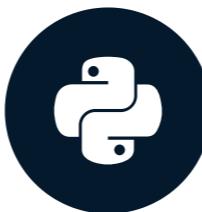


Plotting multiple graphs

INTRODUCTION TO DATA VISUALIZATION IN PYTHON



Bryan Van de Ven
Core Developer of Bokeh

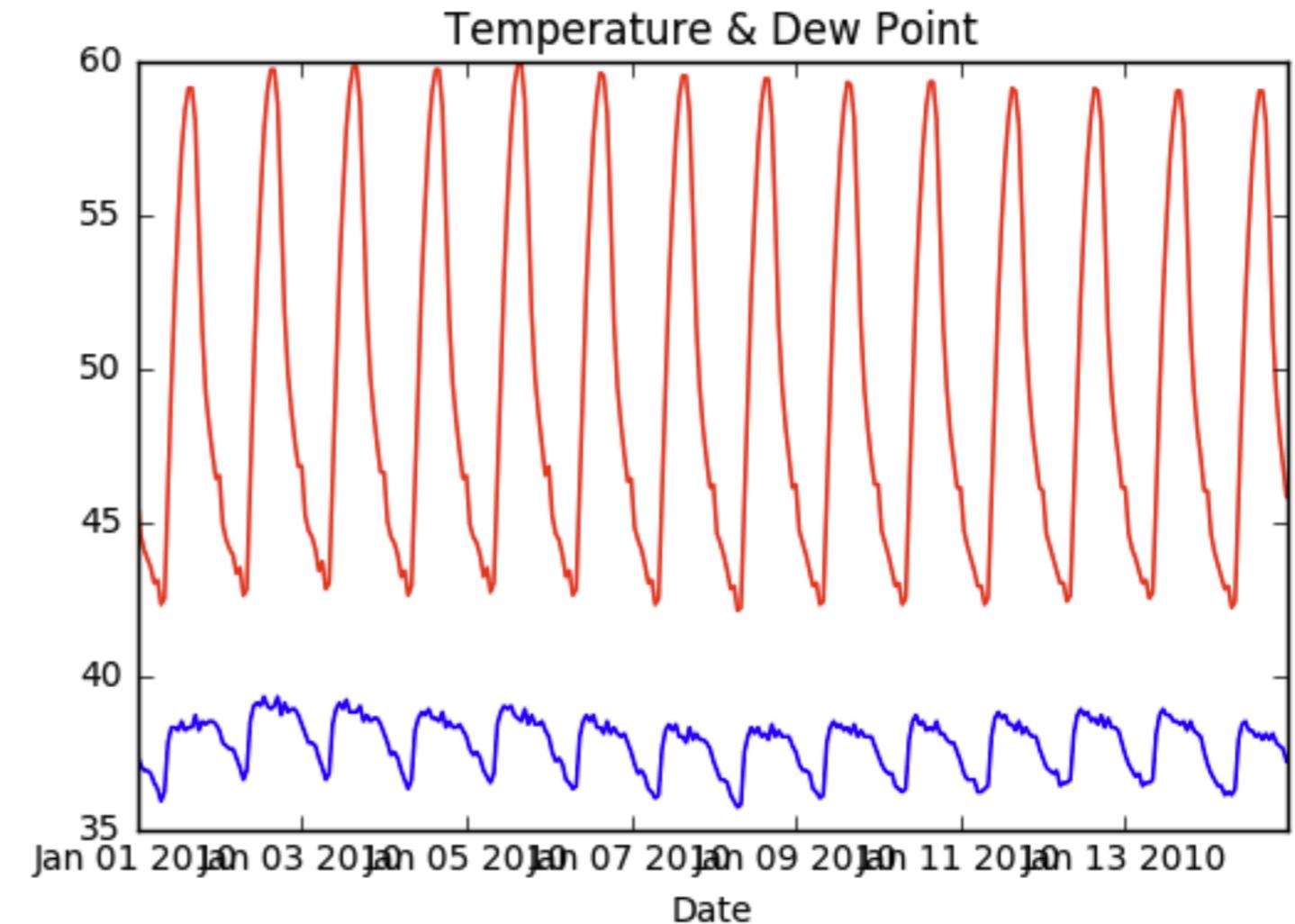
Strategies

- Plotting many graphs on common axes
- Creating axes within a figure
- Creating subplots within a figure

Graphs on common axes

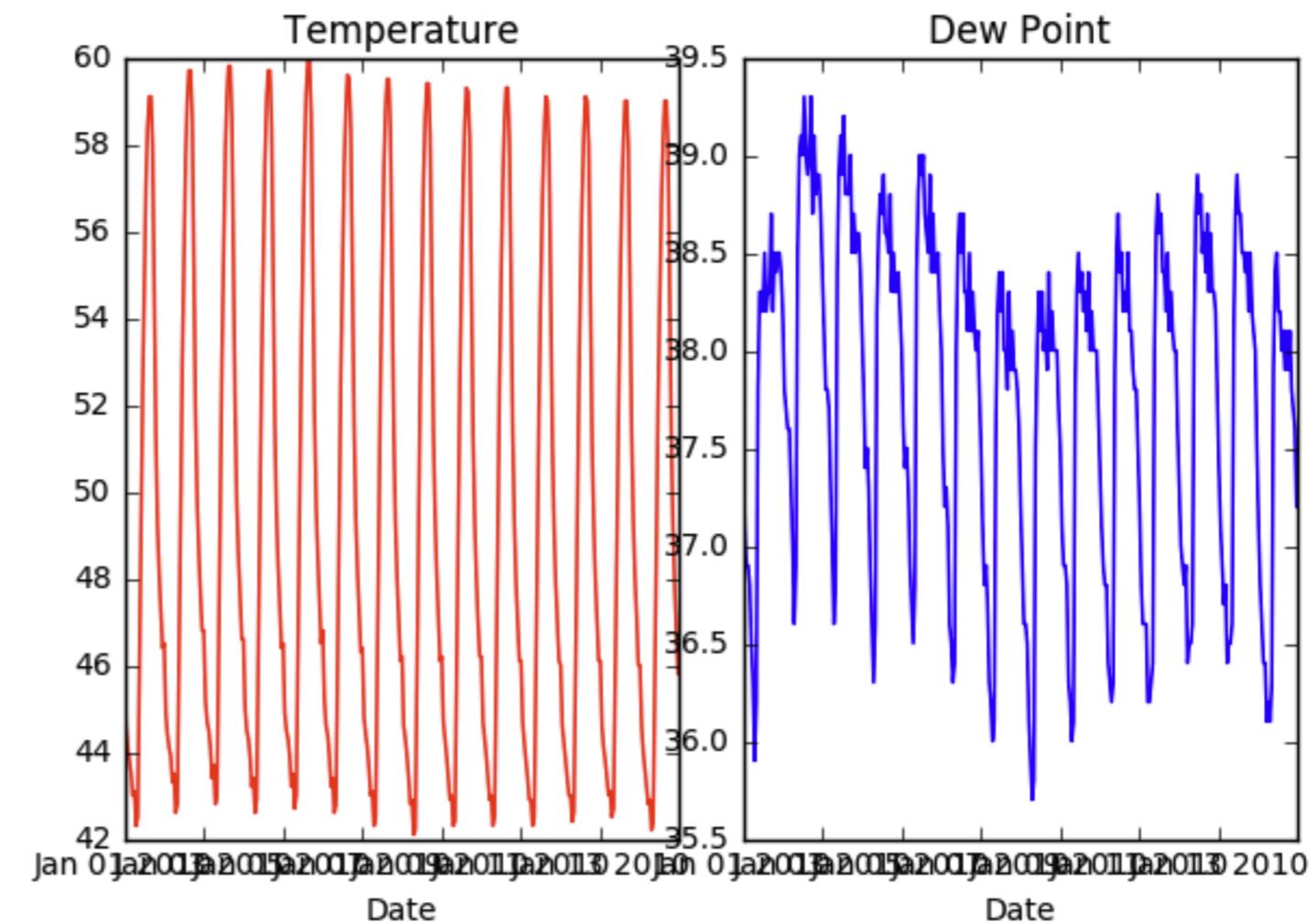
```
import matplotlib.pyplot as plt

plt.plot(t, temperature, 'r')
# Appears on same axes
plt.plot(t, dewpoint, 'b')
plt.xlabel('Date')
plt.title('Temperature & Dew Point')
# Renders plot objects to screen
plt.show()
```



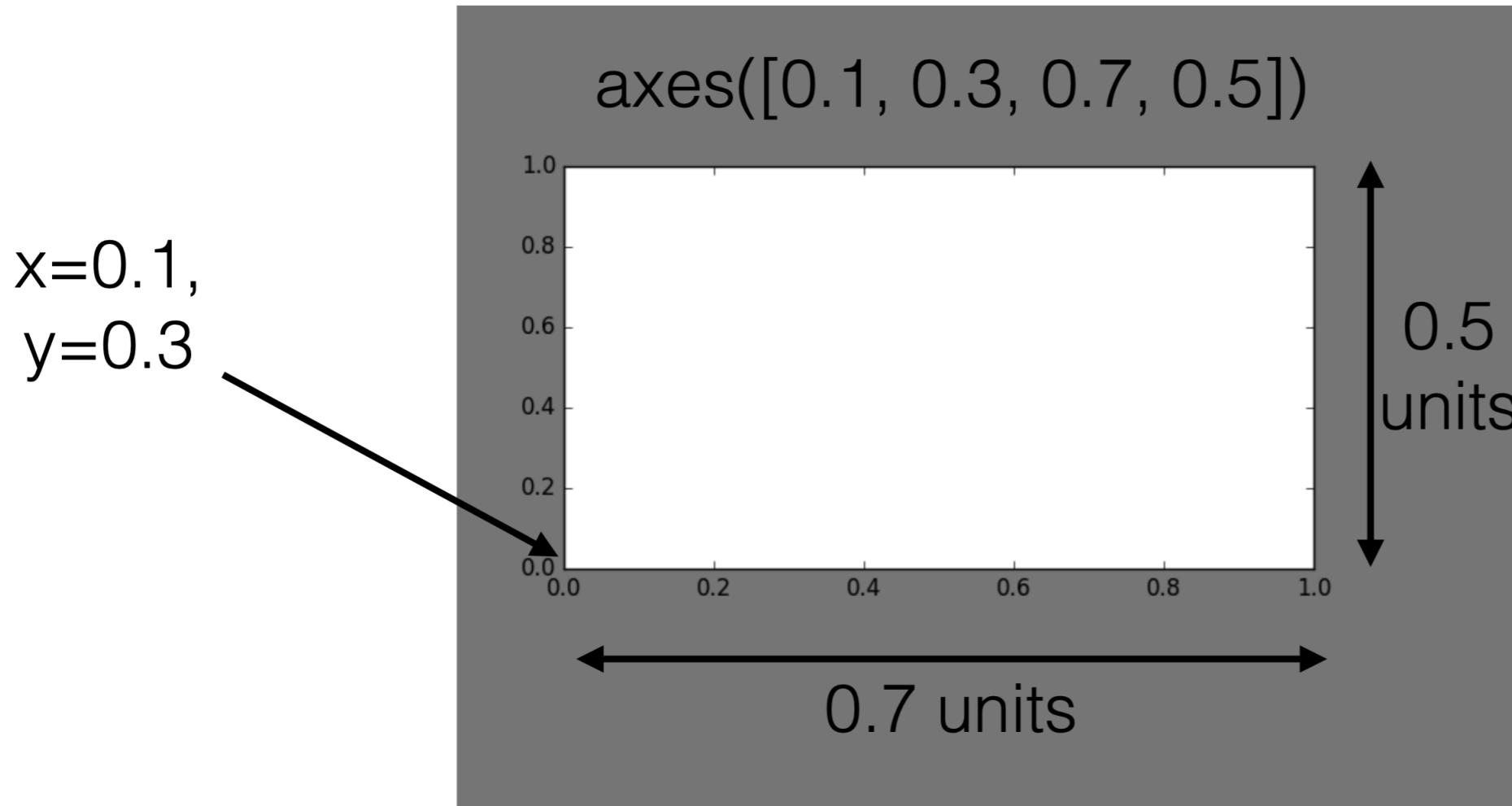
Using axes()

```
plt.axes([0.05,0.05,0.425,0.9])  
plt.plot(t, temperature, 'r')  
plt.xlabel('Date')  
plt.title('Temperature')  
plt.axes([0.525,0.05,0.425,0.9])  
plt.plot(t, dewpoint, 'b')  
plt.xlabel('Date')  
plt.title('Dew Point')  
plt.show()
```

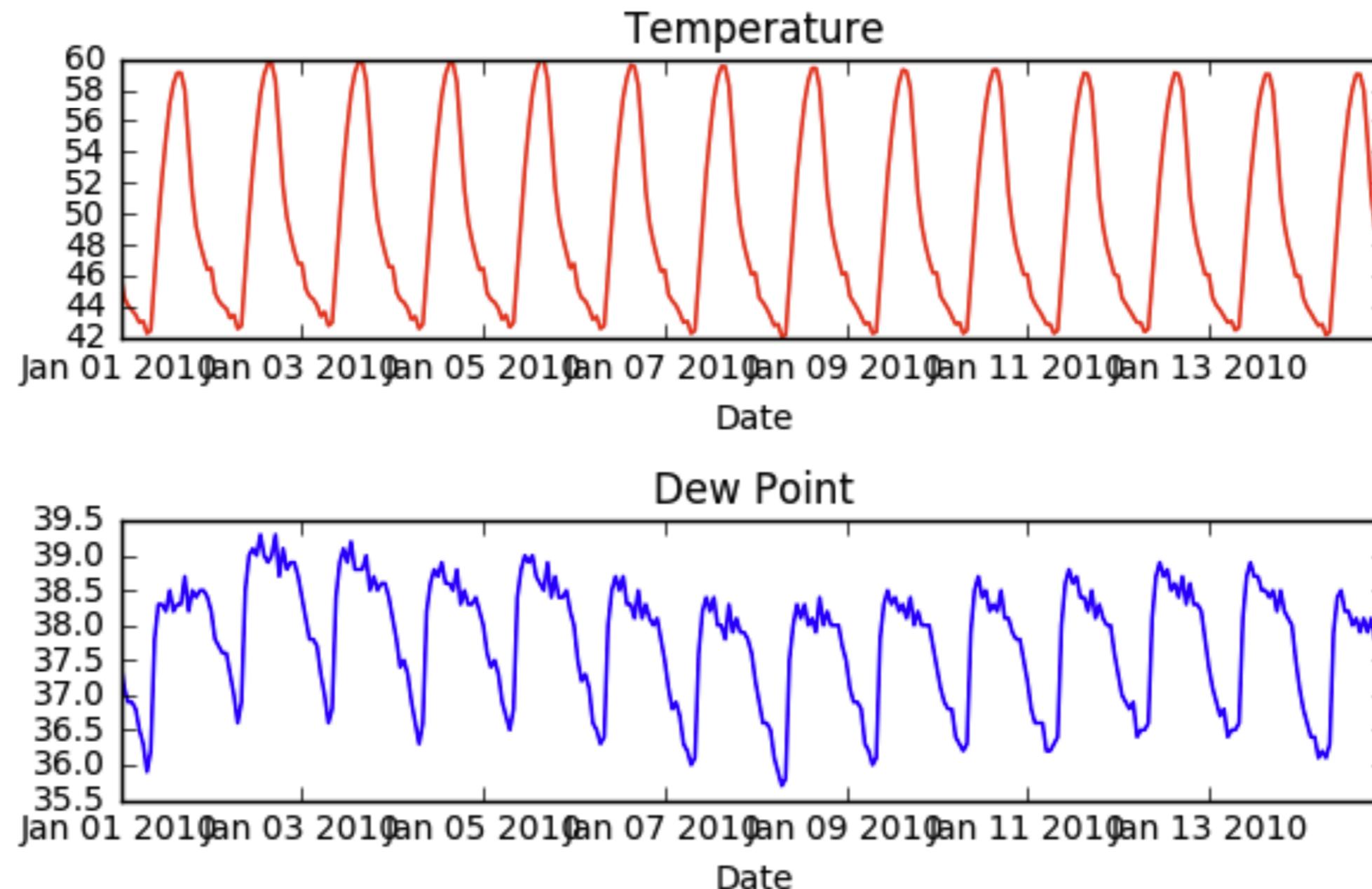


The axes() command

- Syntax: `axes([x_lo, y_lo, width, height])`
- Units between 0 and 1 (figure dimensions)



Using subplot()



Using subplot()

```
plt.subplot(2, 1, 1)  
plt.plot(t, temperature, 'r')  
plt.xlabel('Date')  
plt.title('Temperature')  
  
plt.subplot(2, 1, 2)  
plt.plot(t, dewpoint, 'b')  
plt.xlabel('Date')  
plt.title('Dew Point')  
  
plt.tight_layout()  
plt.show()
```

The subplot() command

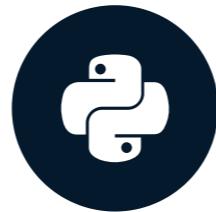
- Syntax: `subplot(nrows, ncols, nsubplot)`
- Subplot ordering:
 - Row-wise from top left
 - Indexed from 1

Let's practice!

INTRODUCTION TO DATA VISUALIZATION IN PYTHON

Customizing axes

INTRODUCTION TO DATA VISUALIZATION IN PYTHON



Bryan Van de Ven

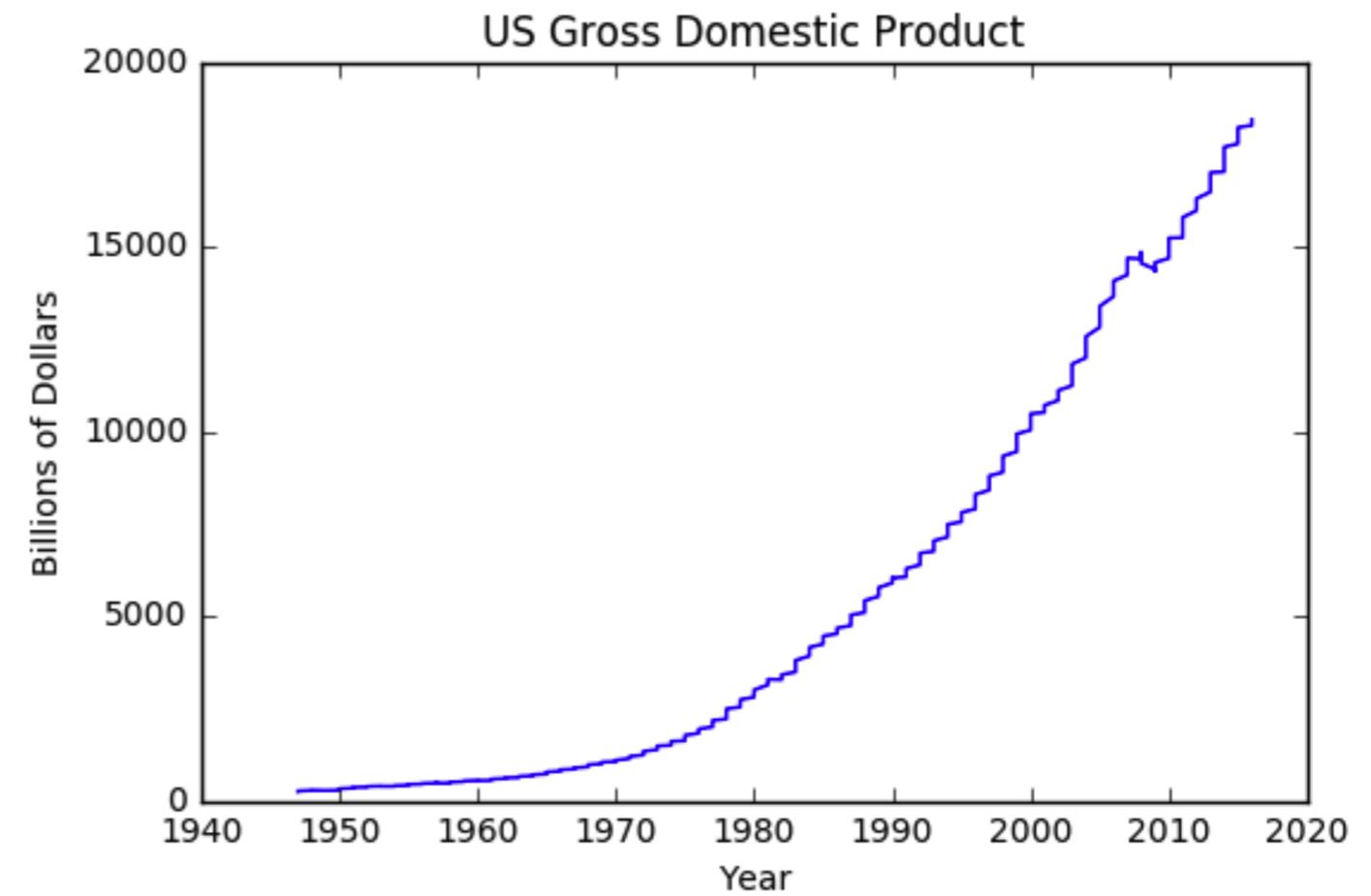
Core Developer of Bokeh

Controlling axis extents

- `axis([xmin, xmax, ymin, ymax])` sets axis extents
- Control over individual axis extents
 - `xlim([xmin, xmax])`
 - `ylim([ymin, ymax])`
- Can use tuples, lists for extents
 - e.g., `xlim((-2, 3))` works
 - e.g., `xlim([-2, 3])` works also

GDP over time

```
import matplotlib.pyplot as plt  
plt.plot(yr, gdp)  
plt.xlabel('Year')  
plt.ylabel('Billions of Dollars')  
plt.title('US Gross Domestic Product')  
  
plt.show()
```

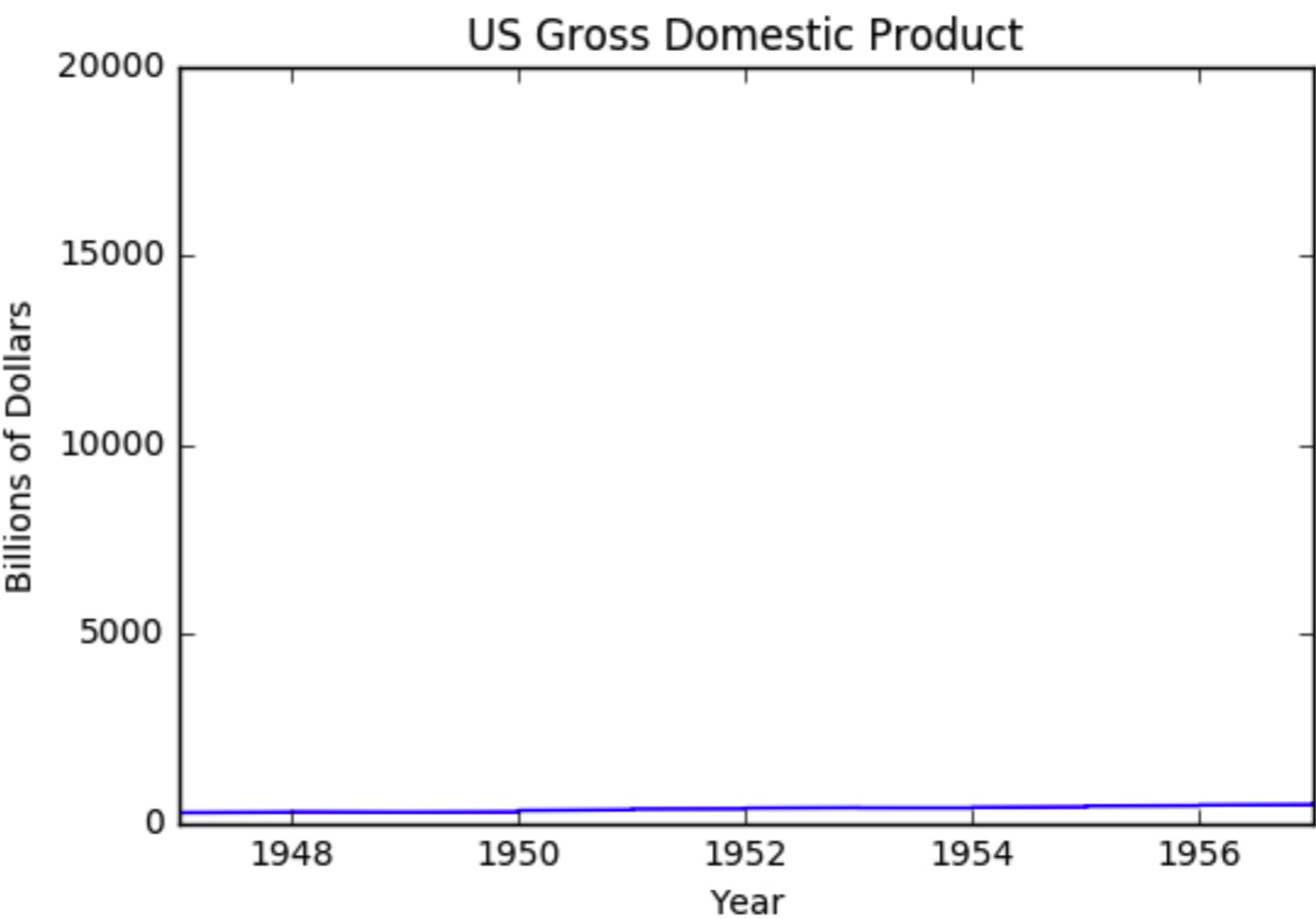


Using xlim()

```
plt.plot(yr, gdp)
plt.xlabel('Year')
plt.ylabel('Billions of Dollars')
plt.title('US Gross Domestic Product')

plt.xlim((1947, 1957))

plt.show()
```

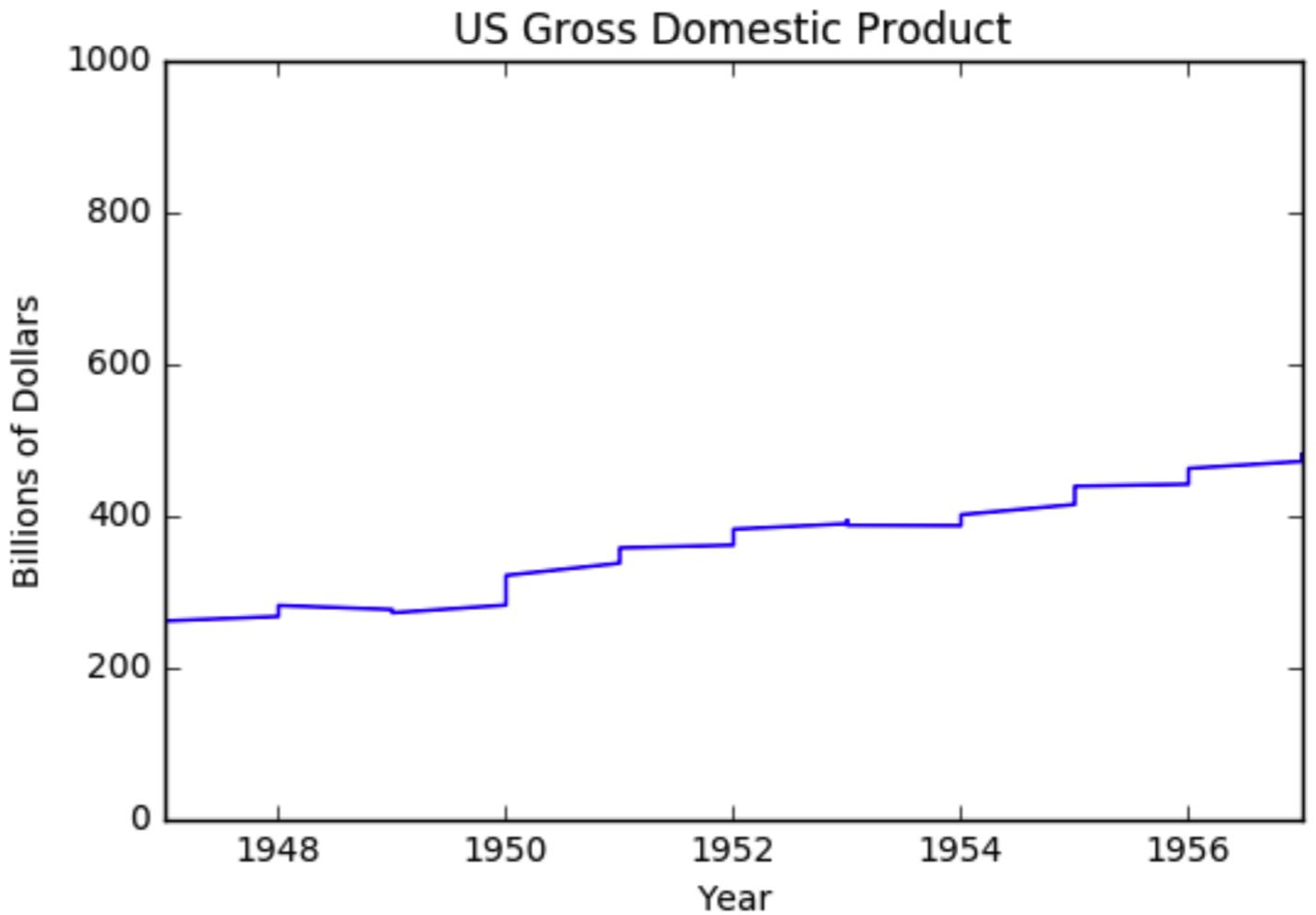


Using xlim() and ylim()

```
plt.plot(yr, gdp)
plt.xlabel('Year')
plt.ylabel('Billions of Dollars')
plt.title('US Gross Domestic Product')

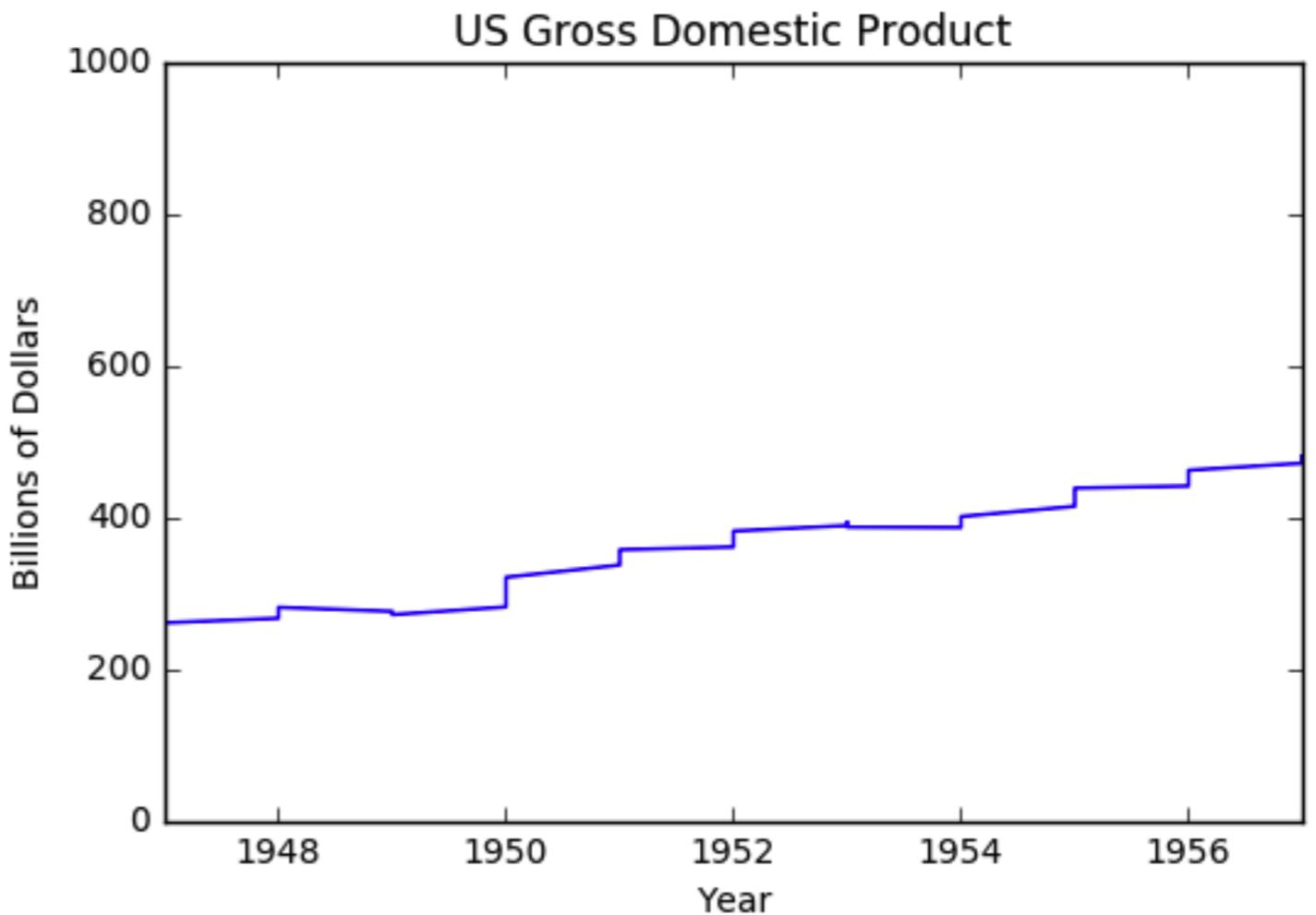
plt.xlim((1947, 1957))
plt.ylim((0, 1000))

plt.show()
```



Using axis()

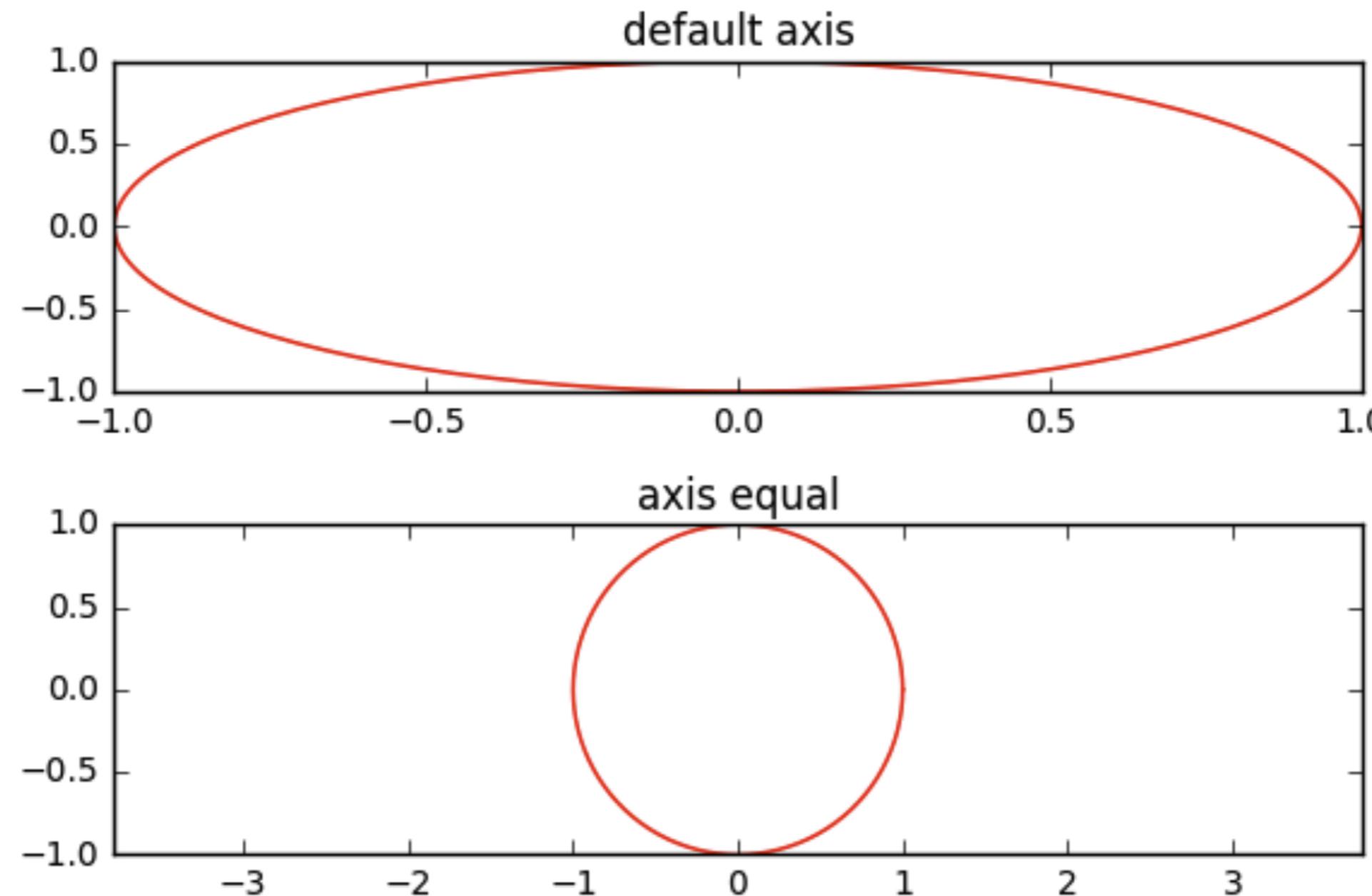
```
plt.plot(yr, gdp)  
plt.xlabel('Year')  
plt.ylabel('Billions of Dollars')  
plt.title('US Gross Domestic Product')  
  
plt.axis((1947, 1957, 0, 600))  
  
plt.show()
```



Other axis() options

Invocation	Result
<code>axis('off')</code>	turns off axis lines, labels
<code>axis('equal')</code>	equal scaling on x and y axes
<code>axis('square')</code>	forces square plot
<code>axis('tight')</code>	sets <code>xlim</code> , <code>ylim</code> to show all data

Using axis('equal')



Using axis('equal')

```
plt.subplot(2, 1, 1)
```

```
plt.plot(x, y, 'red')
```

```
plt.title('default axis')
```

```
plt.subplot(2, 1, 2)
```

```
plt.plot(x, y, 'red')
```

```
plt.axis('equal')
```

```
plt.title('axis equal')
```

```
plt.tight_layout()
```

```
plt.show()
```

Let's practice!

INTRODUCTION TO DATA VISUALIZATION IN PYTHON

Legends, annotations, and styles

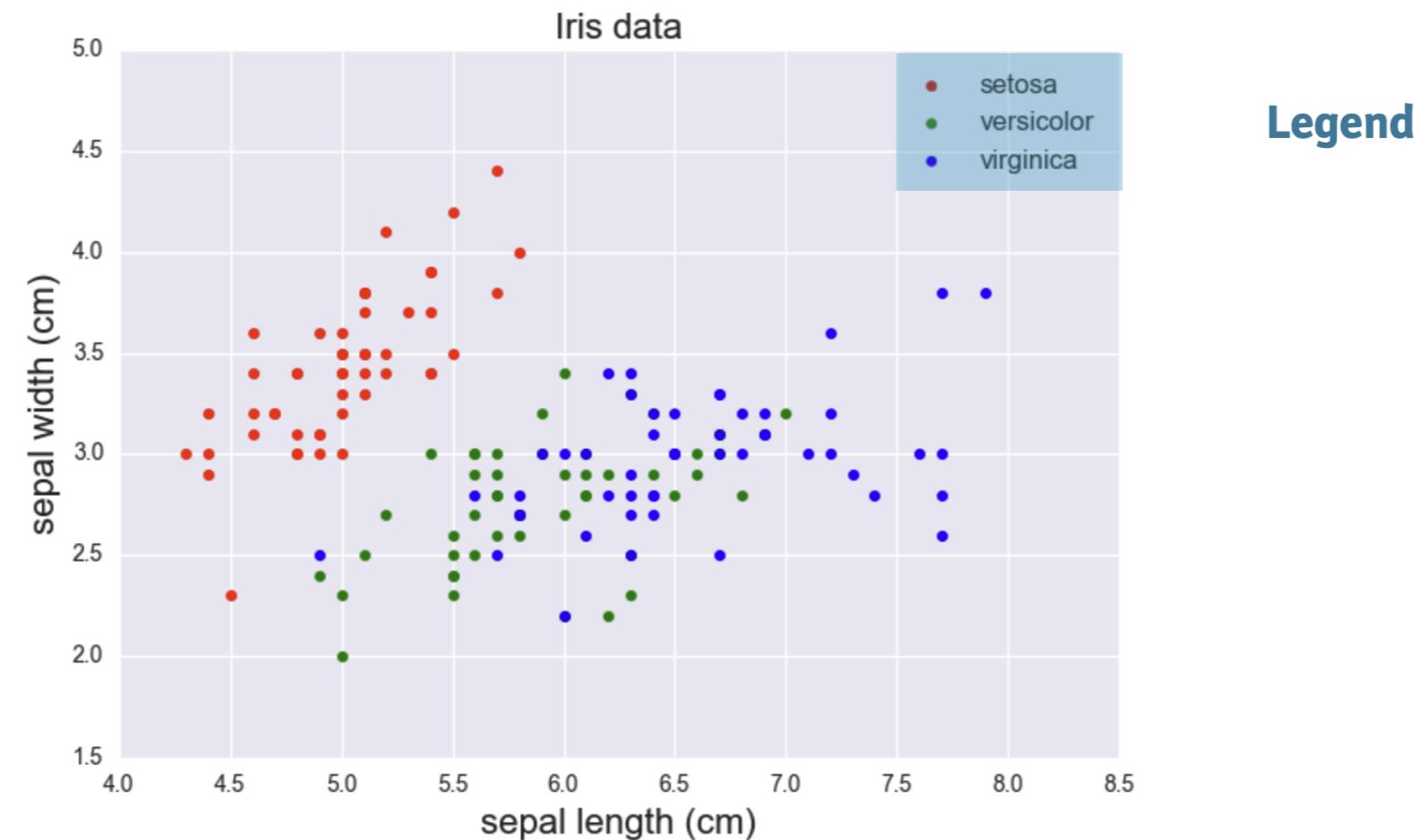
INTRODUCTION TO DATA VISUALIZATION IN PYTHON

Bryan Van de Ven
Core Developer of Bokeh



Legends

- Legend
 - provide labels for overlaid points and curves



Using legend()

```
import matplotlib.pyplot as plt
plt.scatter(setosa_len, setosa_wid,
            marker='o', color='red', label='setosa')

plt.scatter(versicolor_len, versicolor_wid,
            marker='o', color='green', label='versicolor')

plt.scatter(virginica_len, virginica_wid,
            marker='o', color='blue', label='virginica')
plt.legend(loc='upper right')
plt.title('Iris data')
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.show()
```

Legend locations

string	code
'upper left'	2
'center left'	6
'lower left'	3
'upper center'	9
'center'	10
'lower center'	8

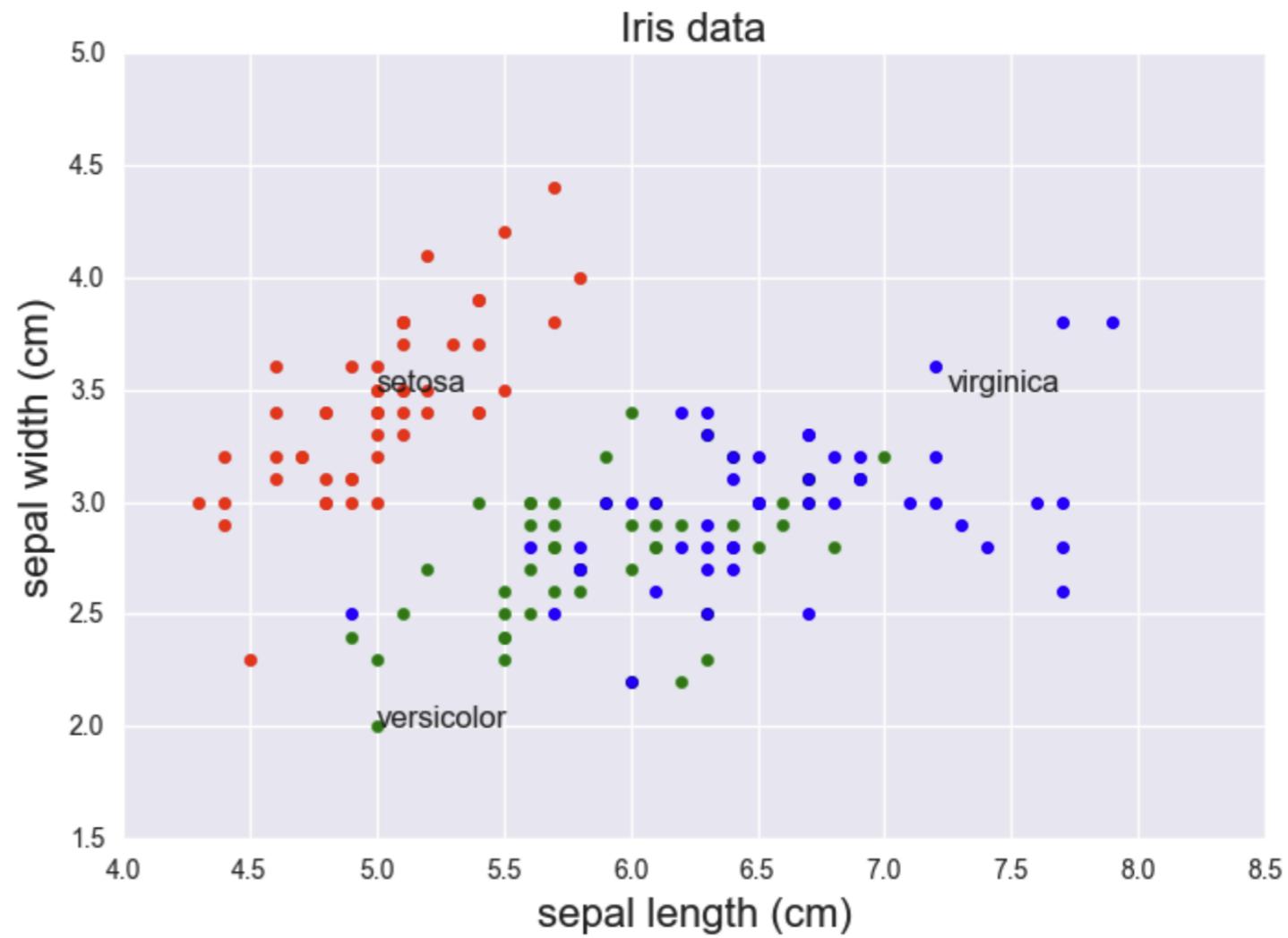
string	code
'upper right'	1
'center right'	7
'lower right'	4
'right'	5
'best'	0

Plot annotations

- Text labels and arrows using `annotate()` method
- Flexible specification of coordinates
- Keyword `arrowprops`: `dict` of arrow properties
 - `width`
 - `color`
 - etc.

Using `annotate()` for text

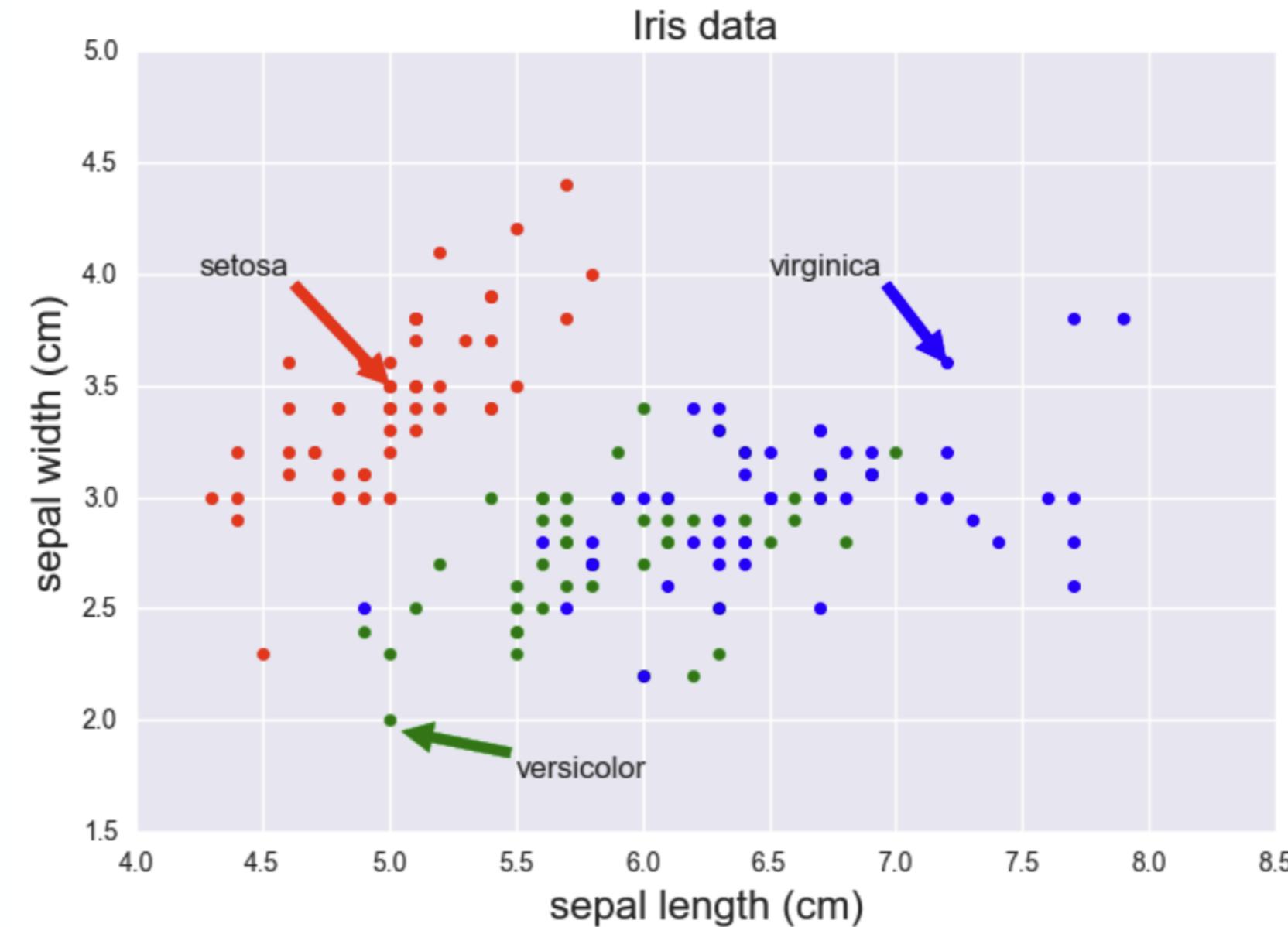
```
plt.annotate('setosa', xy=(5.0, 3.5))  
plt.annotate('virginica', xy=(7.25, 3.5))  
plt.annotate('versicolor', xy=(5.0, 2.0))  
plt.show()
```



Options for `annotate()`

option	description
s	text of label
xy	coordinates to annotate
xytext	coordinates of label
arrowprops	controls drawing of arrow

Using `annotate()` for arrows



Using `annotate()` for arrows

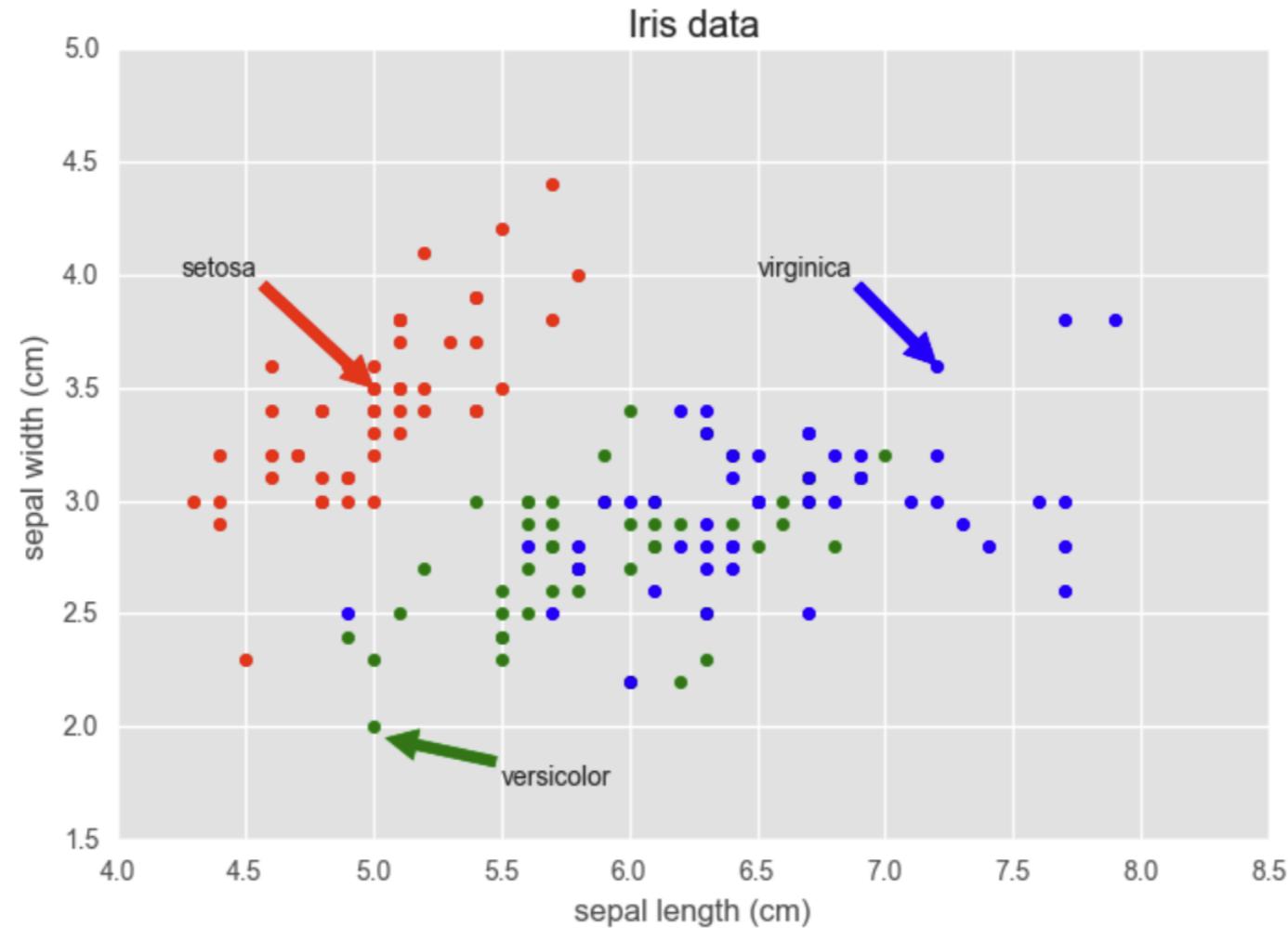
```
plt.annotate('setosa', xy=(5.0, 3.5),  
             xytext=(4.25, 4.0),  
             arrowprops={'color':'red'})  
  
plt.annotate('virginica', xy=(7.2, 3.6),  
             xytext=(6.5, 4.0),  
             arrowprops={'color':'blue'})  
  
plt.annotate('versicolor', xy=(5.05, 1.95),  
             xytext=(5.5, 1.75),  
             arrowprops={'color':'green'})  
  
plt.show()
```

Working with plot styles

- Style sheets in `matplotlib`
- Defaults for lines, points, backgrounds, etc.
- Switch styles globally with `plt.style.use()`
- `plt.style.available` : list of styles

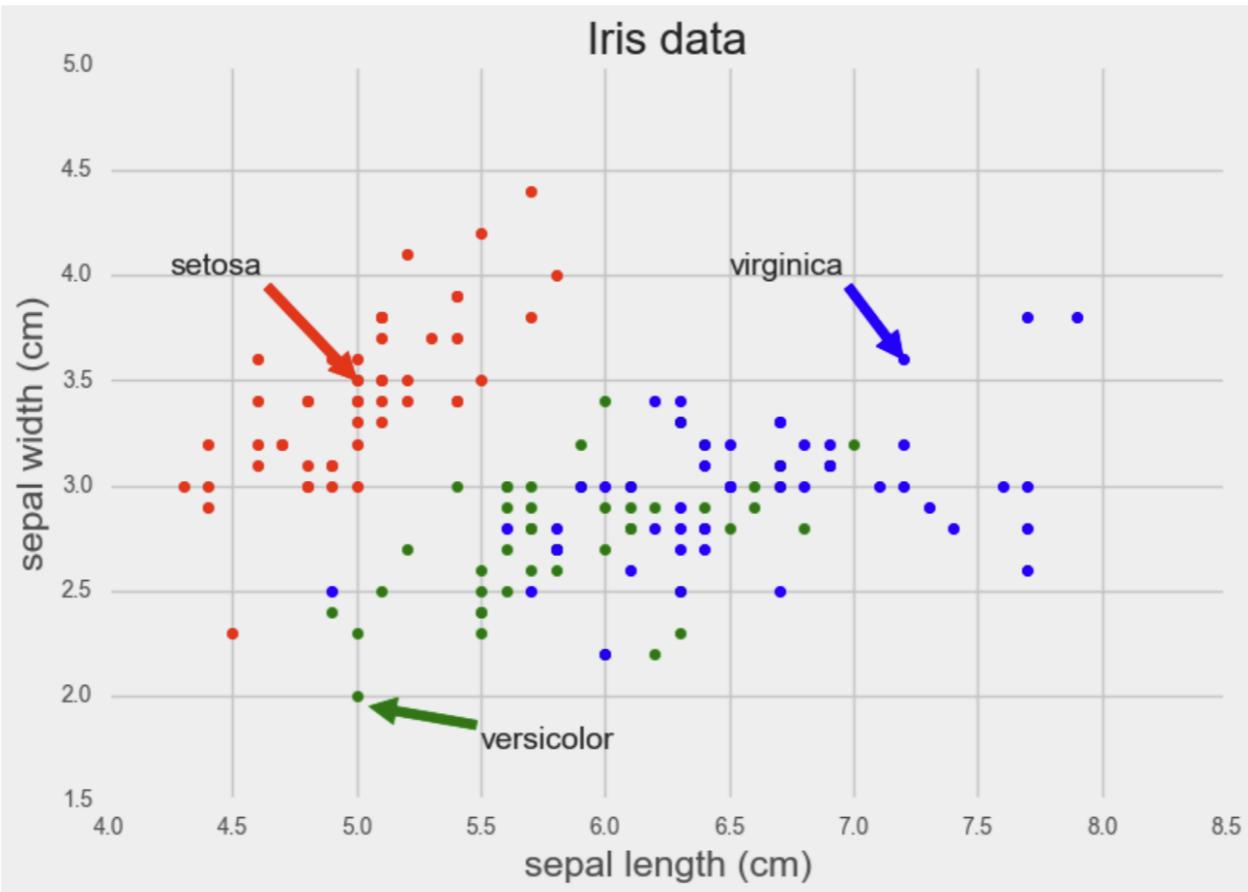
ggplot style

```
import matplotlib.pyplot as plt  
plt.style.use('ggplot')
```



fivethirtyeight style

```
import matplotlib.pyplot as plt  
plt.style.use('fivethirtyeight')
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION IN PYTHON