Introductory Scientific Computing with Python

Introduction, IPython and Plotting

FOSSEE

Department of Aerospace Engineering
IIT Bombay

SciPy India, 2015
December, 2015

Acknowledgement

FOSSEE group (`fossee.in`)
based at
IIT Bombay
and funded by
The National Mission on Education
through ICT,
Ministry of HRD, India

Outline

Checklist

1. IPython
2. Editor
3. Data files:
   - `pendulum.txt`
   - `sslc.txt`
4. Images
   - `bird.png`

About the Tutorial

Intended Audience

- Engg., Mathematics and Science researchers with a reasonable programming background.

Goal: Successful participants will be able to

- Start using Python as plotting, computational tool.
- Use the basic libraries and tools for scientific computing with Python.

Outline

Starting up . . .

Terminal

```
$ ipython -pylab
# OR
$ jupyter console -pylab
```

Running IPython

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

Exiting on the terminal

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

Or hit **Return** or **Ctrl-D** again

Outline

Breaking out of Loops

Breaking out of loops

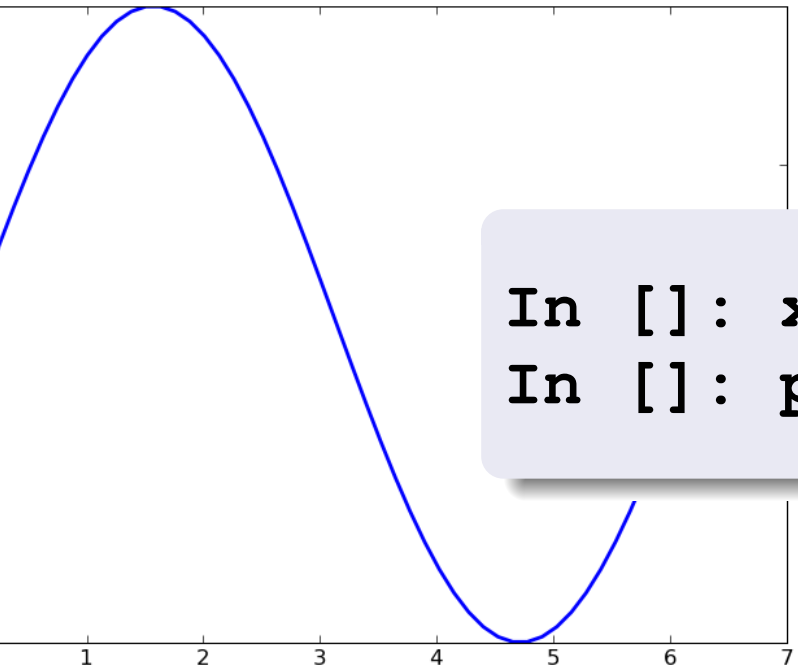
```
In []: while True:
   ...:     print("Hello, World!")
   ...:
Hello, World!
Hello, World!^C(Ctrl-C)
---------------------------------------------------------------------------
KeyboardInterrupt
```

10 m

Outline

Outline

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`**].

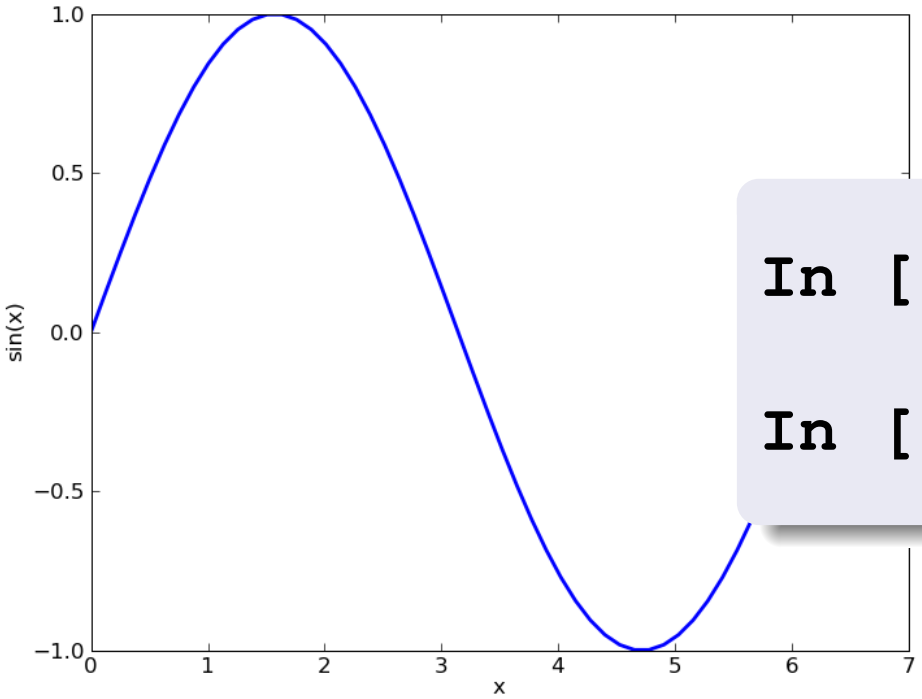
`x[0] = start`
`x[num - 1] = end`

`plot(x, y)`

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

Outline

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)')
```

IPython tips . . .

- Use **`TAB`** to complete command

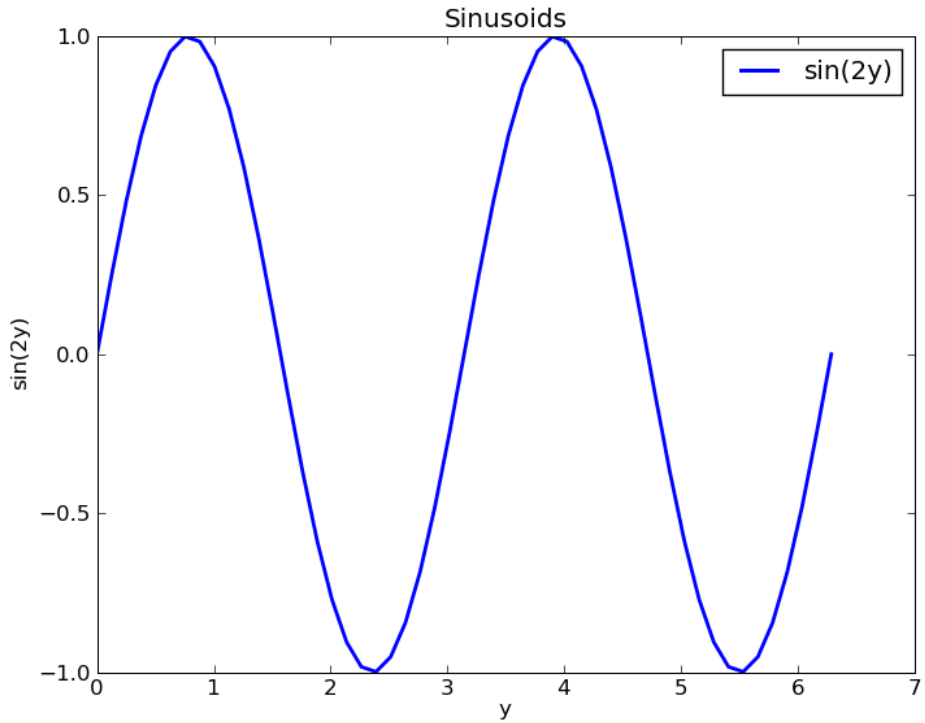
History

- Accesses history (also from past sessions)
- Up and down arrows (**`Ctrl-p`**/**`Ctrl-n`**)
- Search: **`Ctrl-r`** and start typing
- **`Ctrl-a`**: go to start of line
- **`Ctrl-e`**: go to end of line
- **`Ctrl-k`**: kill to end of line

Outline

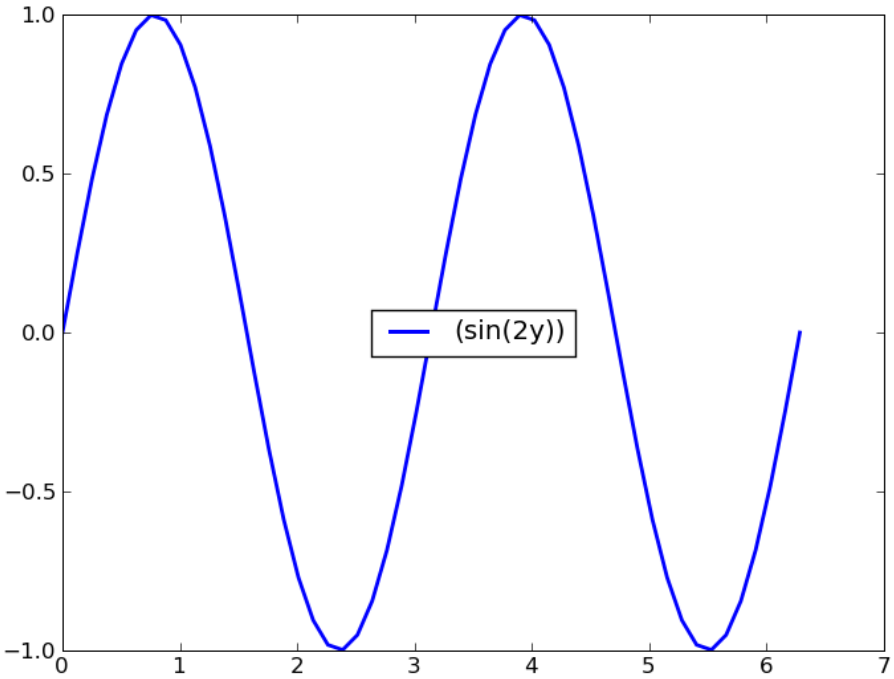
Title and Legends

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



Legend Placement

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

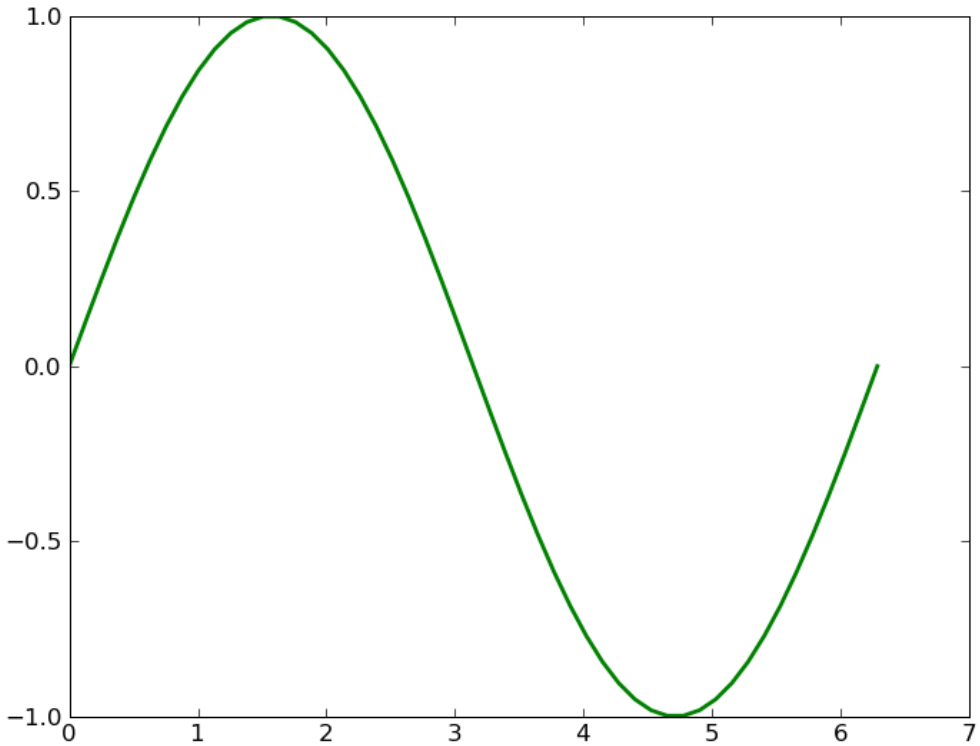



```
'best'
'right'
'center'
```

Showing it better

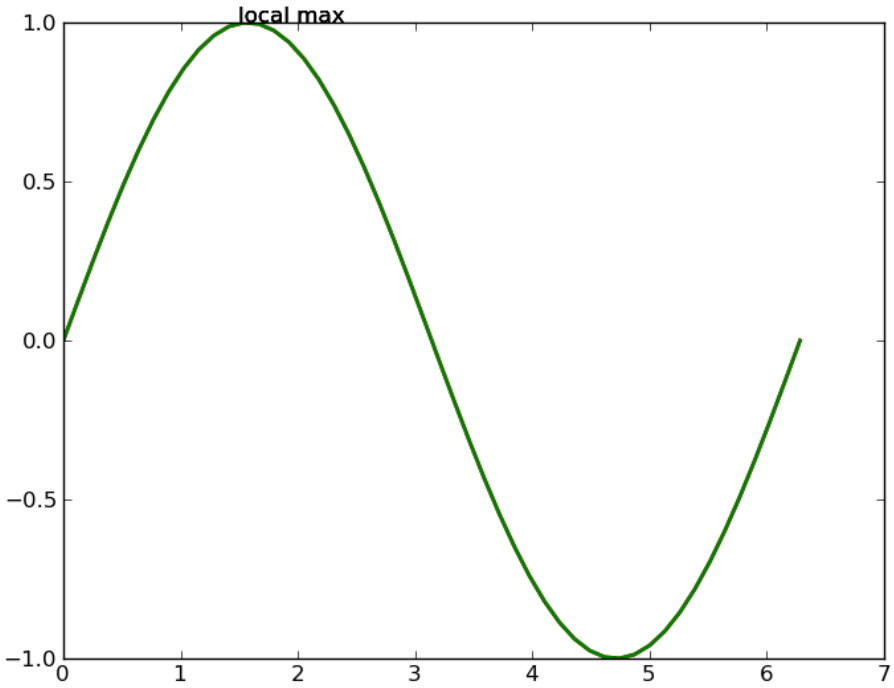
```
In []: plot(y, cos(y), 'r')

In []: clf()
In []: plot(y, sin(y), 'g', linewidth=2)
```



Annotating

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



Saving & Closing

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

In []: close()
```

Supported formats to store images:

- png
- eps - Easy to embed in LaTeX files
- pdf
- raw
- rgba
- svg

Outline

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 []: savefig('cosine.png')
In []: figure(1)
In []: title('sin(y)')
In []: savefig('sine.png')
In []: close()
In []: close()
```

Axes lengths

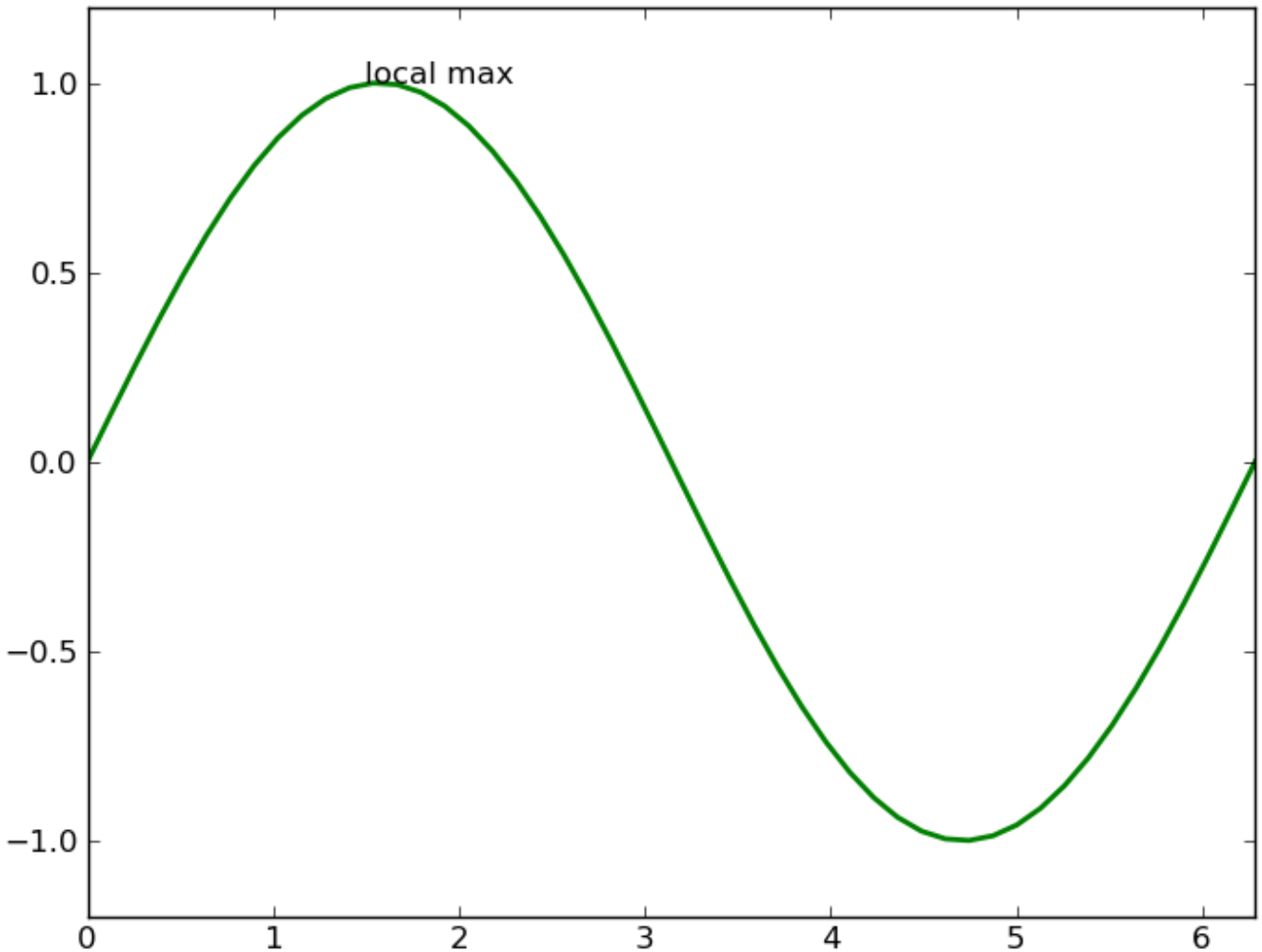
Getting axes lengths

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

Set the axes limits

```
In []: xlim(xmin, 2*pi )
In []: ylim(ymin-0.2, ymax+0.2)
```

Axes lengths



IPython tips . . .

- Try:

  `In []: plot?`

  to get more information on **`plot`**

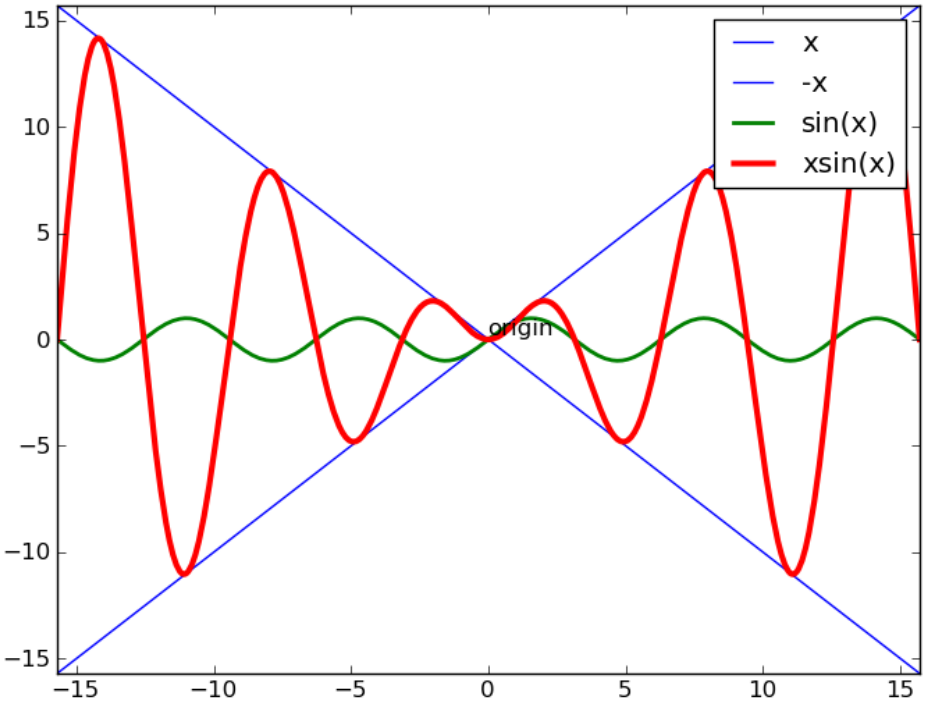
- Try:

  `In []: plot??`

  to see the source code for **`plot`**
- Note: exit pager with “q” or “ESC”

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



Review Problem . . .

Plotting . . .

```
In []: x = linspace(-5*pi, 5*pi, 500)
In []: plot(x, x, 'b')
In []: plot(x, -x, 'b')
In []: plot(x, sin(x), 'g', linewidth=2)
In []: plot(x, x*sin(x), 'r',
            linewidth=3)
```

⋮

Review Problem . . .

Legend & Annotation. . .

```
In []: legend(['x', '-x', 'sin(x)',
                    'xsin(x)'])
In []: annotate('origin', xy = (0, 0))
```

Setting Axes limits. . .

```
In []: xlim(-5*pi, 5*pi)
In []: ylim(-5*pi, 5*pi)
```

Outline

Command History

Use the **`%hist`** magic command of IPython

```
In []: %hist
```

This displays all the commands typed in so far aka Command History.

Careful about errors!

`%hist` will contain the errors as well.

Magic Commands?

Magic commands are commands provided by IPython to make our life easier.

Command History

Use the `%hist` magic command of IPython

```
In []: %hist
```

This displays all the commands typed in so far aka Command History.

Careful about errors!

`%hist` will contain the errors as well.

Magic Commands?

Magic commands are commands provided by IPython to make our life easier.

Saving commands into script

Use the **%save** magic command of IPython

```
%save script_name line_numbers
```

Line numbers can be specified individually separated by spaces or as a range separated by a dash.

```
%save four_plot.py 16 18-27
```

This saves from the history the commands entered on line numbers 16, 18, 19, 20, . . . 27

Python Scripts. . .

Now, four_plot.py is called a Python Script.

- run the script in IPython using
  `%run four_plot.py`

```
NameError: name 'linspace' is not defined
```

To avoid this, run using `%run -i four_plot.py`

You may need to do `show()` to see a plot window

Python Scripts. . .

Now, four_plot.py is called a Python Script.

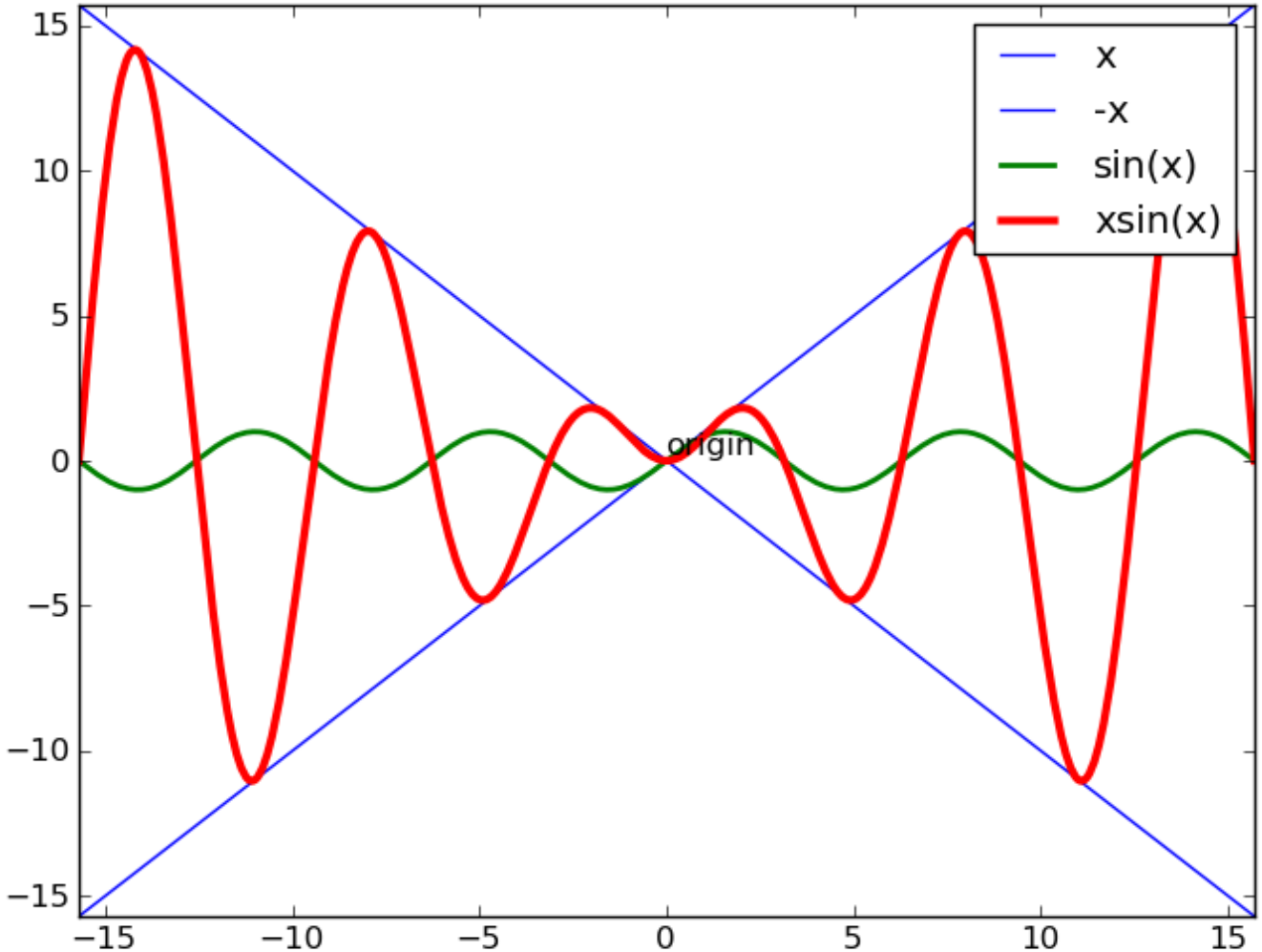
- run the script in IPython using
  `%run four_plot.py`

```
NameError: name 'linspace' is not defined
```

To avoid this, run using `%run -i four_plot.py`

You may need to do **`show()`** to see a plot window

Result graph



What did we learn?

- Starting up IPython
- Creating simple plots
- Adding labels and legends
- Annotating plots
- Changing the looks: size, linewidth
- Accessing history, documentation
- `%hist` - History of commands
- `%save` - Saving commands
- Running a script using `%run -i`

40 m