In [1]:
```
from IPython.display import HTML
HTML("""
<br><br>
<a href=http://wwwgong.pythonanywhere.com/cuspea/default/list_talks target=new>
<font size=+3 color=blue>CUSPEA Talks</font>
</a>
<br><br>
<img src=images/jupyter-notebook-wen-gong.jpg><br>
""")
```

Out[1]:

CUSPEA Talks

(http://wwwgong.pythonanywhere.com/cuspea/default/list_talks)

In [ ]:

Fun with Jupyter (http://jupyter.org/)

Table of Contents

Motivation

- Current Choice

- A New Option

> The **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

Useful for many tasks

- Programming
- Blogging
- Learning
- Research
- Documenting work
- Collaborating
- Communicating
- Publishing results

or even

- Doing homework as a student

In [2]:
```
HTML("<img src=images/office-suite.jpg>")
```

Out[2]:

	Name	Description	Alternative
2	PowerPoint	Presentation app - part of MS Office suite	Jupyter Notebook
3	Word	Text processing app - part of MS Office suite	Jupyter Notebook
4	Excel	Spreadsheet app - calculation, graphing, table, VBA macro	Jupyter Notebook
5	Access	simple database	Jupyter Notebook
6	OneNote	free-form information gathering and multi-user collaboration	Jupyter Notebook
7	Communicator	Part of Skype for Business with basic features - instant messaging, VoIP, video conferencing	wechat, whatsapp, skype, zoom
8	Outlook	personal information manager for email, calendar, tasks, contacts, journal	gMail
9	Publisher	desktop publishing app	
10	InfoPath	(discontinued) app for designing, distributing, filling and submitting electronic forms containing structured data	

See this Demo Notebook (https://nbviewer.jupyter.org/github/waltherg/notebooks/blob/master/2013-12-03-Crank_Nicolson.ipynb)

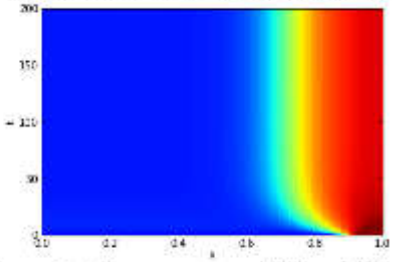
The Crank-Nicolson Method

The Crank-Nicolson method is a well-known finite difference method for the numerical in

We often resort to a Crank-Nicolson (CN) scheme when we integrate numerically reactic

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + f(u),$$

$$\left.\frac{\partial u}{\partial x}\right|_{x=0,L} = 0,$$



where u is our concentration variable, x is the space variable, D is the diffusion coeffici

Note that we use Neumann boundary conditions and specify that the solution u has z boundaries (no-flux boundary conditions).

Introduction

By solving a simple math problem here, I hope to demonstrate many basic features of Jupyter Notebook.

Problem Statement

A hot startup is selling a hot product.

In 2014, it had 30 customers; 3 years later, its customer base expands to 250.

Question: What is its annual growth rate?

Translate into math: Let x be annual growth rate,

then

$$30 \cdot (1+x)^3 = 250$$

Import packages (check out from Library)

In [3]:
```
# math function
import math

# create np array
import numpy as np

# pandas for data analysis
import pandas as pd

# plotting
import matplotlib.pyplot as plt
%matplotlib inline

# symbolic math
import sympy as sy

# html5
from IPython.display import HTML, SVG, YouTubeVideo

# widgets
from collections import OrderedDict
from IPython.display import display, clear_output
from ipywidgets import Dropdown

# csv file
import csv

# work with Sqlite database
import sqlite3
```

Estimate x range (in Elementary School)

If x = 1, then *l.h.s* = 240, therefore x > 1

If x = 1.2, then l.h.s =

In [4]:
```
30*(1+1.2)**3
```

Out[4]:
```
319.4400000000001
```

therefore we know x range = (1.0, 1.2)

Use IPython as a calculator (in Middle School)

In [5]:
```
# import math
```

In [6]:
```
math.exp(math.log(250/30)/3) - 1
```

Out[6]:
```
1.0274006651911334
```

In [7]:
```
10**(math.log10(250/30)/3) - 1
```

Out[7]:
```
1.0274006651911334
```

In [8]:
```
math.pow(10, math.log10(250/30)/3) -1
```

Out[8]:
```
1.0274006651911334
```

annual customer growth rate = 102%

Use Python programming to find solution (in High School)

use loop

In [9]:
```
nstep = 100
x_min, x_max = 1.0, 1.2
dd = (x_max-x_min)/float(nstep)
x_l = [(x_min + i*dd) for i in range(nstep)]
type(x_l)
```

Out[9]:
```
list
```

In [10]:
```
print(x_l)
```

```
[1.0, 1.002, 1.004, 1.006, 1.008, 1.01, 1.012, 1.014, 1.016, 1.018, 1.02, 1.02
2, 1.024, 1.026, 1.028, 1.03, 1.032, 1.034, 1.036, 1.038, 1.04, 1.042, 1.044,
 1.046, 1.048, 1.05, 1.052, 1.054, 1.056, 1.058, 1.06, 1.062, 1.064, 1.066, 1.0
68, 1.07, 1.072, 1.074, 1.076, 1.078, 1.08, 1.082, 1.084, 1.086, 1.088, 1.09,
 1.092, 1.094, 1.096, 1.0979999999999999, 1.1, 1.1019999999999999, 1.104, 1.105
9999999999999, 1.108, 1.1099999999999999, 1.1119999999999999, 1.113999999999999
9, 1.1159999999999999, 1.1179999999999999, 1.1199999999999999, 1.12199999999999
99, 1.1239999999999999, 1.126, 1.128, 1.13, 1.132, 1.134, 1.136, 1.138, 1.14,
 1.142, 1.144, 1.146, 1.148, 1.15, 1.152, 1.154, 1.156, 1.158, 1.16, 1.162, 1.1
64, 1.166, 1.168, 1.17, 1.172, 1.174, 1.176, 1.178, 1.18, 1.182, 1.184, 1.186,
 1.188, 1.19, 1.192, 1.194, 1.196, 1.198]
```

In [11]:
```
for t in x_l:
    err = abs(30*(1+t)**3 - 250)
    if err <= 0.5:
        print("t={x}: error={e:.4f}".format(x=t,e=err))
```

```
t=1.028: error=0.2218
```

create a numpy array

In [12]:
```
# import numpy as np
# import pandas as pd
```

In [13]:
```
print(x_l)
```

```
[1.0, 1.002, 1.004, 1.006, 1.008, 1.01, 1.012, 1.014, 1.016, 1.018, 1.02, 1.02
2, 1.024, 1.026, 1.028, 1.03, 1.032, 1.034, 1.036, 1.038, 1.04, 1.042, 1.044,
 1.046, 1.048, 1.05, 1.052, 1.054, 1.056, 1.058, 1.06, 1.062, 1.064, 1.066, 1.0
68, 1.07, 1.072, 1.074, 1.076, 1.078, 1.08, 1.082, 1.084, 1.086, 1.088, 1.09,
 1.092, 1.094, 1.096, 1.0979999999999999, 1.1, 1.1019999999999999, 1.104, 1.105
9999999999999, 1.108, 1.1099999999999999, 1.1119999999999999, 1.113999999999999
9, 1.1159999999999999, 1.1179999999999999, 1.1199999999999999, 1.12199999999999
99, 1.1239999999999999, 1.126, 1.128, 1.13, 1.132, 1.134, 1.136, 1.138, 1.14,
 1.142, 1.144, 1.146, 1.148, 1.15, 1.152, 1.154, 1.156, 1.158, 1.16, 1.162, 1.1
64, 1.166, 1.168, 1.17, 1.172, 1.174, 1.176, 1.178, 1.18, 1.182, 1.184, 1.186,
 1.188, 1.19, 1.192, 1.194, 1.196, 1.198]
```

using arange()

In [14]:
```
x = np.arange(1.0, 1.2, 0.005)
print(x)
```

```
[ 1.     1.005  1.01   1.015  1.02   1.025  1.03   1.035  1.04   1.045
  1.05   1.055  1.06   1.065  1.07   1.075  1.08   1.085  1.09   1.095  1.1
  1.105  1.11   1.115  1.12   1.125  1.13   1.135  1.14   1.145  1.15
  1.155  1.16   1.165  1.17   1.175  1.18   1.185  1.19   1.195]
```

check its type

In [15]:
```
type(x)
```

Out[15]:
```
numpy.ndarray
```

In [16]:
```
len(x)
```

Out[16]:
```
40
```

In [17]:
```
print(30*(1+x)**3 - 250)
```

```
[-10.          -8.19549625  -6.38197     -4.55939875  -2.72776     -0.88703125
   0.96281      2.82178625   4.68992      6.56723375   8.45375     10.34949125
  12.25448     14.16873875  16.09229     18.02515625  19.96736     21.91892375
  23.87987     25.85022125  27.83        29.81922875  31.81793     33.82612625
  35.84384     37.87109375  39.90791     41.95431125  44.01032     46.07595875
  48.15125     50.23621625  52.33088     54.43526375  56.54939     58.67328125
  60.80696     62.95044875  65.10377     67.26694625]
```

In [18]:
```
x_ge_0 = (30*(1+x)**3 - 250) >= 0
x_ge_0
```

Out[18]:
```
array([False, False, False, False, False, False,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True], dtype=bool)
```

In [19]:
```
x_lt_0 = (30*(1+x)**3 - 250) < 0
x_lt_0
```

Out[19]:
```
array([ True,  True,  True,  True,  True,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False], dtype=bool)
```

x_ge_0 and x_lt_0 are logical array

In [20]:
```
for t in x:
    err = abs(30*(1+t)**3 - 250)
    if err <= 1.0:
        print("t={x}: error={e:.4f}".format(x=t,e=err))
```

```
t=1.0249999999999995: error=0.8870
t=1.0299999999999994: error=0.9628
```

using linspace (https://docs.scipy.org/doc/numpy/reference/generated/numpy.linspace.html)

In [21]:
```
x1 = np.linspace(1.0, 1.2, 100)
x1
```

Out[21]:
```
array([ 1.        ,  1.0020202 ,  1.0040404 ,  1.00606061,  1.00808081,
        1.01010101,  1.01212121,  1.01414141,  1.01616162,  1.01818182,
        1.02020202,  1.02222222,  1.02424242,  1.02626263,  1.02828283,
        1.03030303,  1.03232323,  1.03434343,  1.03636364,  1.03838384,
        1.04040404,  1.04242424,  1.04444444,  1.04646465,  1.04848485,
        1.05050505,  1.05252525,  1.05454545,  1.05656566,  1.05858586,
        1.06060606,  1.06262626,  1.06464646,  1.06666667,  1.06868687,
        1.07070707,  1.07272727,  1.07474747,  1.07676768,  1.07878788,
        1.08080808,  1.08282828,  1.08484848,  1.08686869,  1.08888889,
        1.09090909,  1.09292929,  1.09494949,  1.0969697 ,  1.0989899 ,
        1.1010101 ,  1.1030303 ,  1.10505051,  1.10707071,  1.10909091,
        1.11111111,  1.11313131,  1.11515152,  1.11717172,  1.11919192,
        1.12121212,  1.12323232,  1.12525253,  1.12727273,  1.12929293,
        1.13131313,  1.13333333,  1.13535354,  1.13737374,  1.13939394,
        1.14141414,  1.14343434,  1.14545455,  1.14747475,  1.14949495,
        1.15151515,  1.15353535,  1.15555556,  1.15757576,  1.15959596,
        1.16161616,  1.16363636,  1.16565657,  1.16767677,  1.16969697,
        1.17171717,  1.17373737,  1.17575758,  1.17777778,  1.17979798,
        1.18181818,  1.18383838,  1.18585859,  1.18787879,  1.18989899,
        1.19191919,  1.19393939,  1.1959596 ,  1.1979798 ,  1.2       ])
```

In [22]:
```
for t in x1:
    err = math.fabs(30*(1+t)**3 - 250)
    if err <= 1.0:
        print("t={x}: error={e:.4f}".format(x=t,e=err))
```

```
t=1.0262626262626262: error=0.4208
t=1.0282828282828282: error=0.3265
```

Graph the solution using Matplotlib (in High School)

matplotlib (http://matplotlib.org/contents.html?v=20170307111739) is visualization pkg for python

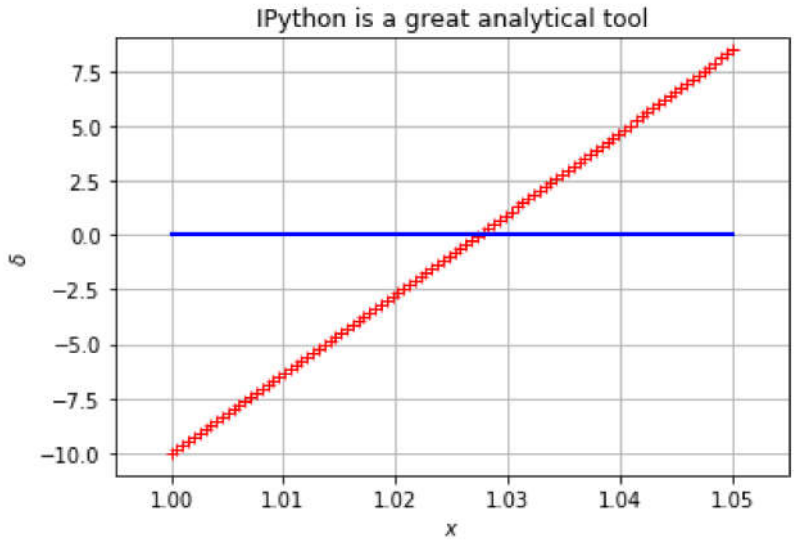
In [23]:
```
# import matplotlib.pyplot as plt
# %matplotlib inline
```

In [24]:
```
x2 = np.linspace(1.0, 1.05, 100)
f1 = 30*(1+x2)**3 - 250

f2 = np.zeros_like(x2)    # draw a horizontal line at y=0
```

x intersection of two lines f1/f2 gives the solution

In [25]:
```
plt.xlabel(r'$x$')
plt.ylabel(r'$\delta$')
plt.grid(True)
plt.title('IPython is a great analytical tool')
plt.axis([0.995,1.055, -11, 9])
#plt.axis([1.02, 1.04, -11, 9])
plt.plot(x2, f1, 'r+')
plt.plot(x2, f2, 'b-', lw=2)
plt.show()
```



Solve equation precisely using SymPy (in High School)

In [26]:
```
# from sympy import *
# import sympy as sy
```

In [27]:
```
sy.var('x')
```

Out[27]:
```
x
```

In [28]:
```
sy.solve(30*(1+x)**3 - 250, x)
```

Out[28]:
```
[-1 + 15**(2/3)/3,
 -1 + 15**(2/3)*(-1/2 - sqrt(3)*I/2)/3,
 -1 + 15**(2/3)*(-1/2 + sqrt(3)*I/2)/3]
```

Ignore other 2 solutions because they are complex numbers

In [29]:
```
grow_rate = -1 + 15**(2/3)/3
grow_rate
```

Out[29]:
```
1.027400665191133
```

Pandas for Big Data Analytics (in College)

pandas (http://pandas.pydata.org/pandas-docs/stable/?v=20170307111739) stands for powerful Python data analysis toolkit

In [30]:
```
# import pandas as pd
```

In [31]:
```
year = [2014, 2015, 2016, 2017]
```

In [32]:
```
customer_count = [30*(1+grow_rate)**i  for i in range(4)]
print(customer_count)
```

```
[30.0, 60.822019955733985, 123.31060371652346, 249.9999999999998]
```

In [33]:
```
df = pd.DataFrame(list(zip(year,customer_count)), columns=['Year','Customers'])
df
```

Out[33]:

	Year	Customers
0	2014	30.000000
1	2015	60.822020
2	2016	123.310604
3	2017	250.000000

In [34]:
```
df.head(2)
```

Out[34]:

	Year	Customers
0	2014	30.00000
1	2015	60.82202

In [35]:
```
df.tail(2)
```

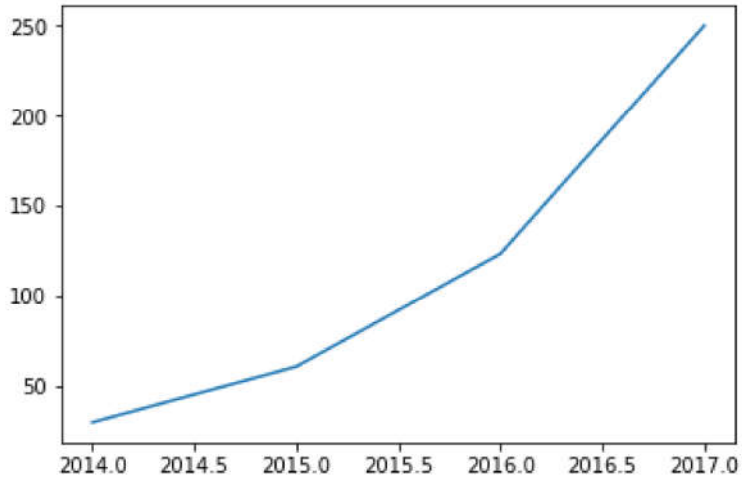
Out[35]:

	Year	Customers
2	2016	123.310604
3	2017	250.000000

Line chart

In [37]:
```
plt.plot(df['Year'], df['Customers'])
```

Out[37]:
```
[<matplotlib.lines.Line2D at 0xb63ad30>]
```



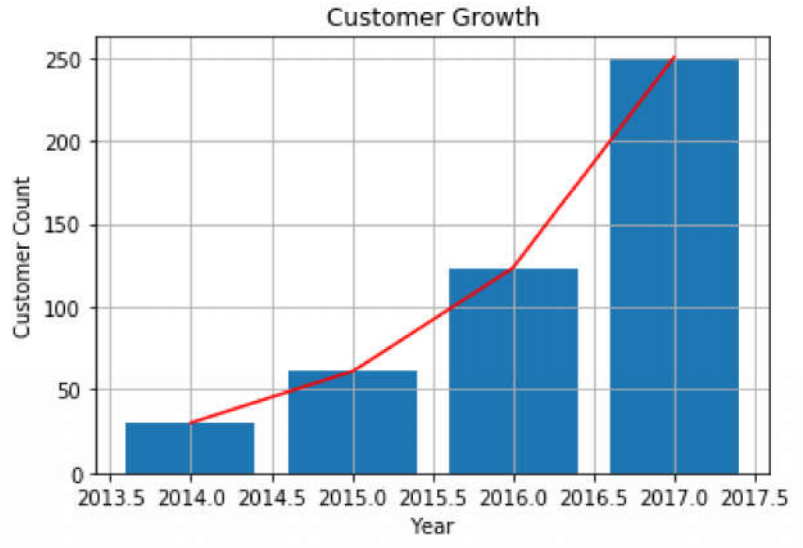
Bar chart

In [39]:
```
plt.xlabel('Year')
plt.ylabel('Customer Count')
plt.grid(True)
plt.title('Customer Growth')

plt.bar(df['Year'], df['Customers'])

plt.plot(df['Year'], df['Customers'], 'r-')
```

Out[39]: [<matplotlib.lines.Line2D at 0xb68c7f0>]



Multimedia with HTML5 -Text, Image, Audio, Video (Having graduated from all the schools)

In [42]:
```
# from IPython.display import HTML, SVG, YouTubeVideo
```

create an HTML table dynamically with Python, and we display it in the (HTML-based) notebook.

Let me create a multiplication table

In [44]:
```
HTML('''
<table style="border: 2px solid black;">
''' +
''.join(['<tr>' +
         ''.join(['<td>{row}x{col}={prod}</td>'.format(
            row=row, col=col, prod=row*col
            ) for col in range(10)]) +
         '</tr>' for row in range(10)]) +
'''
</table>
''')
```

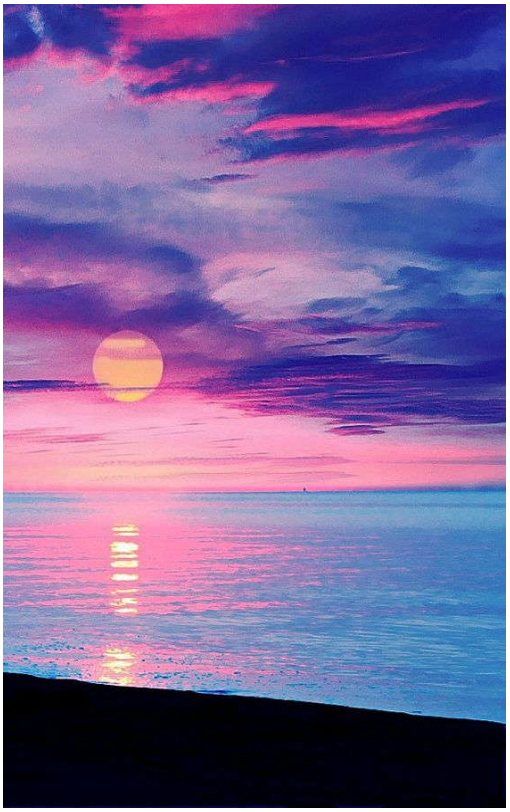
Out[44]:

0x0=0	0x1=0	0x2=0	0x3=0	0x4=0	0x5=0	0x6=0	0x7=0	0x8=0	0x9=0
1x0=0	1x1=1	1x2=2	1x3=3	1x4=4	1x5=5	1x6=6	1x7=7	1x8=8	1x9=9
2x0=0	2x1=2	2x2=4	2x3=6	2x4=8	2x5=10	2x6=12	2x7=14	2x8=16	2x9=18
3x0=0	3x1=3	3x2=6	3x3=9	3x4=12	3x5=15	3x6=18	3x7=21	3x8=24	3x9=27
4x0=0	4x1=4	4x2=8	4x3=12	4x4=16	4x5=20	4x6=24	4x7=28	4x8=32	4x9=36
5x0=0	5x1=5	5x2=10	5x3=15	5x4=20	5x5=25	5x6=30	5x7=35	5x8=40	5x9=45
6x0=0	6x1=6	6x2=12	6x3=18	6x4=24	6x5=30	6x6=36	6x7=42	6x8=48	6x9=54
7x0=0	7x1=7	7x2=14	7x3=21	7x4=28	7x5=35	7x6=42	7x7=49	7x8=56	7x9=63
8x0=0	8x1=8	8x2=16	8x3=24	8x4=32	8x5=40	8x6=48	8x7=56	8x8=64	8x9=72
9x0=0	9x1=9	9x2=18	9x3=27	9x4=36	9x5=45	9x6=54	9x7=63	9x8=72	9x9=81

display image

In [45]:
```
HTML("""
<img src="https://s-media-cache-ak0.pinimg.com/564x/59/b6/cc/59b6cc26b9502fa7d8f4
""")
```

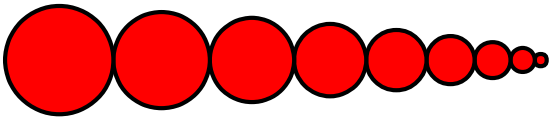
Out[45]:



create a SVG graphics dynamically.

In [46]:
```
SVG('''<svg width="600" height="80">''' +
''.join(['''<circle cx="{x}" cy="{y}" r="{r}"
        fill="red" stroke-width="2" stroke="black">
        </circle>'''.format(
            x=(30+3*i)*(10-i), y=30, r=3.*float(i)
        ) for i in range(10)]) +
'''</svg>''')
```

Out[46]:



embed an audio clip

In [47]:
```
HTML("""

<table>
<tr>
<td>
<a href=https://talkpython.fm>TalkPython </a>
<h2> Guido van Rossum</h2> (Python Creator & Tsar)
<h3> Python past, present, and future </h3>
<br/>
<audio controls>
  <source src="https://downloads.talkpython.fm/static/episode_cache/100-guido-van
Your browser does not support the audio element.
</audio>
</td>
<td>
<img src="https://pbs.twimg.com/profile_images/424495004/GuidoAvatar_400x400.jpg"
</td>
</tr>
</table>
""")
```

Out[47]:

TalkPython (https://talkpython.fm)

Guido van Rossum
(Python Creator & Tsar)

Python past, present, and future

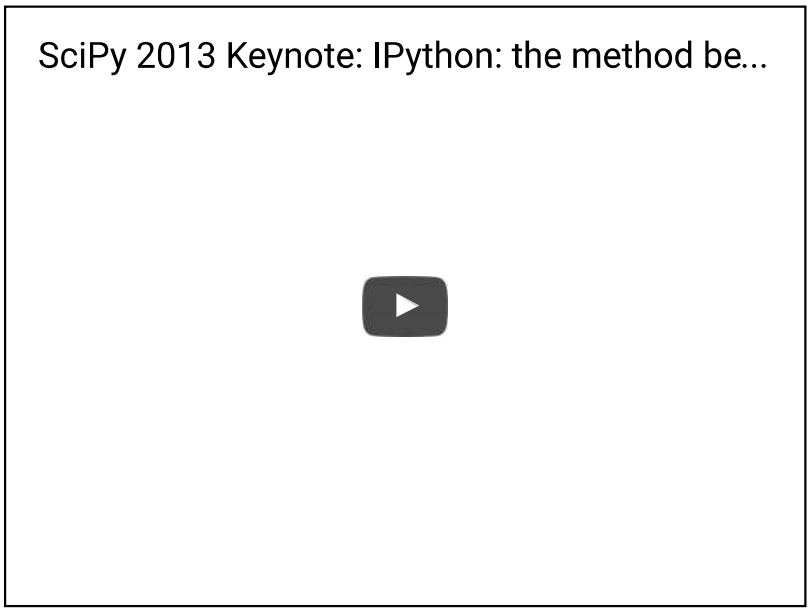
00:00 / 61:48

display a Youtube video by giving its identifier to YoutubeVideo.

SciPy 2013 Keynote: IPython

In [48]:
```
YouTubeVideo('j9YpkSX7NNM')
```

Out[48]:



Interactive widgets

we illustrate the latest interactive features in IPython 2.0+.

This version brings graphical widgets in the notebook that can interact with Python objects.

We will create a drop-down menu allowing us to display one among several videos.

In [49]:
```
# How to comment out multiple lines of code in python
"""
from collections import OrderedDict
from IPython.display import display, clear_output
from ipywidgets import Dropdown
"""
```

Out[49]:
```
' \nfrom collections import OrderedDict\nfrom IPython.display import display, c
lear_output\nfrom ipywidgets import Dropdown\n'
```

In [50]:
```
# We create a Dropdown widget, with a dictionary containing
# the keys (video name) and the values (Youtube identifier) of every menu item.

dw = Dropdown(options=OrderedDict([
                        ('SciPy 2012', 'iwVvqwLDsJo'),
                        ('PyCon 2012', '2G5YTlheCbw'),
                        ('SciPy 2013', 'j9YpkSX7NNM'),
                        ('Guido Van Rossum', 'EBRMq2Ioxsc'),
                        ('Mendelssohn Violin', 'o1dBg__wsuo')
        ]))

# We create a callback function that displays the requested Youtube video.
def on_value_change(name, val):
    clear_output()
    display(YouTubeVideo(val))
# Every time the user selects an item, the function
# `on_value_change` is called, and the `val` argument
# contains the value of the selected item.
dw.on_trait_change(on_value_change, 'value')
# We choose a default value.
dw.value = dw.options['Mendelssohn Violin']

# Finally, we display the widget.
display(dw)
```


```
Widget Javascript not detected.  It may not be installed properly. Did you enab
le the widgetsnbextension? If not, then run "jupyter nbextension enable --py --
sys-prefix widgetsnbextension"
```

Working with SQLite Databases

read blog (https://www.dataquest.io/blog/python-pandas-databases/) at DataQuest (https://www.dataquest.io)

In [51]:
```
# open connection to db
conn = sqlite3.connect("dataset/open_src.sqlite")

# create a cursor
cur = conn.cursor()

# select query
results = cur.execute("select * from os_history limit 100;").fetchall()
#print(results)

# count # of rows
results = cur.execute("select count(*) from os_history;").fetchall()
#print(results)

# store data from csv file into db
with open('dataset/open_src_move_v2_1.csv') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        #print(row['Year'])
        insert_str="""
            insert into os_history(year,subject,subjecturl,person,picture,history
            values ({year}, "{subject}", "{subject_url}", "{person}", "{picture}"
        """
        #print(insert_str.format(year=row['Year'],subject=row['Subject'],subject_
        cur.execute(insert_str.format(year=row['Year'],subject=row['Subject'],sub
    conn.commit()

# create a dataframe
df = pd.read_sql_query("select * from os_history limit 5;",conn)

# inspect data
df
```

Out[51]:

	year	subject	subjecturl	person	picture
0	1983	GNU Project : gcc, Emacs, gdb	https://en.wikipedia.org/wiki/GNU_Project	Richard Stallman	https://upload.wik
1	1984	X Window Systerm	https://en.wikipedia.org/wiki/X_Window_System	X.Org	https://upload.wik
2	1983	GNU Project : gcc, Emacs, gdb	https://en.wikipedia.org/wiki/GNU_Project	Richard Stallman	https://upload.wik

	year	subject	subjecturl	person	picture
3	1984	X Window Systerm	https://en.wikipedia.org/wiki/X_Window_System	X.Org	https://upload.wik
4	1985	GNU Manifesto: GNU's Not Unix	https://en.wikipedia.org/wiki/GNU_Manifesto	Richard Stallman	https://upload.wik

References

Websites

- DataCamp - Jupyter Notebook Tutorial (https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook#gs.ClmI4Jc)
- http://docs.python.org (http://docs.python.org)

It goes without saying that Pythonâ€™s own online documentation is an excellent resource if you need to delve into the finer details of the language and modules. Just make sure youâ€™re looking at the documentation for Python 3 and not earlier versions.

- http://www.python.org/dev/peps (http://www.python.org/dev/peps)

Python Enhancement Proposals (PEPs) are invaluable if you want to understand the motivation for adding new features to the Python language as well as subtle implementation details. This is especially true for some of the more advanced language features. In writing this book, the PEPs were often more useful than the official documentation.

- http://pyvideo.org (http://pyvideo.org)

This is a large collection of video presentations and tutorials from past PyCon conferences, user group meetings, and more. It can be an invaluable resource for learning about modern Python development. Many of the videos feature Python core developers talking about the new features being added in Python 3.

- http://code.activestate.com/recipes/langs/python (http://code.activestate.com/recipes/langs/python)

The ActiveState Python recipes site has long been a resource for finding the solution to thousands of specific programming problems. As of this writing, it contains approximately 300 recipes specific to Python 3. Youâ€™ll find that many of its recipes either expand upon topics covered in this book or focus on more narrowly defined tasks. As such, itâ€™s a good companion.

- http://stackoverflow.com/questions/tagged/python (http://stackoverflow.com/questions/tagged/python)

Stack Overflow currently has more than 175,000 questions tagged as Python-related (and almost 5000 questions specific to Python 3). Although the quality of the questions and answers varies, there is a lot of good material to be found.

Books

- Learning IPython for Interactive Computing and Data Visualization - Second Edition (By Cyrille Rossant) (https://github.com/ipython-books/minibook-2nd-code)
- IPython Interactive Computing and Visualization Cookbook (By Cyrille Rossant) (https://github.com/ipython-books/cookbook-code)
- Python Cookbook, 3rd Edition by David Beazley; Brian K. Jones (https://github.com/dabeaz/python-cookbook)
- Python for Data Analysis by Wes McKinney (https://github.com/wesm/pydata-book)

Other Resources

- Idea
  - Google Search (http://www.google.com)
- Text
  - Wikipedia (https://www.wikipedia.org/)
- Image
  - Google Images (https://www.google.com/imghp)
- Video
  - YouTube (https://www.youtube.com/)

Contributors

- wen.gong@oracle.com (first created on 2017-03-09)

Appendix

How to install Jupyter Notebook

I use Windows and follow this link to install Ananconda Python distribution (https://www.tensorflow.org/install/)

- Follow the instructions on the Anaconda download site (https://www.continuum.io/downloads) to download and install Anaconda.
- open a DOS command box, launch jupyter by typing

```
jupyter notebook
```

- wait till Jupyter homepage to open in a browser
- start to create your own Notebook

How to share a notebook

share your learning or work via nbviewer at http://nbviewer.jupyter.org/ (http://nbviewer.jupyter.org/)

In [52]:
```
HTML("<img src=images/to-be-continued-1.jpg>")
```

Out[52]:

In [ ]:
```

```

In [ ]:
```

```