Python for Science and Engg: Interactive Plotting

FOSSEE

Department of Aerospace Engineering
IIT Bombay

7 November, 2009
Day 1, Session 1

Workshop Schedule: Day 1

Session 1	Sat 10:00–11:00
Session 2	Sat 11:10–12:10
Session 3	Sat 12:15–13:15
Quiz 1	Sat 14:15–14:35
Exercises	Sat 14:35–15:15
Session 4	Sat 15:25–16:25
Session 5	Sat 16:30–17:30
Quiz 2	Sat 17:30–18:00

Workshop Schedule: Day 2

Session 1	Sun 09:00–10:00
Session 2	Sun 10:05–11:05
Session 3	Sun 11:20–12:20
Session 4	Sun 12:25–13:25
Quiz 1	Sun 14:25–14:40
Exercises	Sun 14:40–15:20
Session 5	Sun 15:30–16:30
Quiz 2	Sun 16:30–17:00

About the Workshop

Intended Audience

- Engg., Mathematics and Science teachers.
- Interested students from similar streams.

Goal: Successful participants will be able to

- Use Python as plotting, computational tool
- Understand how to use Python as a scripting and problem solving language.
- Train students for the same

Outline

1. Getting started
2. Plotting
   - Drawing plots
   - Decoration
   - More decoration
3. Multiple plots

Checklist

1. IPython
2. Editor: We recommend scite.
3. Data files:
    - `sslc1.txt`
    - `pendulum.txt`
    - `points.txt`
    - `pos.txt`
4. Images:
    - `lena.png`
    - `smoothing.gif`

Starting up . . .

```
$ ipython -pylab
```

```
In []: print "Hello, World!"
Hello, World!
```

Exiting

```
In []: ^D(Ctrl-D)
Do you really want to exit([y]/n)? y
```

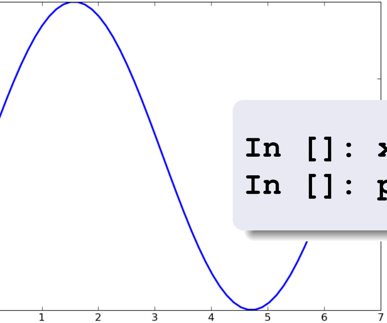
Outline

1. Getting started
2. Plotting
   - Drawing plots
   - Decoration
   - More decoration
3. Multiple plots

Outline

1. Getting started
2. Plotting
   - Drawing plots
   - Decoration
   - More decoration
3. Multiple plots

First Plot



```
In []: x = linspace(0, 2*pi, 50)
In []: plot(x, sin(x))
```

Walkthrough

```
x = linspace(start, stop, num)
```

returns **`num`** evenly spaced points, in the interval [**`start`**, **`stop`**].

```
x[0] = start
x[num - 1] = end
```

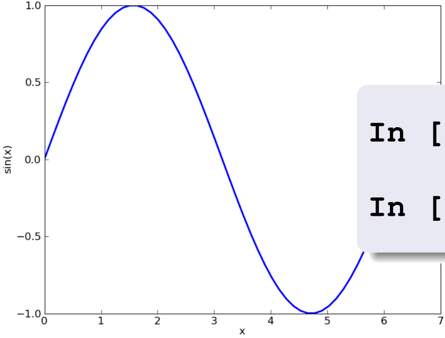
```
plot(x, y)
```

plots **`x`** and **`y`** using default line style and color

Outline

1. Getting started
2. Plotting
   - Drawing plots
   - Decoration
   - More decoration
3. Multiple plots

Adding Labels



```
In []: xlabel('x')

In []: ylabel('sin(x)')
```

Another example

```
In []: clf()
```

Clears the plot area.

```
In []: y = linspace(0, 2*pi, 50)
In []: plot(y, sin(2*y))
In []: xlabel('y')
In []: ylabel('sin(2y)')
```

Outline

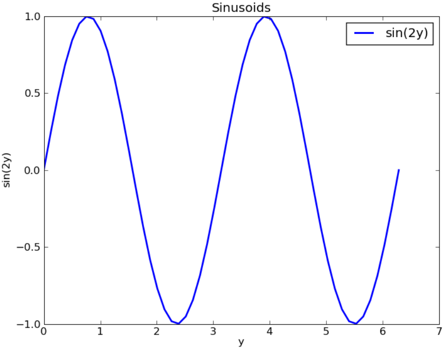
1. Getting started

2. Plotting
   - Drawing plots
   - Decoration
   - More decoration

3. Multiple plots

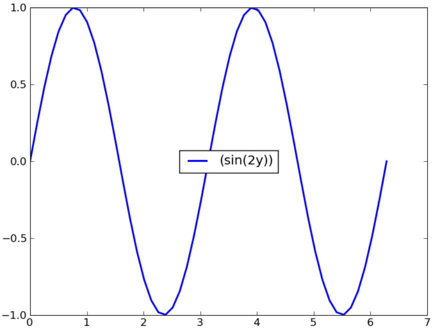
Title and Legends

```
In []: title('Sinusoids')
In []: legend(['sin(2y)'])
```



Legend Placement

```
In []: legend(['sin(2y)'], loc = 'center')
```



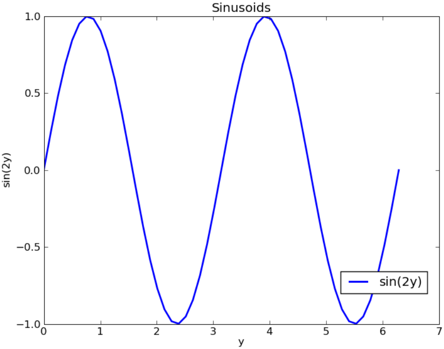
'upper right'
'upper left'
'lower left'
'lower right'
'center left'
'center right'
'lower center'
'upper center'

'best', 'right', 'center'

For arbitrary location

```
In []: legend(['sin(2y)'], loc=(.8,.1))
```

Specify south-east corner position



Saving & Closing

```
In []: savefig('sin.png')

In []: close()
```

Outline

1. Getting started
2. Plotting
   - Drawing plots
   - Decoration
   - More decoration
3. Multiple plots

Overlaid Plots

```
In []: clf()
In []: plot(y, sin(y))
In []: plot(y, cos(y))
In []: xlabel(’y’)
In []: ylabel(’f(y)’)
In []: legend([’sin(y)’, ’cos(y)’])
```

By default plots would be overlaid!

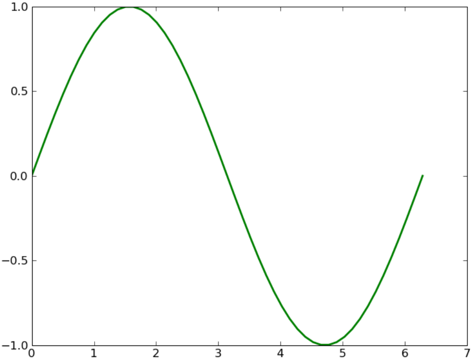
Plotting separate figures

```
In []: clf()
In []: figure(1)
In []: plot(y, sin(y))
In []: figure(2)
In []: plot(y, cos(y))
In []: figure(1)
In []: title('sin(y)')
In []: close()
In []: close()
```

Showing it better

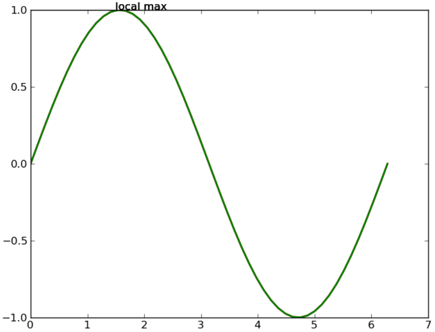
```
In []: plot(y, sin(y), 'g')

In []: plot(y, cos(y), 'r', linewidth=2)
```



Annotating

```
In []: annotate(’local max’, xy=(1.5, 1))
```



Axes lengths

Get the axes limits

```
In []: xmin, xmax = xlim()
In []: ymin, ymax = ylim()

In []: xmax = 2*pi
```

Set the axes limits

```
In []: xlim(xmin, xmax)
In []: ylim(ymin-0.2, ymax+0.2)
```

Review Problem

1. Plot x, -x, sin(x), xsin(x) in range -5π to 5π
2. Add a legend
3. Annotate the origin
4. Set axes limits to the range of x

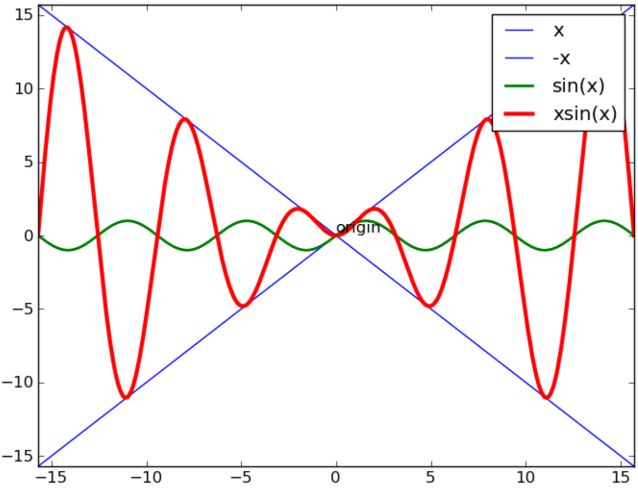
```
In []: x=linspace(-5*pi, 5*pi, 500)
In []: plot(x, x, 'b')
In []: plot(x, -x, 'b')
```

$\vdots$

Review Problem . . .

```
In []: plot(x, sin(x), 'g', linewidth=2)
In []: plot(x, x*sin(x), 'r',
             linewidth=3)

In []: legend(['x', '-x', 'sin(x)',
                'xsin(x)'])
In []: annotate('origin', xy = (0, 0))
In []: xlim(-5*pi, 5*pi)
In []: ylim(-5*pi, 5*pi)
```



Is this what you have?

Saving Commands

Save commands of review problem into file

- Use `%hist` command of IPython
- Identify the required line numbers
- Then, use `%save` command of IPython

```
In []: %hist
In []: %save four_plot.py 16 18-27
```

Careful about errors!

`%hist` will contain the errors as well,
so be careful while selecting line numbers.

Python Scripts. . .

This is called a Python Script.

- run the script in IPython using

```
%run -i four_plot.py
```

What did we learn?

- Creating simple plots.
- Adding labels and legends.
- Annotating plots.
- Changing the looks: size, linewidth
- `%hist`
- Saving commands to a script
- Running a script using `%run -i`