Introductory Scientific Computing with Python

Introduction, IPython and Plotting

FOSSEE

Department of Aerospace Engineering
IIT Bombay

Mumbai, India

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. Editor - we recommend Canopy
2. IPython
3. Data files:
   - `pendulum.txt`
   - `data.csv`
4. Images
   - `lena.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 . . .

Start a terminal

- Canopy command prompt (Tools menu)

On Terminal

```
$ ipython --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
```

IPython?

- An enhanced Python interpreter

Outline

Breaking out of Loops

Breaking out of loops

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

Exercise

- Exit the IPython interpreter
- Close the terminal
- Restart the terminal (Canopy tools menu)
- Restart IPython using:

```
$ ipython --pylab
```

10 m

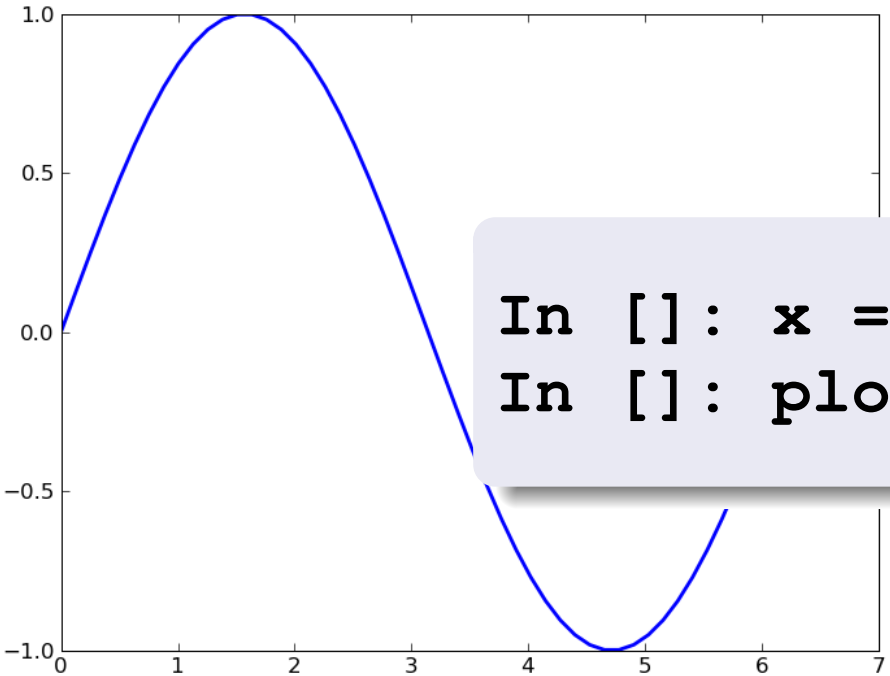
Outline

Important instructions

- For the first session, please do not experiment
- Follow along and type everything!
- Case matters
- Every character you type matters!

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`**].

```
In [ ]: x[0]
Out[ ]: 0.0

In [ ]: x[49]
Out[ ]: 6.283185307179586
```

Walkthrough . . .

`plot(x, y)`

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

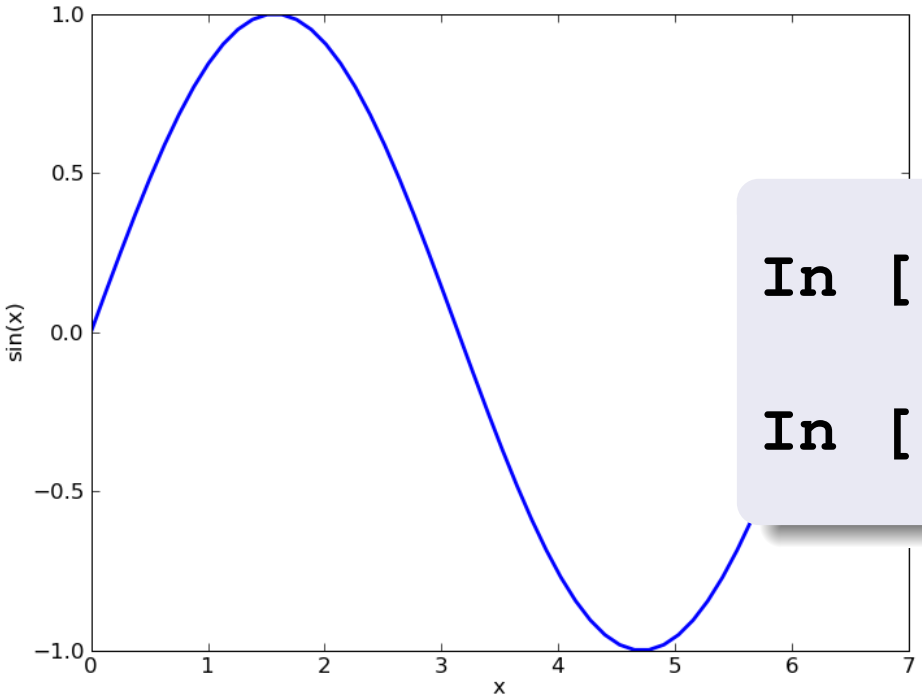
15 m

Important instructions

- Please do not close the plot windows or IPython
- For the first session, please do not experiment
- Follow along and type everything!
- Case matters
- Every character you type matters!

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
- Try:

```
In []: pl<TAB>
```

History

- Up arrow and down arrow
- Left / right to move and edit
- Type some text and press up arrow:

```
In []: pl<Up Arrow>
```

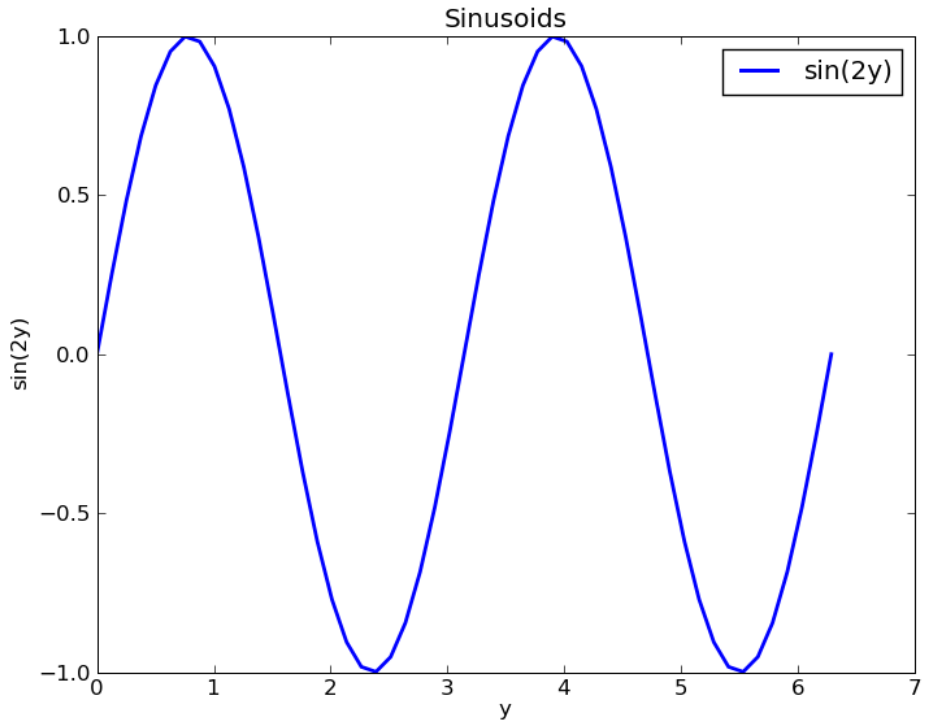
Advanced IPython tips . . .

- 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

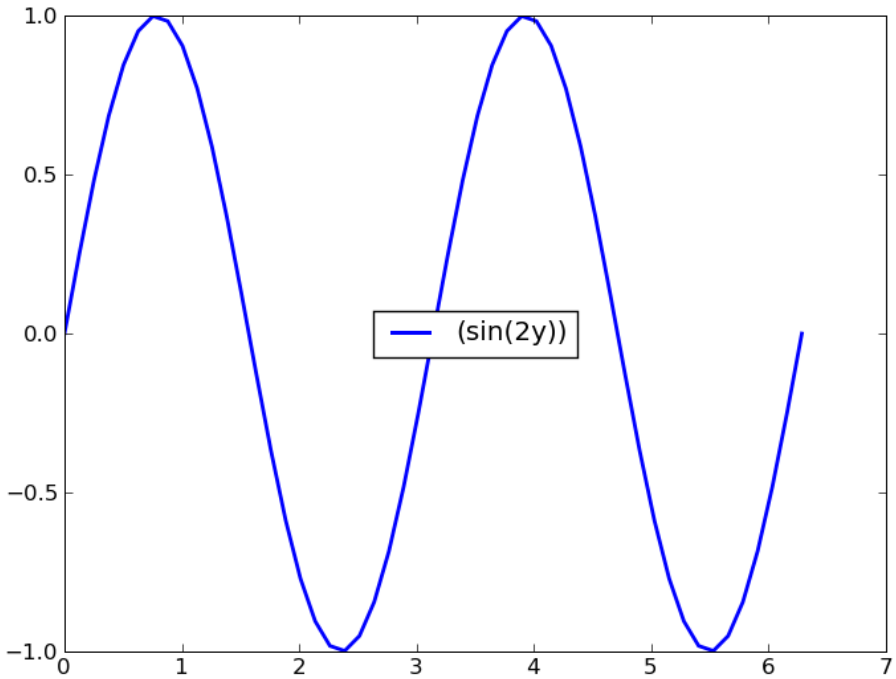
Title and Legends

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



Legend Placement

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




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

30 m

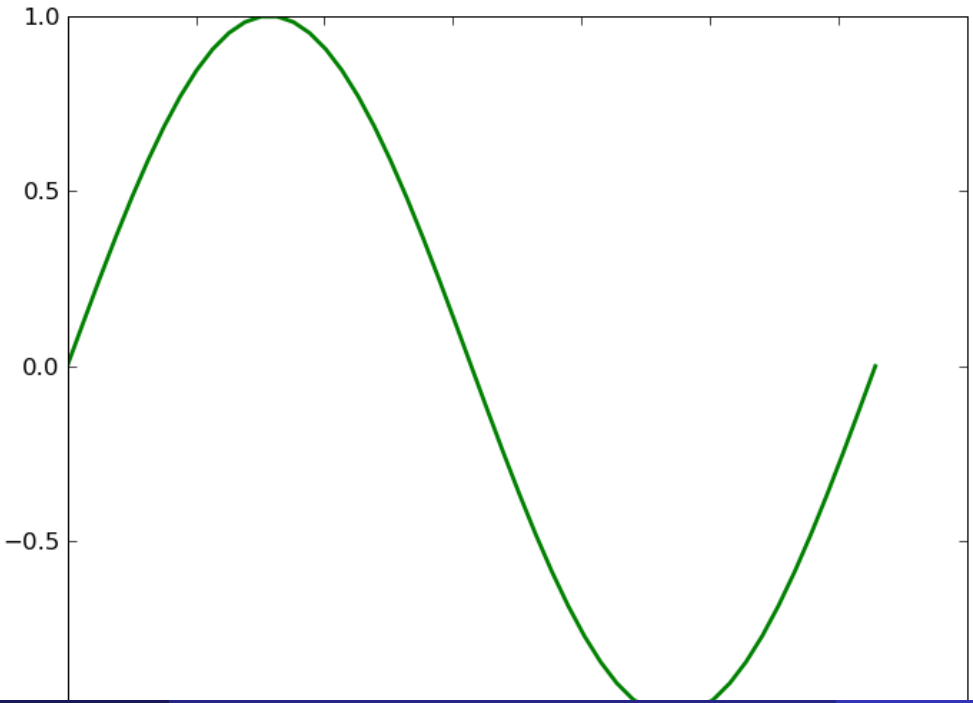
Important instructions

- Please do not close the plot windows or IPython
- For the first session, please do not experiment
- Follow along and type everything!
- Case matters
- Every character you type matters!

Showing it better

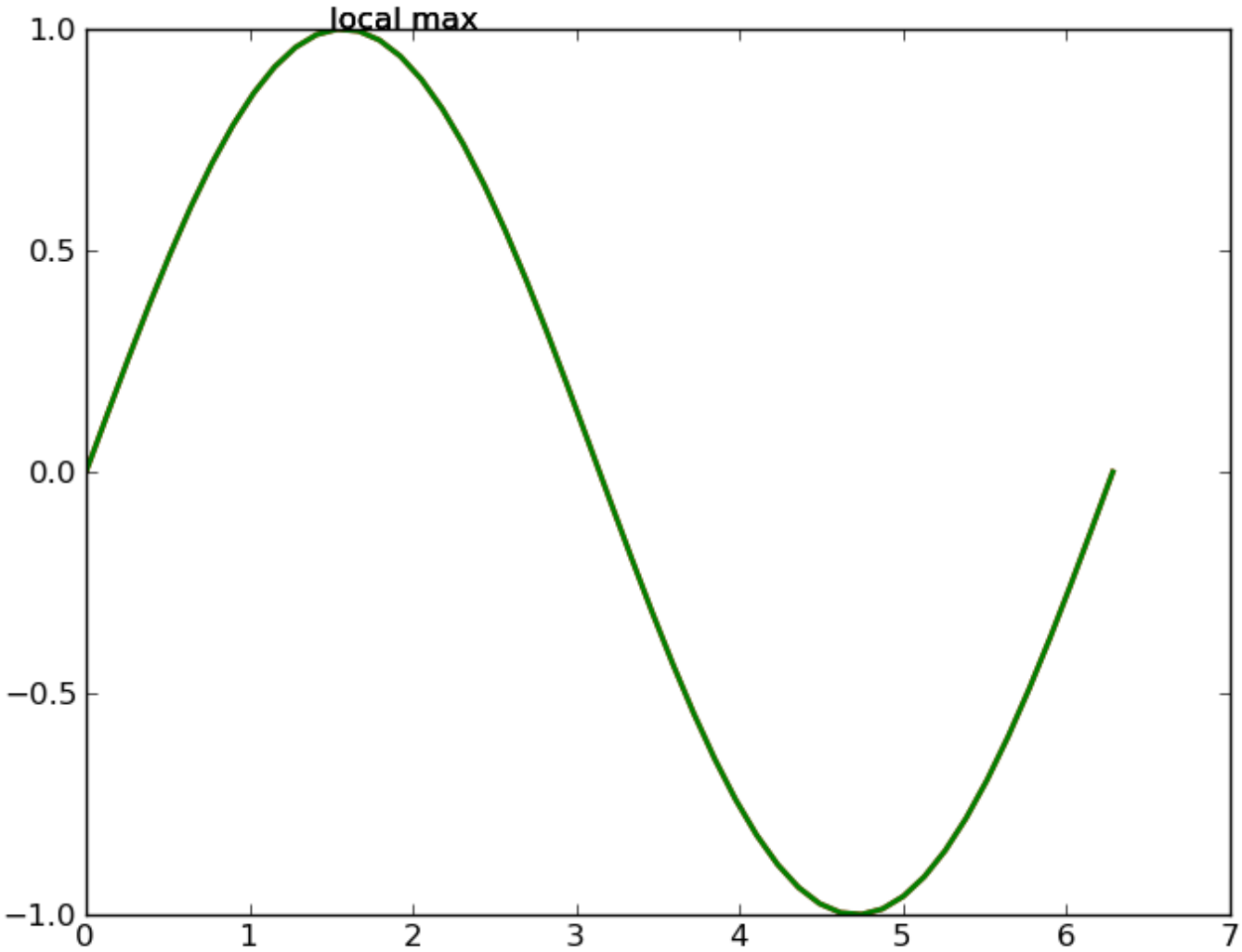
```
In []: plot(y, cos(y), 'r')
# See a red plot!

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()
```

Some supported formats:

- png
- pdf
- svg

Recap

- `linspace(start, end, num)`
- `plot(x, y)`
- `clf()`
- `xlabel, ylabel`
- `title, legend, annotate`
- `savefig`
- `close`

40 m

Outline

Overlaid Plots

```
In []: y = linspace(0, 2*pi, 50)

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)'])
# Note how we made two legends
```

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()
```

Get 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)
```

Get Axes lengths

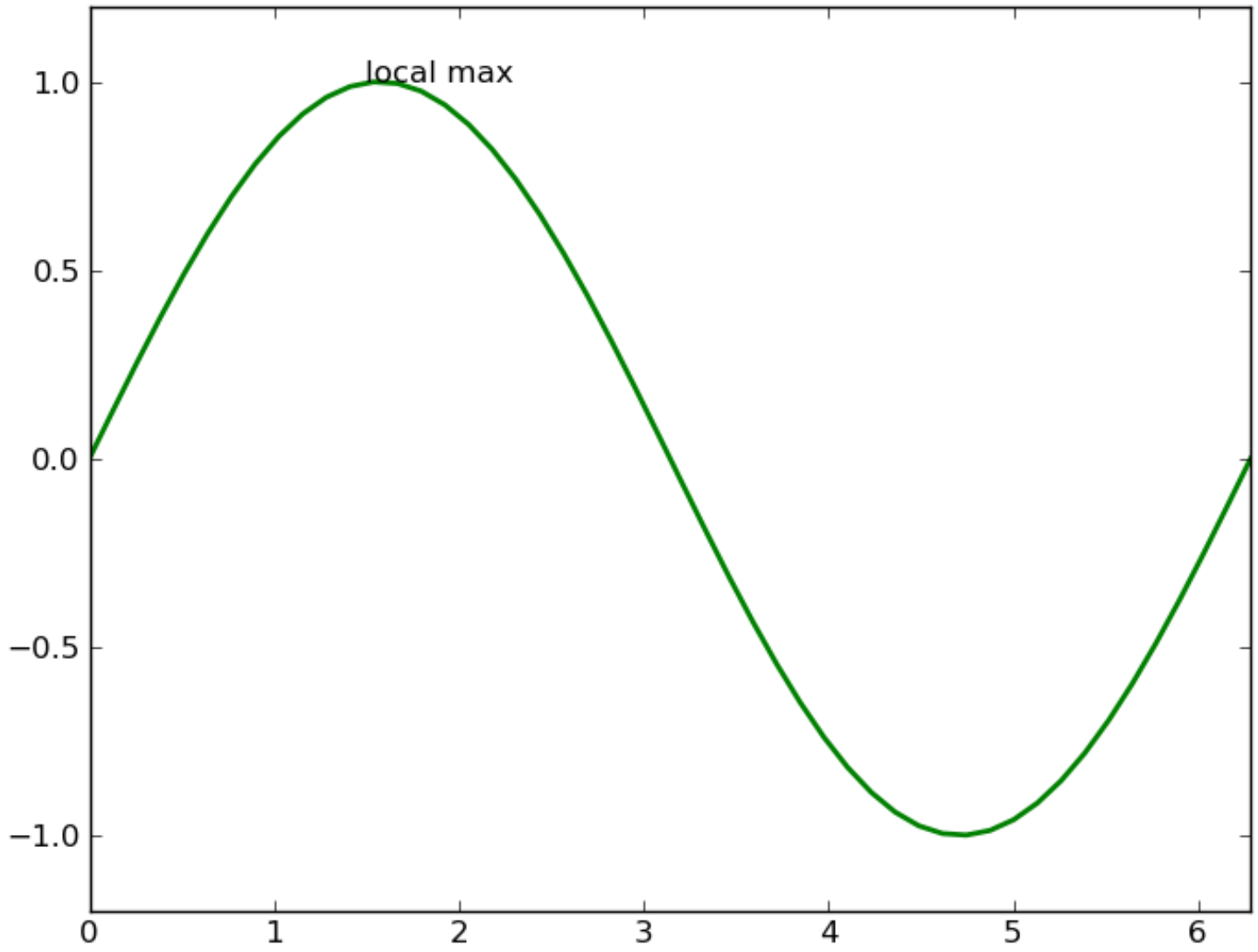
Getting axes lengths

```
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 tip: documentation

- Try:

  ```
  In []: plot?
  ```

  to get more information on **`plot`**
- Use arrow keys to scroll docs
- Note: exit help pager with “q”

IPython tip: source

- Try:

  `In []: plot??`

  to see the source code for **`plot`**

50 m

Review Problem

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