Introductory Scientific Computing with Python

Saving scripts

FOSSEE

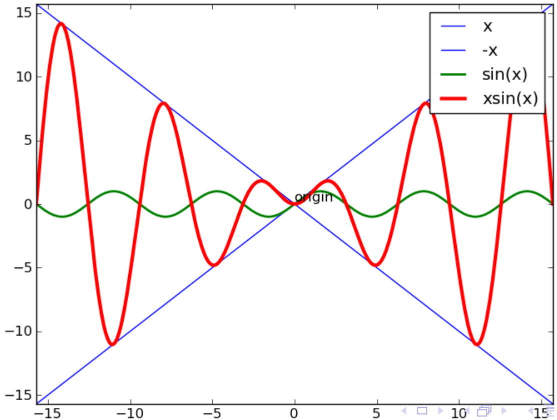
Department of Aerospace Engineering
IIT Bombay

Mumbai, India

Outline

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



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)
```

5 m

Outline

Command History

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

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

This displays the “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 the “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 specified individually separated by spaces or as a range separated by a dash.

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

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

Saving commands into a script

- Save lines relevant for the review problem
- Hint: example
  `%save four_plot.py 16 18-27`
- Choose the lines carefully
- Edit `four_plot.py` on Canopy
- Make sure all the lines are correct
- Save the script

10 m

Creating scripts: alternative

- Create a new file on Canopy
- Copy commands for assignment with your mouse
- Save the script to `four_plot.py`

Where is the script saved?

- `%save` saves into the current directory

- Use `%pwd` to print the current directory
- Use `%cd` to change the directory

- Question: how do you find out more about `%cd`?

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`

Where is the plot?

```
In []: show()
```

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`

Where is the plot?

```
In []: show()
```

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`

Where is the plot?

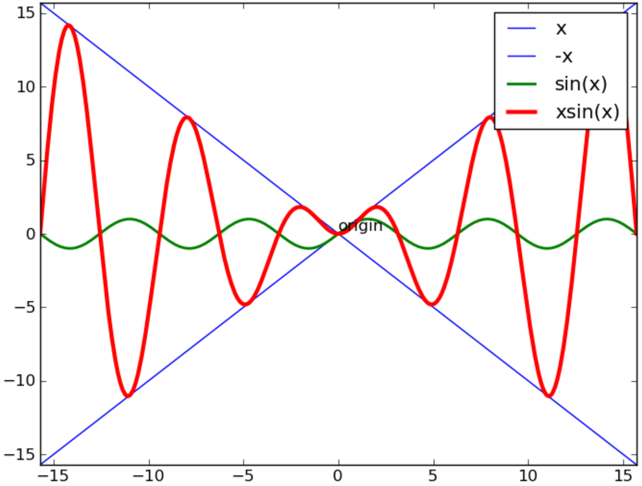
```
In []: show()
```

Exercise

- Add the **`show()`** command to **`four_plot.py`**
- Save the file
- Test that it works

15 m

Result graph



Running with Python

- Start a new Canopy terminal
- Change directory to where you saved `four_plot.py`
- Run the script as:

```
$ python four_plot.py
```

Do you see:

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

Running with Python

- Start a new Canopy terminal
- Change directory to where you saved **`four_plot.py`**
- Run the script as:

```
$ python four_plot.py
```

Do you see:

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

Imports

- **`ipython --pylab`** does magic
- Import libraries using **`import`**

```
In []: from pylab import linspace
```

20 m

Exercise

- Add `from pylab import linspace` to top of `four_plot.py`
- Test that it works

```
# four_plot.py
from pylab import linspace # <-- added
x = linspace(-5*pi, 5*pi, 500)
plot(x, x, 'b')
...
```

Try again

- On Canopy terminal
- Run the script as:

```
$ python four_plot.py
```

Do you see:

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

Try again

- On Canopy terminal
- Run the script as:

```
$ python four_plot.py
```

Do you see:

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

Exercise

- Change line 1 to `from pylab import *`
- Test that it works

```
# four_plot.py
from pylab import * # <-- added
x = linspace(-5*pi, 5*pi, 500)
plot(x, x, 'b')
...
```

25 m

Solution

```
from pylab import *
x = linspace(-5*pi, 5*pi, 500)
plot(x, x, 'b')
plot(x, -x, 'b')
plot(x, sin(x), 'g', linewidth=2)
plot(x, x*sin(x), 'r', linewidth=3)
legend(['x', '-x', 'sin(x)', 'xsin(x)'])
annotate('origin', xy = (0, 0))
xlim(-5*pi, 5*pi)
ylim(-5*pi, 5*pi)
show()
```

Note on script file names

- Should start with a letter
- Can use _ (underscore) and numbers
- No `.` allowed
- No spaces or special characters

Test

- `1_script.py`
- `script_1.py`
- `one11.py`
- `one script.py`
- `one,script;xxx.py`
- `one.two.py`

Using Canopy

- Much easier
- Write code in the editor
- Embedded IPython
- Save (Ctrl-S or Cmd-S)
- Run selection: Ctrl-Shift-R (Cmd-Shift-R on OS X)
- Run code: Ctrl-R (Cmd-R on OS X)
- Change directory with menu (`%cd`)

30 m

What did we learn?

- Starting up IPython
- Creating simple plots
- Annotating: labels, legends, annotation
- Changing the looks: color, linewidth
- Accessing history, documentation
- `%hist` - History of commands
- Creating a Python script with `%save`
- Running a script using `%run -i`
- Importing functionality
- Running a script with `python script.py`