Python for Science and Engg: Plotting experimental data

FOSSEE

Department of Aerospace Engineering
IIT Bombay

25 September, 2010
Day 1, Session 2

Outline

Outline

Why would I plot f(x)?

Do we plot analytical functions or experimental data?

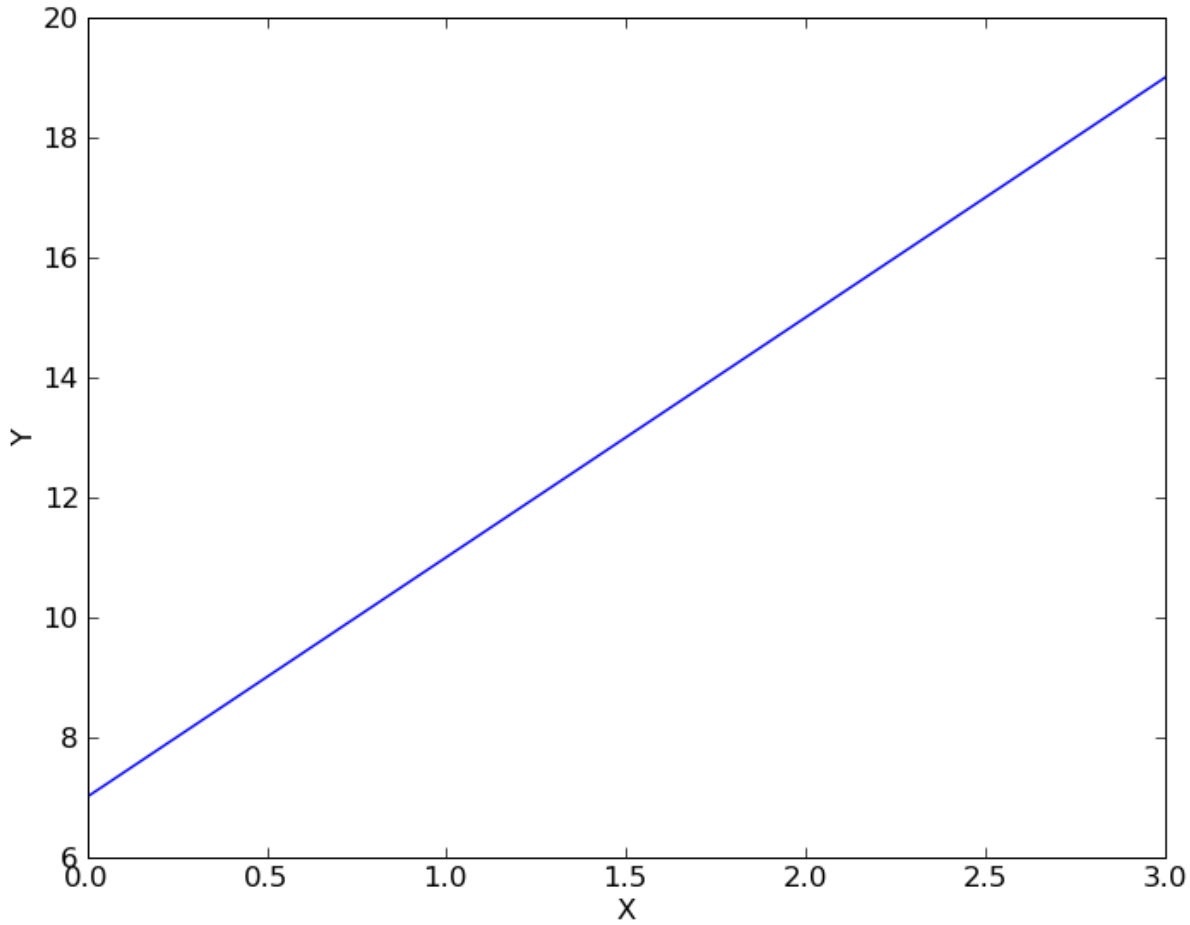
```
In [ ]: time = [0, 1, 2, 3]

In [ ]: distance = [7, 11, 15, 19]

In [ ]: plot(time,distance)
Out[ ]: [<matplotlib.lines.Line2D object at 0xa73a

In [ ]: xlabel('time')
Out[ ]: <matplotlib.text.Text object at 0x986e9ac>

In [ ]: ylabel('distance')
Out[ ]: <matplotlib.text.Text object at 0x98746ec>
```



Is this what you have?

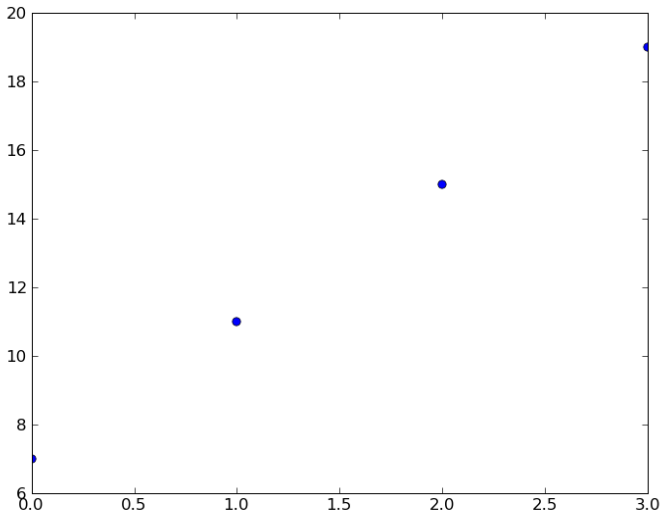
Plotting points

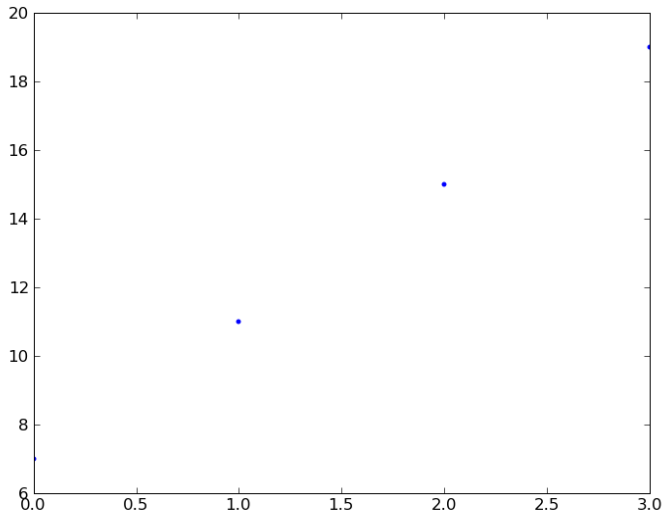
- What if we want to plot the points!

```
In [ ]: clf()

In [ ]: plot(time, distance, 'o')
Out[ ]: [<matplotlib.lines.Line2D object

In [ ]: clf()
In [ ]: plot(time, distance, '.')
Out[ ]: [<matplotlib.lines.Line2D object
```


20
18
16
14
12
10
8
6
0.0
0.5
1.0
1.5
2.0
2.5
3.0


20
18
16
14
12
10
8
6
0.0
0.5
1.0
1.5
2.0
2.5
3.0

Additional Plotting Attributes

- `'o'` - Filled circles
- `'.'` - Small Dots
- `'-'` - Lines
- `'--'` - Dashed lines

Outline

Lists: Introduction

```
In [ ]: time = [0, 1, 2, 3]

In [ ]: distance = [7, 11, 15, 19]
```

What are **x** and **y**?

lists!!

Lists: Initializing & accessing elements

```
In []: mtlist = []
```

Empty List

```
In []: p = [ 2, 3, 5, 7]

In []: p[1]
Out[]: 3

In []: p[0]+p[1]+p[-1]
Out[]: 12
```

List: Slicing

Remember...

```
In []:  p = [ 2, 3, 5, 7]
```

```
In []: p[1:3]
Out[]: [3, 5]
```

A slice

```
In []: p[0:-1]
Out[]: [2, 3, 5]
In []: p[::2]
Out[]: [2, 5]
```

```
list[initial:final:step]
```

List operations

```
In [ ]: b = [ 11, 13, 17]
In [ ]: c = p + b

In [ ]: c
Out[ ]: [2, 3, 5, 7, 11, 13, 17]

In [ ]: p.append(11)
In [ ]: p
Out[ ]: [ 2, 3, 5, 7, 11]
```

Outline

Simple Pendulum - L and T

Let us look at the Simple Pendulum experiment.

L	T	T^2
0.1	0.69	
0.2	0.90	
0.3	1.19	
0.4	1.30	
0.5	1.47	
0.6	1.58	
0.7	1.77	
0.8	1.83	
0.9	1.94	

$L \alpha T^2$

Lets use lists

```
In []: L = [0.1, 0.2, 0.3, 0.4, 0.5,
            0.6, 0.7, 0.8, 0.9]

In []: t = [0.69, 0.90, 1.19,
            1.30, 1.47, 1.58,
            1.77, 1.83, 1.94]
```

Plotting L vs T^2

- We must square each of the values in `t`
- How to do it?
- We use a `for` loop to iterate over `t`

Plotting L vs T^2

```
In []: tsq = []

In []: for time in t:
  ....:     tsq.append(time*time)
  ....:
  ....:
```

This gives `tsq` which is the list of squares of `t` values.

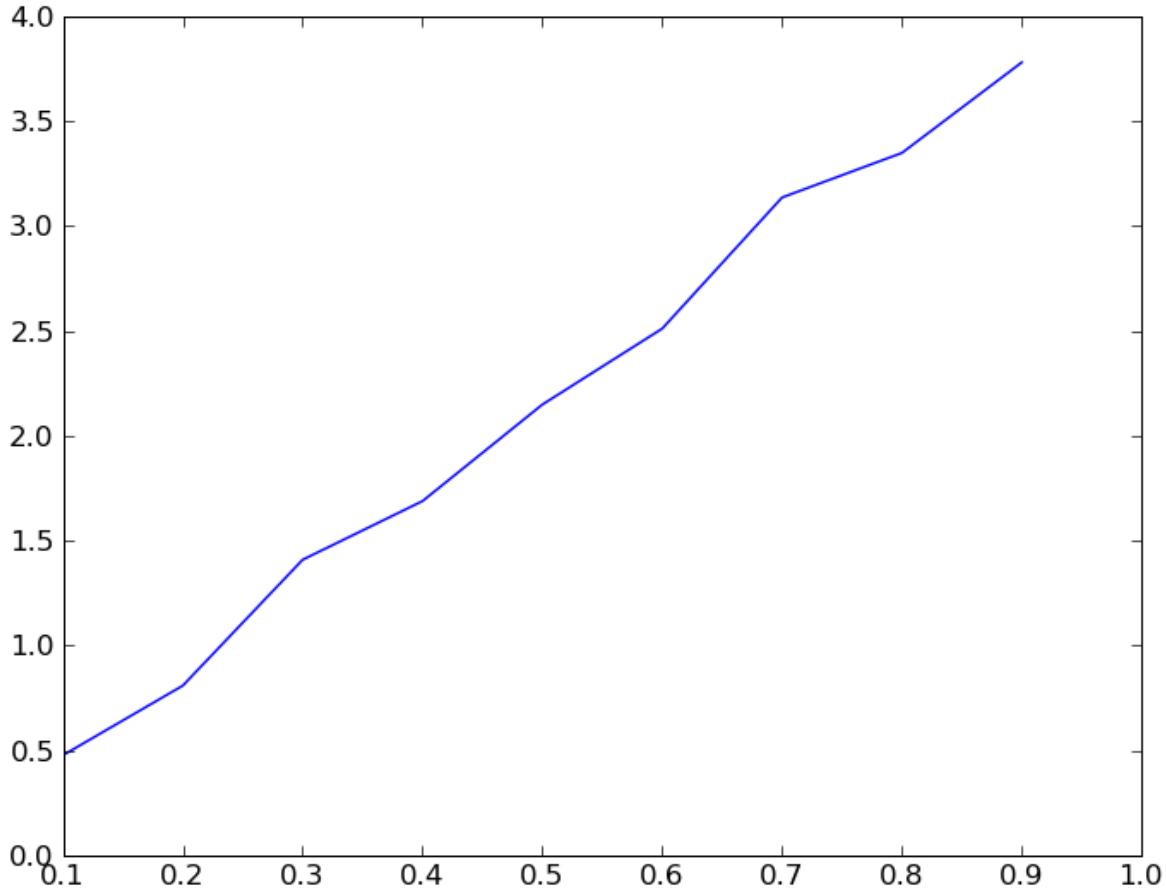
```
In []: print len(L), len(t), len(tsq)
Out[]: 9 9 9
```

How to come out of the `for` loop?

Hitting the “ENTER” key twice returns the cursor to the previous indentation level

```
In []: for time in t:
  ....:     tsq.append(time*time)
  ....:
  ....:

In []: plot(L, tsq)
```


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

What about larger data sets?

Data is usually present in a file!

Lets look at the **pendulum.txt** file.

```
In []: cat pendulum.txt
1.0000e-01 6.9004e-01
1.1000e-01 6.9497e-01
1.2000e-01 7.4252e-01
1.3000e-01 7.5360e-01

...
```

Reading `pendulum.txt`

- File contains L vs. T values
- First Column - L values
- Second Column - T values
- Let us generate a plot from the data file

Plotting from `pendulum.txt`

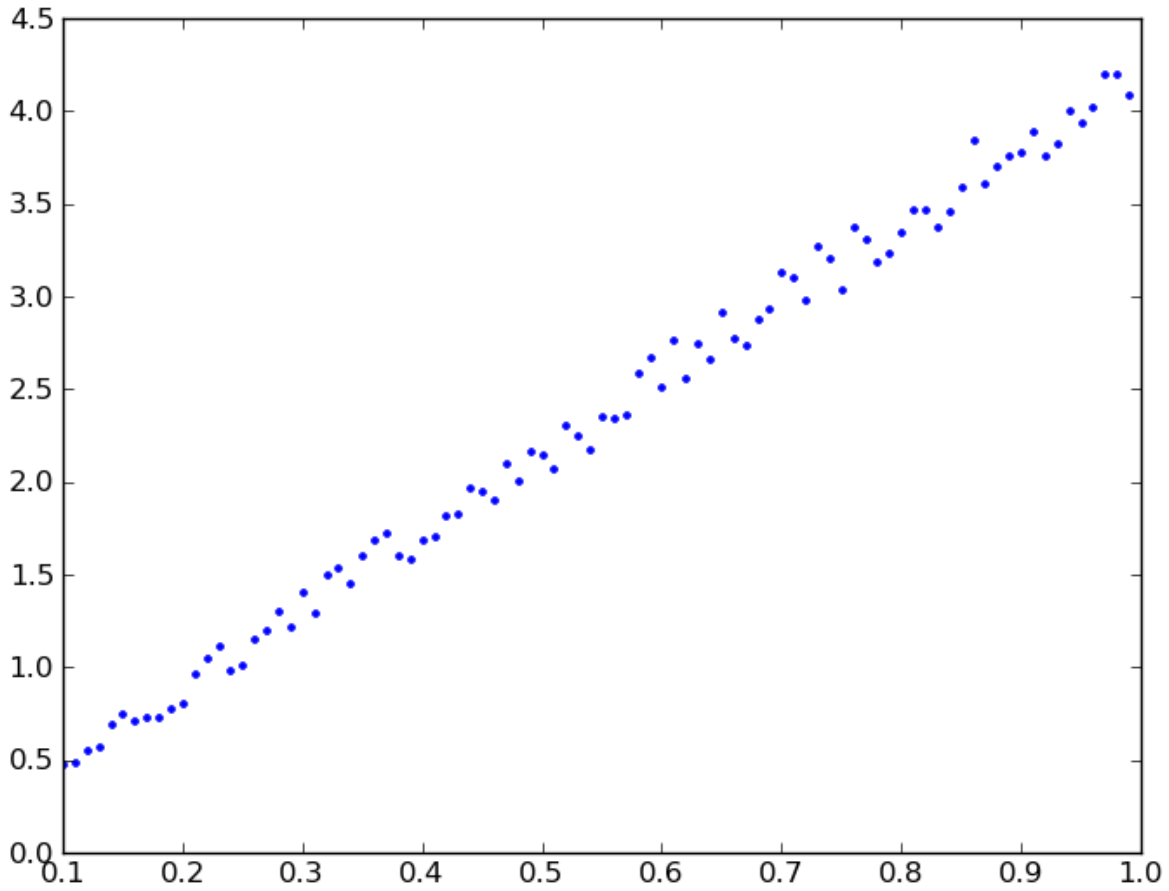
Open a new script

Save as **`pendulum_plot.py`** after typing first line

```
L = []
t = []
for line in open('pendulum.txt'):
    point = line.split()
    L.append(float(point[0]))
    t.append(float(point[1]))
tsq = []
for time in t:
    tsq.append(time*time)
plot(L, tsq, '.')
```

Save and run

- Save as pendulum_plot.py.
- Run using `%run -i pendulum_plot.py`


4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

Reading files . . .

```
for line in open('pendulum.txt'):
```

- opening file '**pendulum.txt**'
- reading the file line by line
- **line** is a **string**

Outline

Strings

Anything within “quotes” is a string!

```
' This is a string '
" This too! "
""" This one too! """
''' And one more! '''
```

Strings and `split()`

```
In [ ]: greet = 'hello world'

In [ ]: greet.split()
Out[ ]: ['hello', 'world']
```

This is what happens with **`line`**

```
In [ ]: line = '1.20 7.42'

In [ ]: point = line.split()

In [ ]: point
Out[ ]: ['1.20', '7.42']
```

Getting floats from strings

```
In [ ]: type(point[0])
Out[ ]: <type 'str'>
```

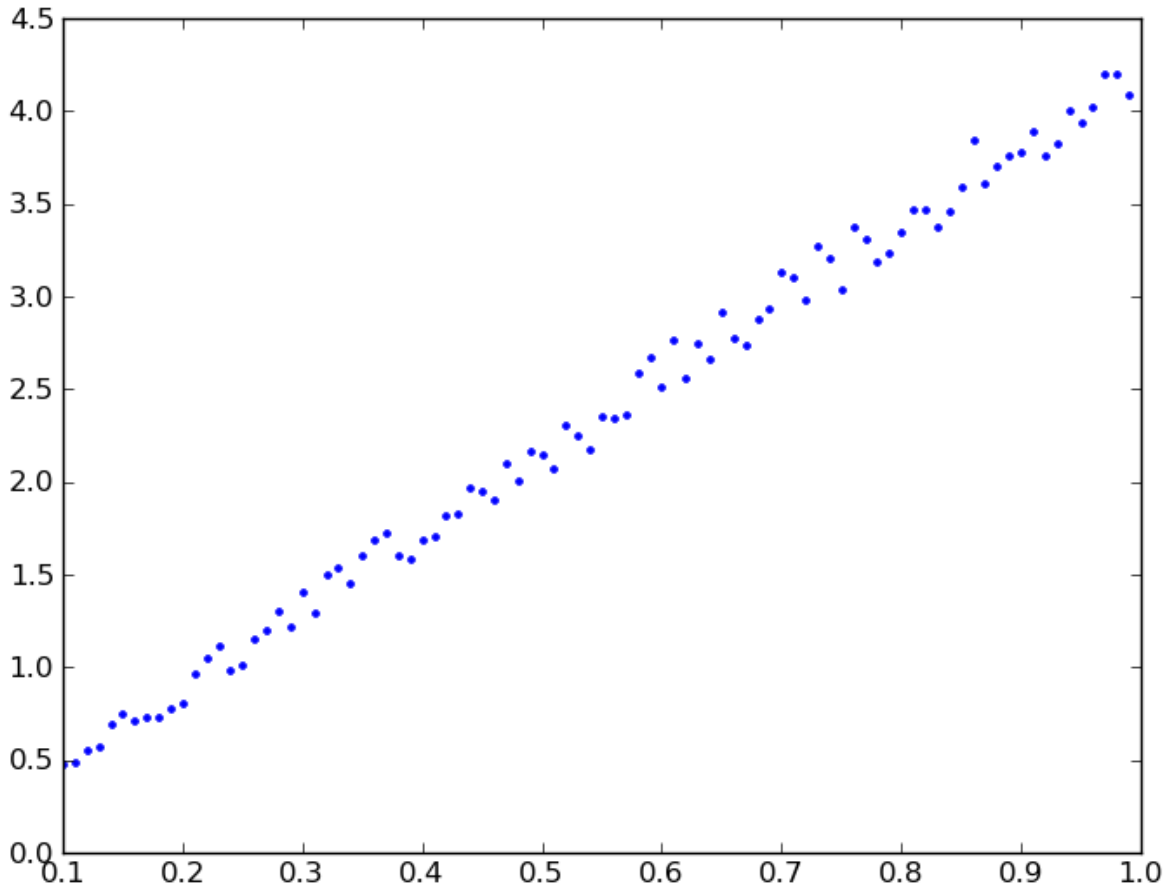
But, we need floating point numbers

```
In [ ]: t = float(point[0])

In [ ]: type(t)
Out[ ]: <type 'float'>
```

Let's review the code

```
L = []
t = []
for line in open('pendulum.txt'):
    point = line.split()
    L.append(float(point[0]))
    t.append(float(point[1]))
tsq = []
for time in t:
    tsq.append(time*time)
plot(L, tsq, '.')
```


4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

Outline

What did we learn?

- Plotting points
- Plot attributes
- Lists
- `for`
- Reading files
- Tokenizing
- Strings