02. Styling and Theming

Bokeh Tutorial

(http://bokeh.pydata.org/)

In [1]:
```
from bokeh.io import output_notebook, show
from bokeh.plotting import figure
```

In [2]:
```
output_notebook()
```

(https://bokeh.pydata.org)

BokehJS 0.12.14 successfully loaded.

Colors and Properties

Colors

There are many places where you may need to specify colors. Bokeh can accept colors in a variety of different ways:

- any of the 147 named CSS colors (http://www.w3schools.com/cssref/css_colornames.asp), e.g `'green'`, `'indigo'`
- an RGB(A) hex value, e.g., `'#FF0000'`, `'#44444444'`
- a 3-tuple of integers *(r,g,b)* between 0 and 255
- a 4-tuple of *(r,g,b,a)* where *r*, *g*, *b* are integers between 0 and 255 and *a* is a floating point value between 0 and 1

Properties

Regardless of what API (`models`, `plotting`, or `charts`) is used to create a Bokeh plot, styling the visual aspects of the plot can always be accomplished by setting attributes on the Bokeh model objects that comprise the resulting plot. Visual properties come in three kinds: line, fill, and text properties. For full information with code and examples see the Styling Visual Properties (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html) section of the user guide.

Line Properties

Set the visual appearance of lines. The most common are `line_color`, `line_alpha`, `line_width` and `line_dash`.

Fill Properties

Set the visual appearance of filled areas: `fill_color` and `fill_alpha`.

Text Properties

Set the visual appearance of lines of text. The most common are `text_font`, `text_font_size`, `text_color`, and `text_alpha`.

Sometimes a prefix is used with property names, e.g. to distinguish between different line properties on the same object, or to give a more meaningful name. For example, to set the line width of the plot outline, you would say `myplot.outline_line_width = 2` .

Plots

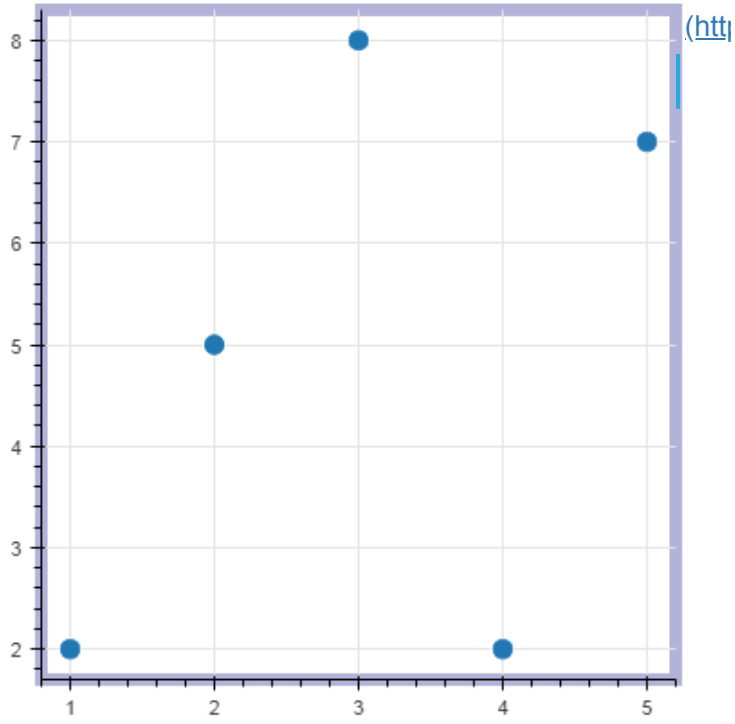
Many top-level attributes of plots (outline, border, etc.) can be configured. See the Plots (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#plots) section of the styling guide for full information.

Here is an example that tweaks the plot outline:

In [3]:
```
# create a new plot with a title
p = figure(plot_width=400, plot_height=400)
p.outline_line_width = 7
p.outline_line_alpha = 0.3
p.outline_line_color = "navy"

p.circle([1,2,3,4,5], [2,5,8,2,7], size=10)

show(p)
```



(http

In [4]:
```
# EXERCISE Create a plot of your own and customize several plot-level properties
```

Glyphs

It's also possible to style the visual properties of glyphs. When using `bokeh.plotting` this is often done when calling the glyph methods:

```
p.circle(line_color="red", fill_alpha=0.2, ...)
```

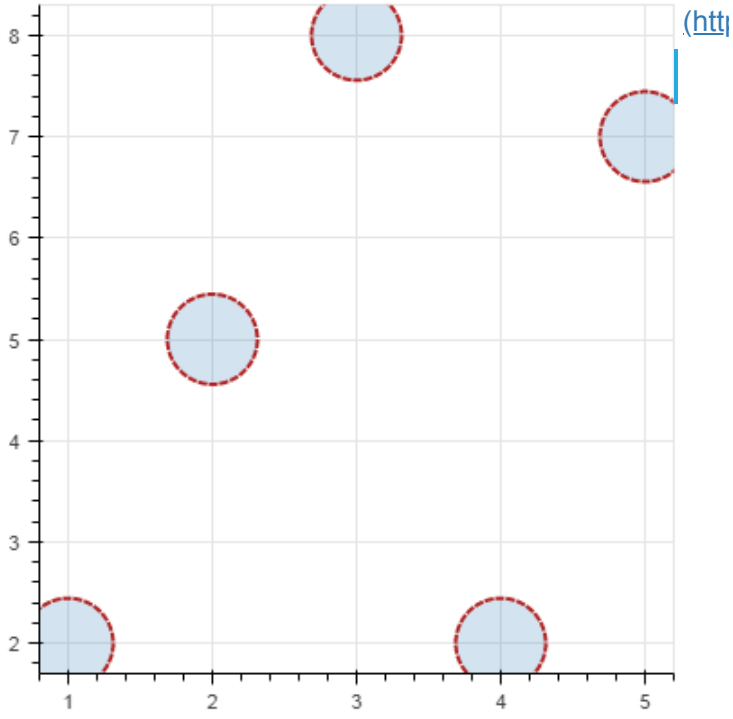
But it is also possible to set these properties directly on glyph objects. Glyph objects are found on `GlyphRenderer` objects, which are returned by the `Plot.add_glyph` and `bokeh.plotting` glyph methods like `circle`, `rect`, etc. Let's look at an example:

In [4]:
```
p = figure(plot_width=400, plot_height=400)

# keep a reference to the returned GlyphRenderer
r = p.circle([1,2,3,4,5], [2,5,8,2,7])

r.glyph.size = 50
r.glyph.fill_alpha = 0.2
r.glyph.line_color = "firebrick"
r.glyph.line_dash = [5, 1]
r.glyph.line_width = 2

show(p)
```



(htt

Selection and non-selection visuals

You can also control how glyphs look when there are selections involved. The set of "selected" points is displayed according to the optional `.selection_glyph` property of a `GlyphRenderer`:

```
r.selection_glyph = Circle(fill_alpha=1, fill_color="firebrick", line_color=None)
```

When there is a non-empty selection, the set of "unselected: points is displayed according to the optional `.nonselection_glyph` property of a `GlyphRenderer`:

```
r.nonselection_glyph = Circle(fill_alpha=0.2, fill_color="grey", line_color=None)
```

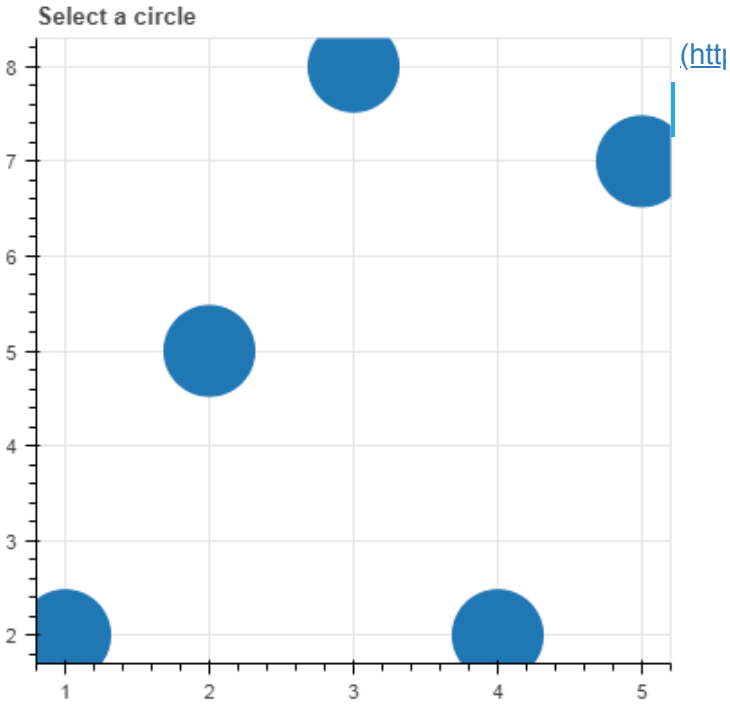
When using the `bokeh.plotting` interface, it is easier to pass these visual properties to the glyph methods as shown below. The glyph method will create the selection or nonselection glyphs and attach them to the renderer for you.

In [5]:
```
p = figure(plot_width=400, plot_height=400, tools="tap", title="Select a circle")
renderer = p.circle([1, 2, 3, 4, 5], [2, 5, 8, 2, 7], size=50,

                    # set visual properties for selected glyphs
                    selection_color="firebrick",

                    # set visual properties for non-selected glyphs
                    nonselection_fill_alpha=0.2,
                    nonselection_fill_color="grey",
                    nonselection_line_color="firebrick",
                    nonselection_line_alpha=1.0)

show(p)
```



It is also possible to specify the visual appearance of glyphs when they are "inspected", e.g. by a hover tool. This is accomplished by setting an optional `hover_glyph` on the glyph renderer:

```
r.hover_glyph = Circle(fill_alpha=1, fill_color="firebrick", line_color=None)
```

Or if using `bokeh.plotting` glyph methods, by passing `hover_fill_alpha`, etc. to the glyph method. Lets look at an example that works together with a `HoverTool` configured for "hline" hit-testing.

In [ ]:
```
from bokeh.models.tools import HoverTool
from bokeh.sampledata.glucose import data

subset = data.loc['2010-10-06']

x, y = subset.index.to_series(), subset['glucose']

# Basic plot setup
p = figure(width=600, height=300, x_axis_type="datetime", title='Hover over points')

p.line(x, y, line_dash="4 4", line_width=1, color='gray')

cr = p.circle(x, y, size=20,
              fill_color="grey", hover_fill_color="firebrick",
              fill_alpha=0.05, hover_alpha=0.3,
              line_color=None, hover_line_color="white")

p.add_tools(HoverTool(tooltips=None, renderers=[cr], mode='hline'))

show(p)
```

In [8]:
```
# EXERCISE: experiment with standard, selected, hover glyph visual properties
```

Axes

Axes (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#axes)

To style axes, you first must get ahold of `Axis` objects. The simplest way is to use some convenience methods on `Plot` : axis (http://bokeh.pydata.org/en/latest/docs/reference/plotting.html#bokeh.plotting.Figure.axis), xaxis (http://bokeh.pydata.org/en/latest/docs/reference/plotting.html#bokeh.plotting.Figure.xaxis), and yaxis (http://bokeh.pydata.org/en/latest/docs/reference/plotting.html#bokeh.plotting.Figure.yaxis). These methods return lists of axis objects:

```
>>> p.xaxis
[<bokeh.models.axes.LinearAxis at 0x106fa2390>]
```

However, you can set properties on all the elements of the list as if it was a single object:

```
p.xaxis.axis_label = "Temperature"
p.axis.major_label_text_color = "orange"
```

These are referred to as "splattable" lists, and tab completion works on them as well.

In [9]:
```
# EXERCISE Try out tab completion. Type p.xaxis.<press tab key> to see a list of attributes
```

- **axis**
  - line properties (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#line-properties)
- **axis_label**
  - text properties (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#text-properties)
  - `axis_label_standoff`
- **major_label**
  - text properties (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#text-properties)

    - orientation
- **major_tick**
    - line_properties (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#line-properties)
    - major_tick_in
    - major_tick_out
- **minor_tick**
    - line_properties (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#line-properties)
    - minor_tick_in
    - minor_tick_out

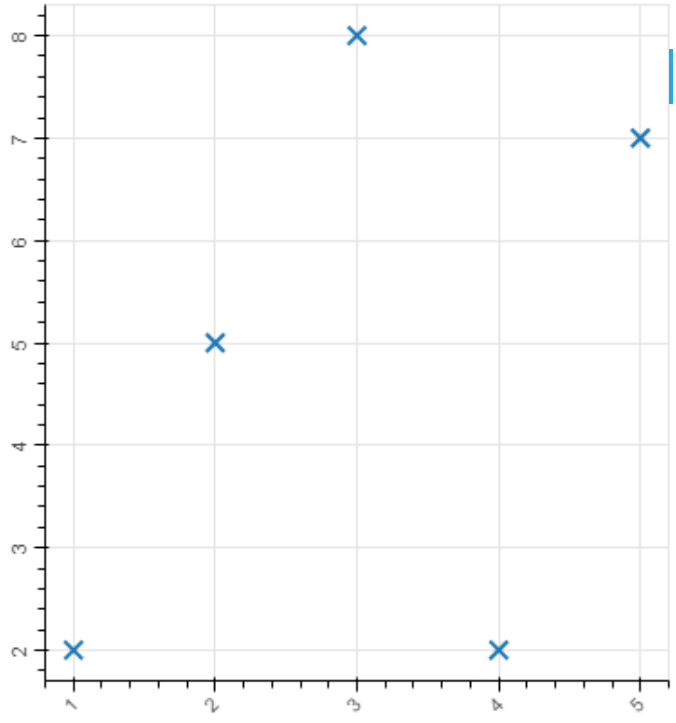
As a simple first example, let's change the orientation of the major tick labels on both axes of a plot:

In [9]:
```
from math import pi

p = figure(plot_width=400, plot_height=400)
p.x([1,2,3,4,5], [2,5,8,2,7], size=10, line_width=2)

p.xaxis.major_label_orientation = pi/4
p.yaxis.major_label_orientation = "vertical"

show(p)
```



(http

The next example shows customizations on several of the different Axis properties at once:

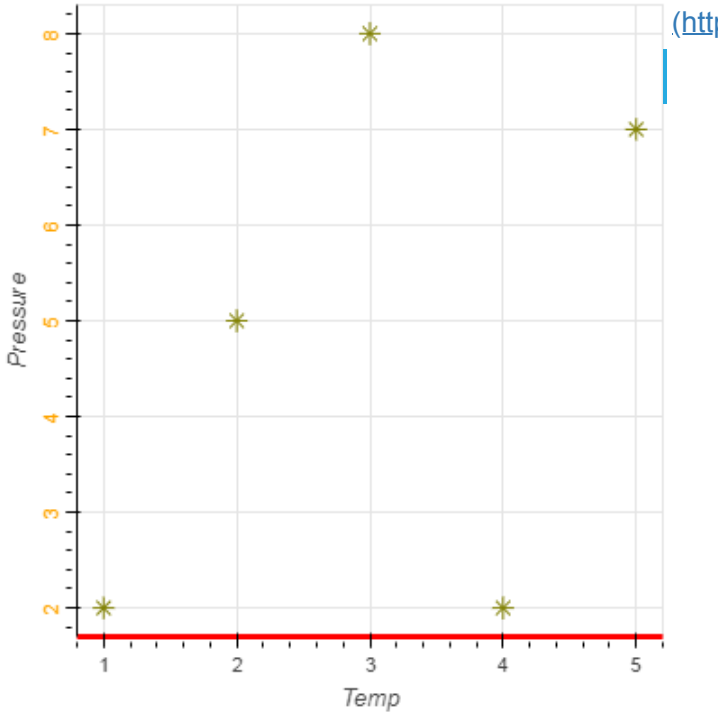
In [10]:
```
p = figure(plot_width=400, plot_height=400)
p.asterisk([1,2,3,4,5], [2,5,8,2,7], size=12, color="olive")

# change just some things about the x-axes
p.xaxis.axis_label = "Temp"
p.xaxis.axis_line_width = 3
p.xaxis.axis_line_color = "red"

# change just some things about the y-axes
p.yaxis.axis_label = "Pressure"
p.yaxis.major_label_text_color = "orange"
p.yaxis.major_label_orientation = "vertical"

# change things on all axes
p.axis.minor_tick_in = -3
p.axis.minor_tick_out = 6

show(p)
```



(http

In [12]:
```
# EXERCISE Create a plot of your own and customize several axis properties
```

There are further customizations possible. See the User Guide (http://bokeh.pydata.org/en/latest/docs/user_guide.html) for more information on topics such as tick label formatting (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#tick-label-formats) or limiting axis bounds (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#bounds).

Grids

Grids (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#grids)

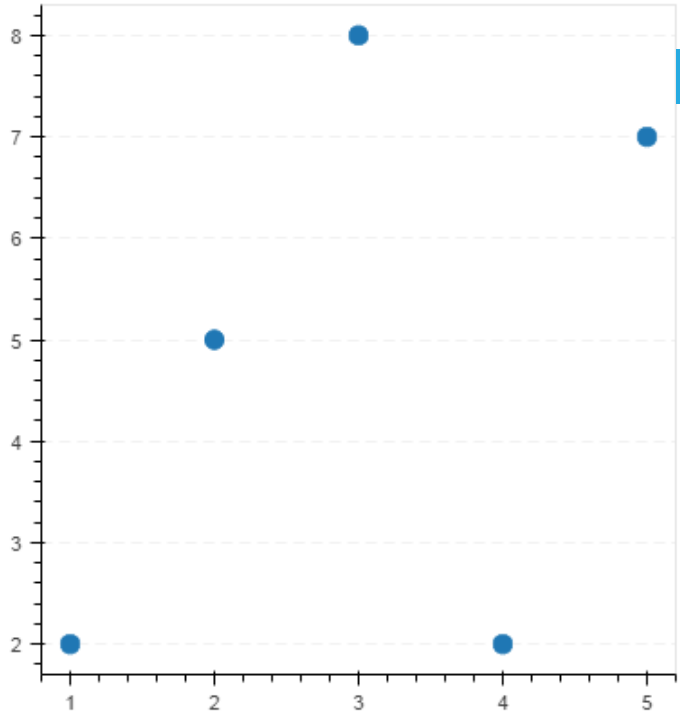
- **grid** line properties (http://bokeh.pydata.org/en/latest/docs/user_guide/styling.html#line-properties)
- **band** fill properties ()

In [11]:
```
p = figure(plot_width=400, plot_height=400)
p.circle([1,2,3,4,5], [2,5,8,2,7], size=10)

# change just some things about the x-grid
p.xgrid.grid_line_color = None

# change just some things about the y-grid
p.ygrid.grid_line_alpha = 0.5
p.ygrid.grid_line_dash = [6, 4]

show(p)
```



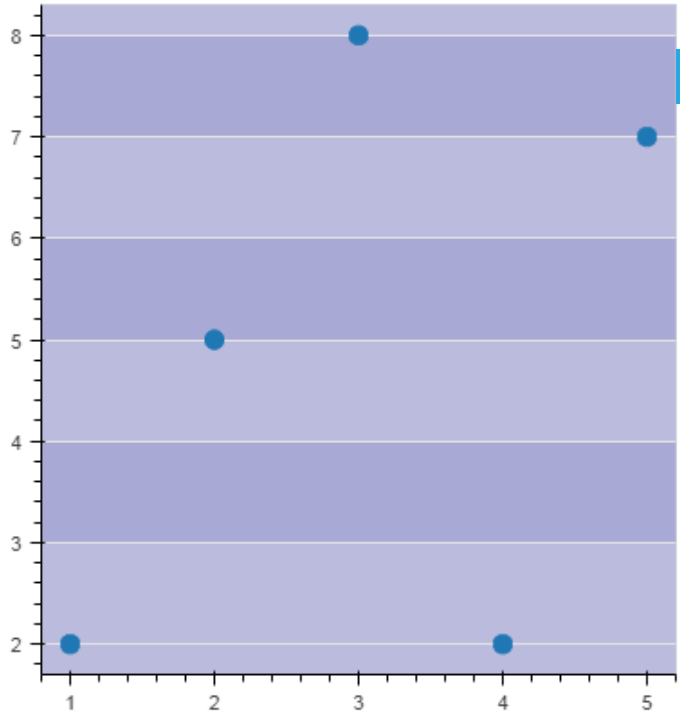
(htt

In [12]:
```
p = figure(plot_width=400, plot_height=400)
p.circle([1,2,3,4,5], [2,5,8,2,7], size=10)

# change just some things about the x-grid
p.xgrid.grid_line_color = None

# change just some things about the y-grid
p.ygrid.band_fill_alpha = 0.1
p.ygrid.band_fill_color = "navy"

show(p)
```



(htt

In [13]:
```
# EXERCISE Create a plot of your own and customize several grid properties

```