
png 16
```

8.5 Embedding Images

- Images can be added using Python code
- Or by adding inline HTML
- This slide shows inline html (double click to show)
- The next slide will use Python syntax to add it

In [14]:
```
from IPython.display import Image
Image('static/img/pyconza.png')
```

Out[14]:

8.6 Adding YouYube videos

- You can also add rich media like YouTube Videos into the notebook

In [15]:
```
#you can embed the HTML or programatically add it.
from IPython.display import YouTubeVideo
YouTubeVideo('iwVvqwLDsJo', width=1000, height=600)
```

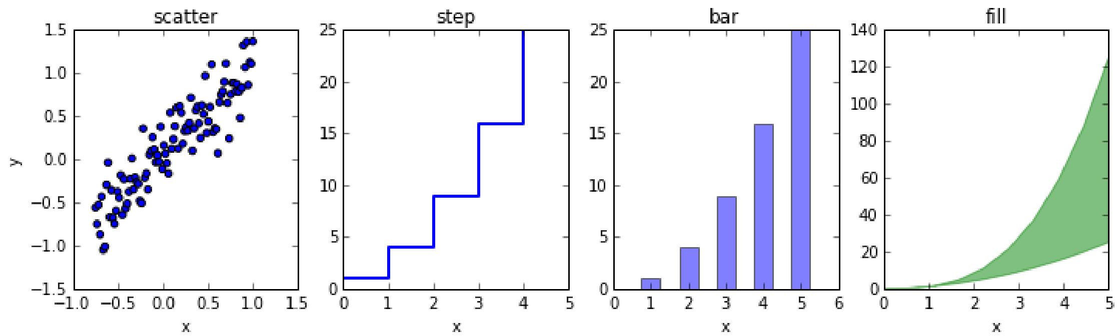
Out[15]:
```
<IPython.lib.display.YouTubeVideo at 0x1061ca6d0>
```

8.7 Plotting with Matplotlib

- Matplotlib is a Python 2D plotting library
- Produces publication quality figures
- Variety of hardcopy formats and interactive environments across platforms
- Matplotlib can be used in
- Python scripts,
- The Python and IPython shell,
- Web application servers,
- Graphical user interface toolkits.

In [16]:
```
from matplotlib.pylab import xkcd
from numpy import *

#generate some data
n = array([0,1,2,3,4,5])
xx = np.linspace(-0.75, 1., 100)
x = linspace(0, 5, 10)
y = x ** 2

fig, axes = plt.subplots(1, 4, figsize=(12,3))
```

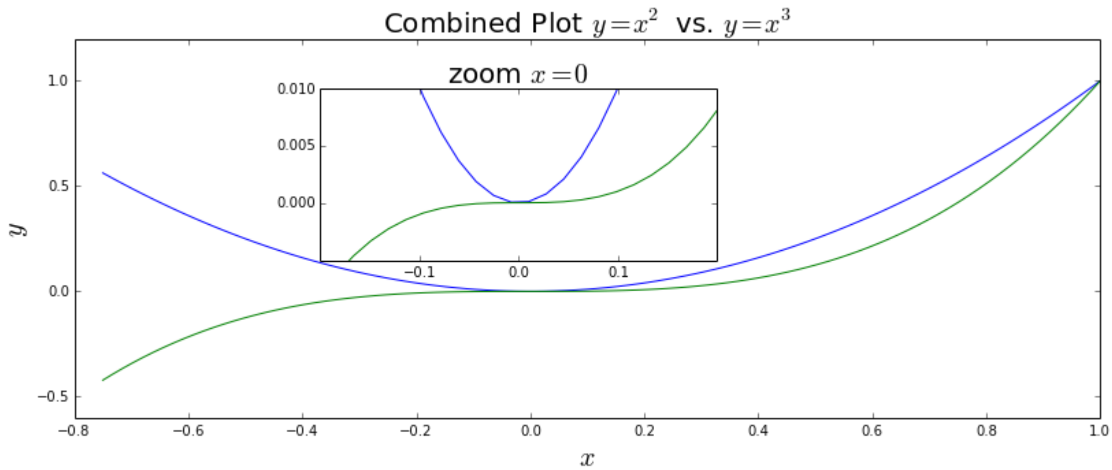
```
axes[0].scatter(xx, xx + 0.25*randn(len(xx)))
axes[0].set_title('scatter')
axes[1].step(n, n**2, lw=2)
axes[1].set_title('step')
axes[2].bar(n, n**2, align="center", width=0.5, alpha=0.5)
axes[2].set_title('bar')
axes[3].fill_between(x, x**2, x**3, color="green", alpha=0.5);
axes[3].set_title('fill')

for i in range(4):
    axes[i].set_xlabel('x')
axes[0].set_ylabel('y')
show()
```



8.8 Combined plots

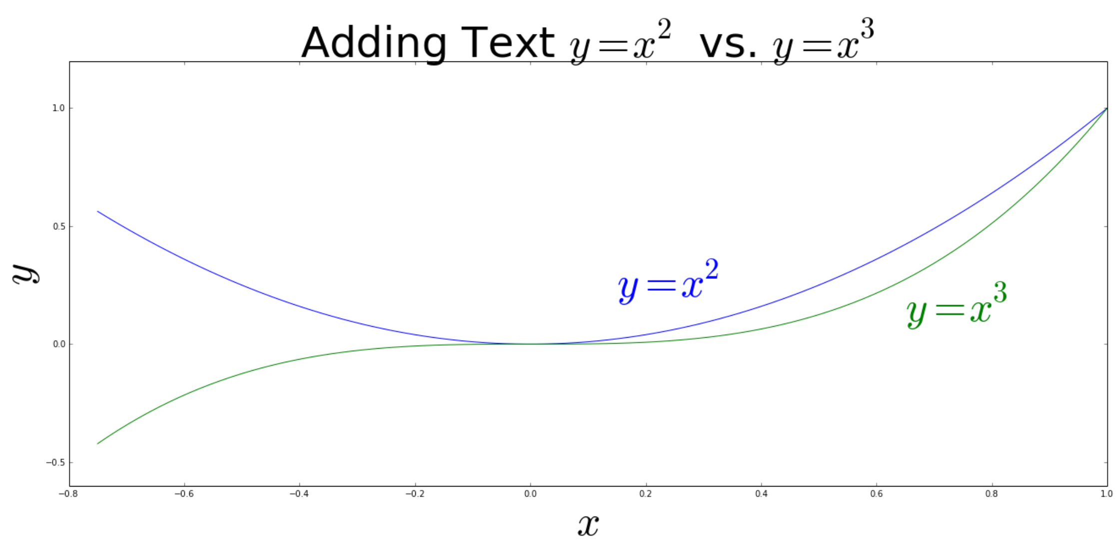
- Matplotlib allows plots to be combined

In [17]:
```
#CustomPlot()
font_size = 20
figsize(11.5, 5)
fig, ax = plt.subplots()

ax.plot(xx, xx**2, xx, xx**3)
ax.set_title(r"Combined Plot $y=x^2$ vs. $y=x^3$", fontsize = font_size)
ax.set_xlabel(r'$x$', fontsize = font_size)
ax.set_ylabel(r'$y$', fontsize = font_size)
fig.tight_layout()
# inset
inset_ax = fig.add_axes([0.29, 0.45, 0.35, 0.35]) # X, Y, width, height
inset_ax.plot(xx, xx**2, xx, xx**3)
inset_ax.set_title(r'zoom $x=0$',fontsize=font_size)
# 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]);
show()
```



8.9 Adding text to a plot

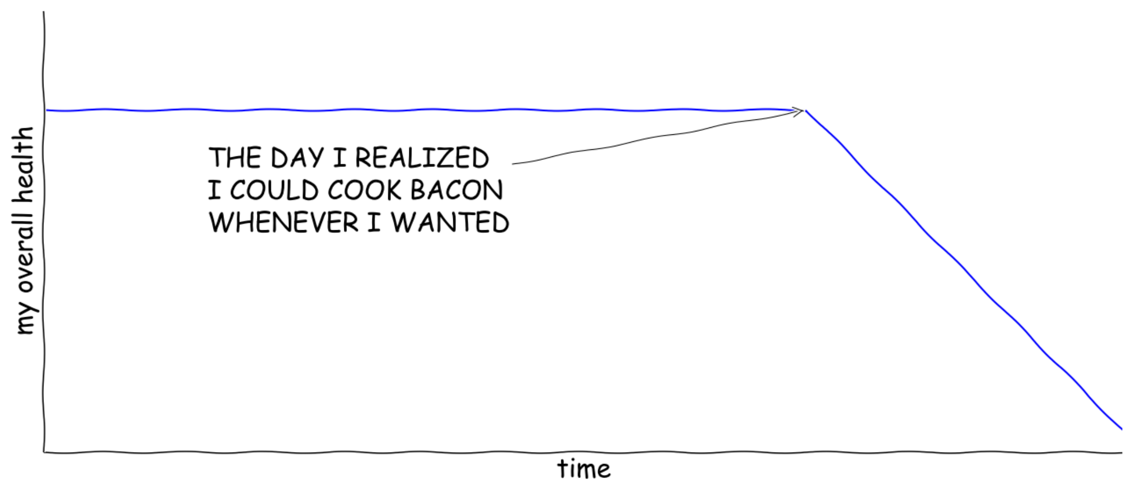
In [18]:
```
#CustomPlot()
figsize(22, 9)
font_size = 50
fig, ax = plt.subplots()
ax.plot(xx, xx**2, xx, xx**3)

ax.set_xlabel(r'$x$', fontsize = font_size)
ax.set_ylabel(r'$y$', fontsize = font_size)
ax.set_title(r"Adding Text $y=x^2$ vs. $y=x^3$", fontsize = font_size)

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

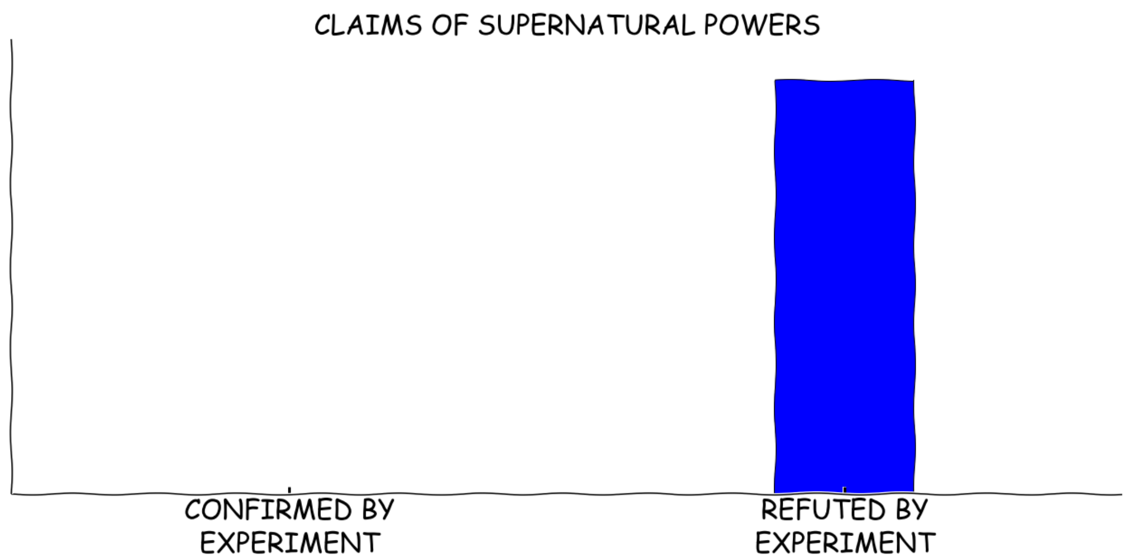


8.10 xkcd style plotting

- matplolib v1.3 now includes a setting to make plots resemble xkcd styles.

In [19]:
```
from matplotlib import pyplot as plt
import numpy as np

fsize = 30

with plt.xkcd():
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    plt.xticks([])
    plt.yticks([])
    ax.set_ylim([-30, 10])

    data = np.ones(100)
    data[70:] -= np.arange(30)

    plt.annotate(
        'THE DAY I REALIZED\nI COULD COOK BACON\nWHENEVER I WANTED',
        xy=(70, 1), arrowprops=dict(arrowstyle='->'), xytext=(15, -10),size=fsize)

    plt.plot(data)

    plt.xlabel('time', size=fsize)
    plt.ylabel('my overall health', size=fsize)

    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    ax.bar([-0.125, 1.0-0.125], [0, 100], 0.25)
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    ax.xaxis.set_ticks_position('bottom')
    ax.set_xticks([0, 1])
    ax.set_xlim([-0.5, 1.5])
    ax.set_ylim([0, 110])
    ax.set_xticklabels(['CONFIRMED BY\nEXPERIMENT', 'REFUTED BY\nEXPERIMENT'],size=fsize)
    plt.yticks([])

    plt.title("CLAIMS OF SUPERNATURAL POWERS",size=fsize)

    plt.show()
```





8.11 Symbolic math using SymPy

- SymPy is a Python library for symbolic mathematics.
- It aims to become a full-featured computer algebra system (CAS) while keeping the code as simple as possible in order to be comprehensible and easily extensible.
- SymPy is written entirely in Python and does not require any external libraries.

In [20]:
```
from sympy import *
init_printing(use_latex=True)
from sympy import solve

x = Symbol('x')
y = Symbol('y')
```

```
series(exp(x), x, 1, 5)
```

Out[20]:

$$e + ex + \frac{ex^{2}}{2} + \frac{ex^{3}}{6} + \frac{ex^{4}}{24} + \mathcal{O}\left(x^{5}\right)$$

```
In [21]: eq = ((x+y)**2 * (x+1))

         eq
```

Out[21]:

$$\left(x + 1\right) \left(x + y\right)^{2}$$

```
In [22]: solve(eq)
```

```
/Users/tobie/anaconda/lib/python2.7/site-packages/IPython/core/formatters.py:239: FormatterWarning: Exc
\begin{bmatrix}\begin{Bmatrix}x : -1\end{Bmatrix}, & \begin{Bmatrix}x : - y\end{Bmatrix}\end{bmatrix}
^
Unknown symbol: \begin (at char 0), (line:1, col:1)
  FormatterWarning,
```

Out[22]:

$$\left [\left \{ x : -1\right \}, \quad \left \{ x : - y\right \}\right]$$

```
In [23]: a = 1/x + (x*sin(x) - 1)/x

         a
```

Out[23]:

$$\frac{1}{x} \left(x \sin{\left (x \right)} - 1\right) + \frac{1}{x}$$

```
In [24]: simplify(a)
```

Out[24]:

$$\sin{\left (x \right)}$$

```
In [25]: integrate(x**2 * exp(x) * cos(x), x)
```

Out[25]:

$$\frac{x^{2} e^{x}}{2} \sin{\left (x \right)} + \frac{x^{2} e^{x}}{2} \cos{\left (x \right)} - x e^{x} \sin{\left (x \right)} + \frac{e^{x}}{2} \sin{\left (x \right)} - \frac{e^{x}}{2} \cos{\left (x \right)}$$

8.12 Data Analysis Using the Pandas Library

- pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive
- The two primary data structures of pandas
- Series (1-dimensional) and
- DataFrame (2-dimensional) **Think Spreadhseet, Timeseries**

Handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

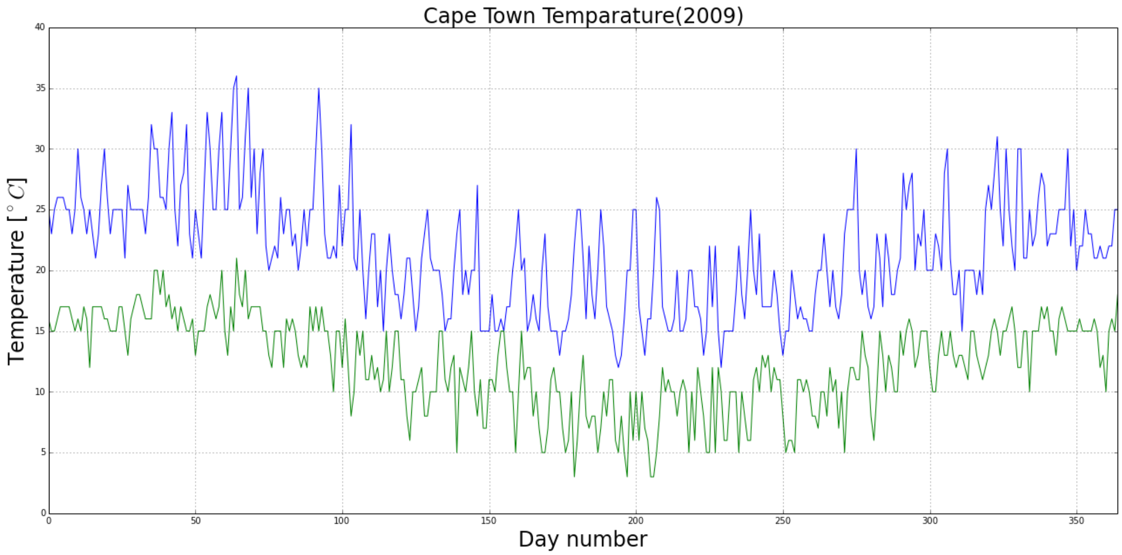
- For R users, DataFrame provides everything that R's data.frame provides.
- pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

In [26]:
```
from pandas import DataFrame, read_csv

Cape_Weather = DataFrame( read_csv('static/data/CapeTown_2009_Temperatures.csv' ))

Cape_Weather.head(10)
```

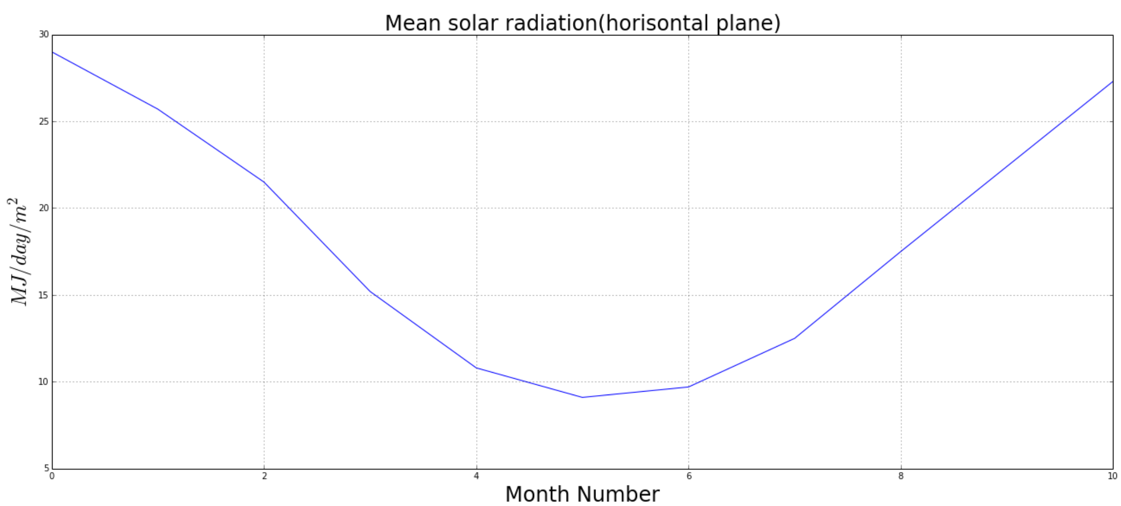
Out[26]:

	high	low	radiation
0	25	16	29.0
1	23	15	25.7
2	25	15	21.5
3	26	16	15.2
4	26	17	10.8
5	26	17	9.1
6	25	17	9.7
7	25	17	12.5
8	23	16	17.5
9	25	15	22.4

In [27]:
```
#CustomPlot()
figsize(22, 10)
font_size = 24

title('Cape Town Temparature(2009)',fontsize = font_size)
xlabel('Day number',fontsize = font_size)
ylabel(r'Temperature [$^\circ C$] ',fontsize = font_size)

Cape_Weather.high.plot()
Cape_Weather.low.plot()
show()
```




```
In [28]: #CustomPlot()
         figsize(22, 9)
         font_size = 24

         title( 'Mean solar radiation(horisontal plane)', fontsize=font_size)
         xlabel('Month Number', fontsize = font_size)
         ylabel(r'$MJ / day / m^2$',fontsize = font_size)

         Cape_Weather.radiation[0:11].plot()
         show()
```




```
In [29]: # lets look at a proxy for heating degree and cooling degree days

         level = 25
```

```
        print '>25 =', Cape_Weather[ Cape_Weather['high'] > level  ].count()['high']
        print '<=25 =', Cape_Weather[ Cape_Weather['high'] <= level ].count()['high']
```

```
>25 = 59
<=25 = 306
```

```
In [30]: # Basic descriptive statistics
         print Cape_Weather.describe()
```

```
high          low  radiation
count  365.000000  365.000000   12.00000
mean    21.545205   12.290411   19.15000
std      4.764943    3.738431    7.65156
min     12.000000    3.000000    9.10000
25%     18.000000   10.000000   12.07500
50%     21.000000   12.000000   19.50000
75%     25.000000   15.000000   26.10000
max     36.000000   21.000000   29.10000
```

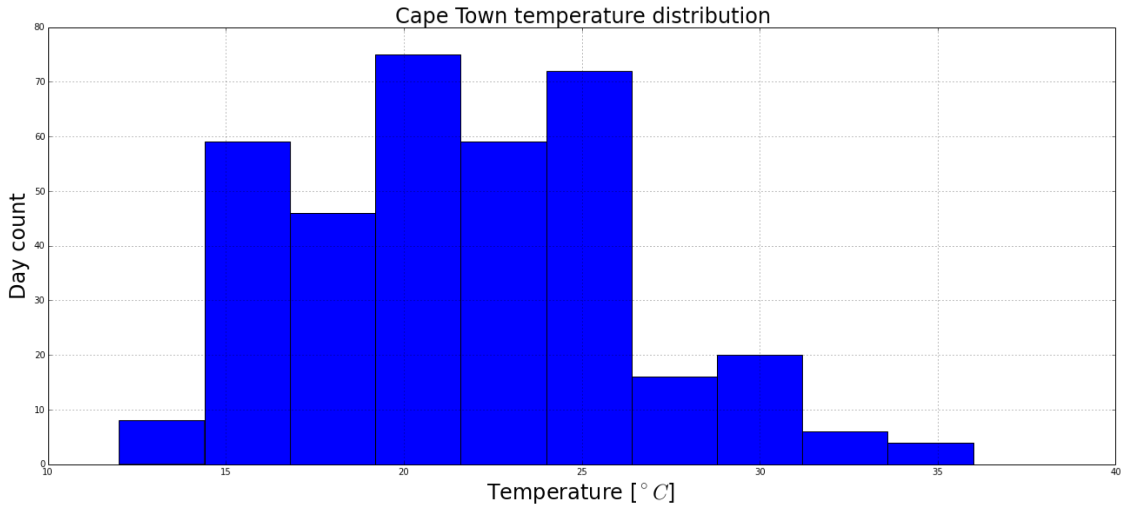
9 Run a SQL query on a dataframe!

- the pandasql module allows you to query the df as if it is a sqlite database

```
In [31]: from pandasql import sqldf

         #helper function to extract df from the globals list
         pysqldf = lambda q: sqldf(q, globals())

         temp_range = pysqldf( '''
                                  select
                                      count(high) Count,
                                      (high-low) T_spread

                                  from
                                     Cape_Weather

                                  where
                                     high > 25 and
                                     low < 10''' )
         temp_range
```

```
Out[31]:    Count  T_spread
         0      3        21
```

```
In [32]: #CustomPlot()
         figsize(22, 9)
         font_size = 24
         title('Cape Town temperature distribution', fontsize=font_size)
         ylabel('Day count',fontsize = font_size)
         xlabel(r'Temperature [$^\circ C $] ',fontsize = font_size)
         Cape_Weather['high'].hist(bins=10)
         show()
```



9.1 Typesetting

9.1.1 LaTex

- LaTex is rendered using the mathjax.js JavaScript library

```
In [33]: from IPython.display import Math

         Math(r'F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx')
```

Out[33]:

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx$$

```
In [34]: from IPython.display import Latex
         Latex(r"""\begin{eqnarray}
         \nabla \times \vec{\mathbf{B}} -\, \frac1c\, \frac{\partial\vec{\mathbf{E}}}{\partial t} & = \
         \nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho \\
         \nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\mathbf{B}}}{\partial t} & =
         \nabla \cdot \vec{\mathbf{B}} & = 0
         \end{eqnarray}""")
```

Out[34]:

$$\nabla \times \vec{\mathbf{B}} - \frac{1}{c} \frac{\partial\vec{\mathbf{E}}}{\partial t} = \frac{4\pi}{c}\vec{\mathbf{j}} \quad (1)$$

$$\nabla \cdot \vec{\mathbf{E}} = 4\pi\rho \quad (2)$$

$$\nabla \times \vec{\mathbf{E}} + \frac{1}{c} \frac{\partial\vec{\mathbf{B}}}{\partial t} = \vec{\mathbf{0}} \quad (3)$$

$$\nabla \cdot \vec{\mathbf{B}} = 0 \quad (4)$$

9.2 Saving a Gist

- It is possible to save specific lines of code to a GitHub gist.
- This is achieved with the `%pastebin` magic as demonstrated below.
- This makes sharing code easy!

```
In [35]: %pastebin "cell one" 0-10
```

```
Out[35]: u'https://gist.github.com/142a1ff6555516d47687'
```

9.3 Connect to this kernel remotely

- Using the *%connect*info_ magic you can obtain the connection info to connect to this workbook from another IPython console or qtconsole using :
- ipython qtconsole –existing
- Using the port, signature and key you can also connect to a remote IPython kernel via SSH

```
In [36]: %connect_info
         # lets connect to this kernel using the command "ipython qtconsole --existing"
```


```
{
  "stdin_port": 51938,
  "ip": "127.0.0.1",
  "control_port": 51939,
  "hb_port": 51940,
  "signature_scheme": "hmac-sha256",
  "key": "df51596c-add0-44b5-8485-44fe2356b32c",
  "shell_port": 51936,
  "transport": "tcp",
  "iopub_port": 51937
}
```


```
Paste the above JSON into a file, and connect with:
    $> ipython <app> --existing <file>
or, if you are local, you can connect with just:
    $> ipython <app> --existing kernel-4a0ac3b5-fce6-444b-aa6c-d2ac4c49d2d0.json
or even just:
    $> ipython <app> --existing
if this is the most recent IPython session you have started.
```

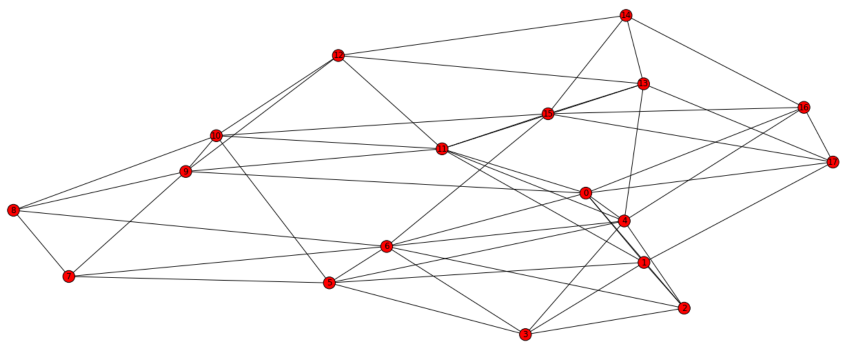
10 Adding Interactivity

- IPython includes an architecture for interactive widgets
- Ties together Python code running in the kernel and JavaScript/HTML/CSS in the browser
- These widgets enable users to explore their code and data interactively

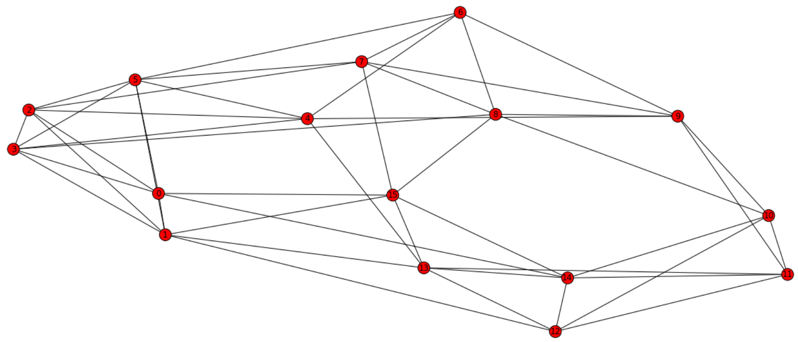
```
In [37]: from IPython.html.widgets import interact
         import matplotlib.pyplot as plt
         import networkx as nx

         # wrap a few graph generation functions so they have the same signature

         def random_lobster(n, m, k, p):
             return nx.random_lobster(n, p, p / m)
```

```
def powerlaw_cluster(n, m, k, p):
    return nx.powerlaw_cluster_graph(n, m, p)

def erdos_renyi(n, m, k, p):
    return nx.erdos_renyi_graph(n, p)

def newman_watts_strogatz(n, m, k, p):
    return nx.newman_watts_strogatz_graph(n, k, p)

def plot_random_graph(n, m, k, p, generator):
    g = generator(n, m, k, p)
    nx.draw(g)
    plt.show()
```

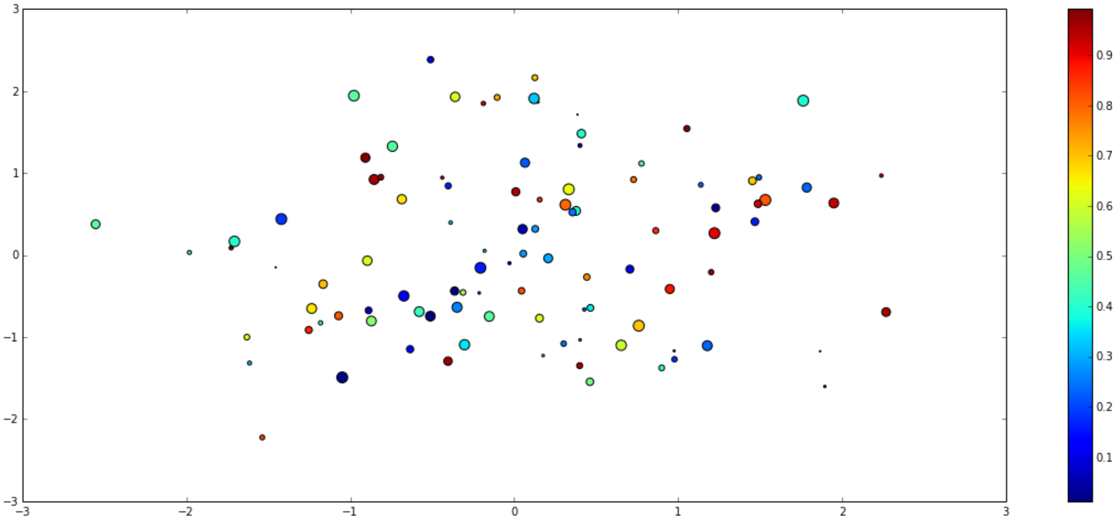
In [38]:
```
#static
plot_random_graph( 18, 5, 5, 0.449, newman_watts_strogatz)
```



In [39]:
```
interact(
        plot_random_graph, n=(2,30), m=(1,10), k=(1,10), p=(0.0, 1.0, 0.001),
        generator={'lobster': random_lobster,
                   'power law': powerlaw_cluster,
                   'Newman-Watts-Strogatz': newman_watts_strogatz,
                   u'Erdős-Rényi': erdos_renyi,
                   });
```



```
In [40]: from IPython.html.widgets import interact
         import matplotlib.pyplot as plt
         import numpy as np

         %matplotlib inline

         np.random.seed(0)
         x, y = np.random.normal(size=(2, 100))
         s, c = np.random.random(size=(2, 100))

         def draw_scatter(size=100, cmap='jet', alpha=1.0):
             fig, ax = plt.subplots(figsize=(20, 8))
             points = ax.scatter(x, y, s=size*s, c=c, alpha=alpha, cmap=cmap)
             fig.colorbar(points, ax=ax)
             return fig

         colormaps = sorted(m for m in plt.cm.datad if not m.endswith("_r"))

In [41]: interact(
                     draw_scatter, size=[0, 2000], alpha=[0.0, 1.0], cmap=colormaps
                 );
```



11 Exploring Beat Frequencies using the Audio Object

- This example uses the Audio object and Matplotlib to explore the phenomenon of beat frequencies.

In [42]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

from IPython.html.widgets import interactive
from IPython.display import Audio, display
import numpy as np

def beat_freq(f1=220.0, f2=224.0):
    max_time = 3
    rate = 8000
    times = np.linspace(0,max_time,rate*max_time)
    signal = np.sin(2*np.pi*f1*times) + np.sin(2*np.pi*f2*times)
    print(f1, f2, abs(f1-f2))
    display(Audio(data=signal, rate=rate))
    return signal
```

In [43]:
```
v = interactive(beat_freq, f1=(200.0,300.0), f2=(200.0,300.0))
display(v)
```

```
(220.0, 224.0, 4.0)

<IPython.lib.display.Audio at 0x10cccbe90>
```

11.1 Lorenz System of Differential Equations

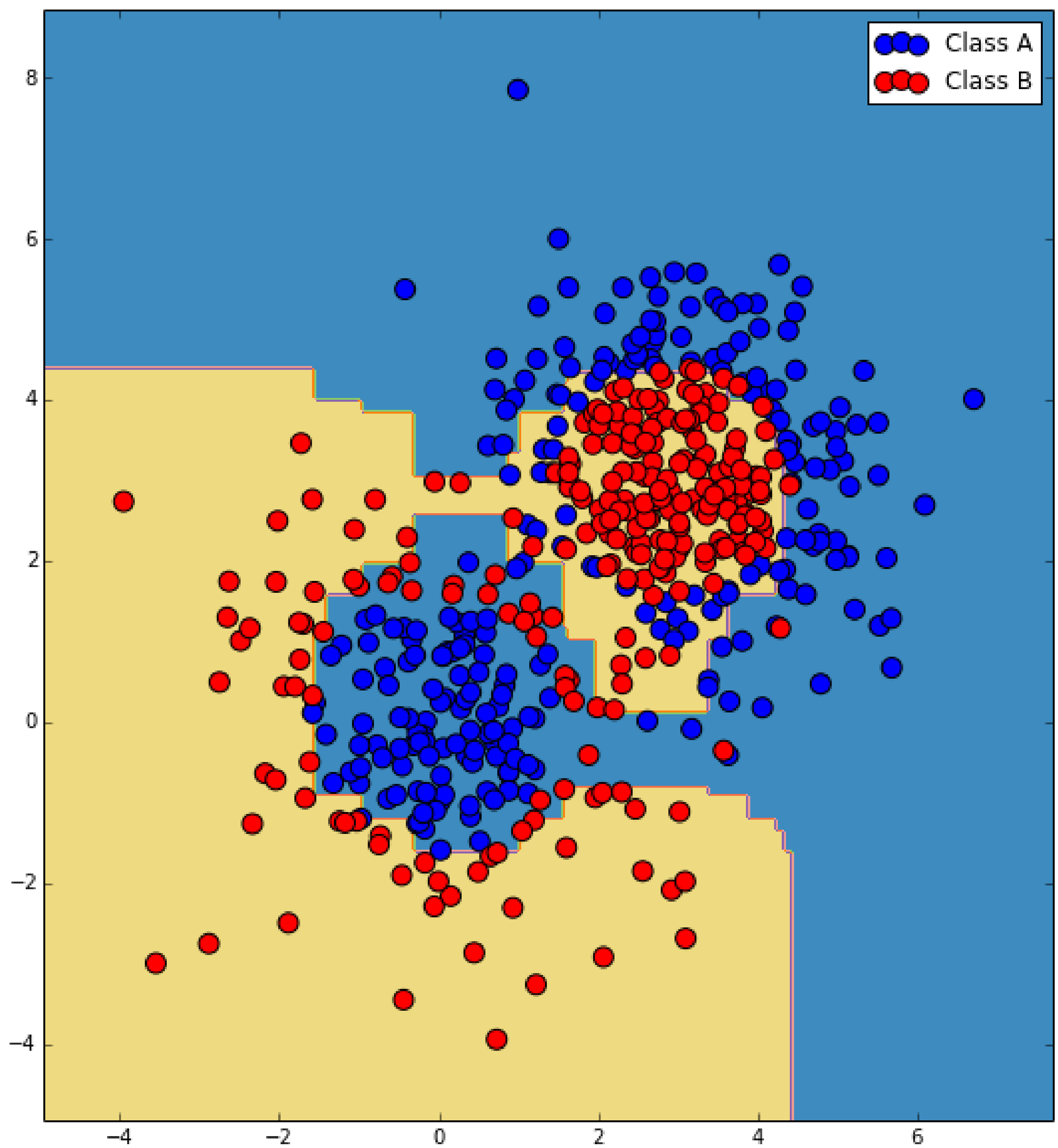
- Also example to link to other notebooks
- Click Lorenz Differential Equations

12 Machine Learning with Sci-Kit Learn

In [44]:
```
solve = True

import pylab as pl
import numpy as np

from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_gaussian_quantiles


# Construct dataset
X1, y1 = make_gaussian_quantiles(cov=2.,
                                 n_samples=200, n_features=2,
                                 n_classes=2, random_state=1)
X2, y2 = make_gaussian_quantiles(mean=(3, 3), cov=1.5,
                                 n_samples=300, n_features=2,
                                 n_classes=2, random_state=1)
#Training Samples
X = np.concatenate((X1, X2))

#Training Target
y = np.concatenate((y1, - y2 + 1))

# Create and fit an AdaBoosted decision tree
bdt = AdaBoostClassifier(
                          DecisionTreeClassifier( max_depth=1),
                                                  algorithm="SAMME",
                                                  n_estimators=200
                        )

bdt.fit(X, y)

plot_colors = "br"
plot_step = 0.05
class_names = "AB"

pl.figure(figsize=(20, 10))

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

if solve:
    # Plot the decision boundaries
    pl.subplot(121)

    xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
                         np.arange(y_min, y_max, plot_step))

    Z = bdt.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)
    cs = pl.contourf(xx, yy, Z, cmap=pl.cm.Paired)
```

```
            pl.axis("tight")


        # Plot the training points
        for i, n, c in zip(range(2), class_names, plot_colors):
            idx = np.where(y == i)
            pl.scatter(X[idx, 0], X[idx, 1],
                       c=c, cmap=pl.cm.Paired,
                       label="Class %s" % n, s=100)
        pl.xlim(x_min, x_max)
        pl.ylim(y_min, y_max)
        pl.legend(loc='upper right')
        pl.xlabel("Decision Boundary",size = 20)
```

```
Out[44]: <matplotlib.text.Text at 0x10e526810>
```



Decision Boundary

12.1 Publishing your Work

- Ability to convert an .ipynb notebook document file into various static formats.
- Currently, nbconvert is provided as a command line tool, run as a script using IPython.

This page is converted and published to the following formats using this tool:

- HTML
- PDF (the PDF is created using wkhtml2pdf that takes the html file as an input)

- LATEX
- Reveal.js slideshow

```
In [ ]: !ipython nbconvert pyconza_ipython.ipynb --to html --post serve
```

13 Writing Books!

- Need internet connection but check out
- Probabilistic-Programming-and-Bayesian-Methods-for-Hackers

14 THANK YOU FOR YOUR TIME

- Find all of this - http://bit.ly/pyconza-notebook
- Join the Gauteng Python Users Group - GPUG
- ask @tooblippe, @wasbeer
  - meetup.com http://www.meetup.com/Gauteng-Python-Users-Group/
  - Google groups #gpugsa
  - website - http://gautengpug.github.io/
  - github - http://github.com/gautengpug
- Python, DataAnalysis, Python for kids - check out http://www.insightstack.co.za