03. Adding Annotations

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.

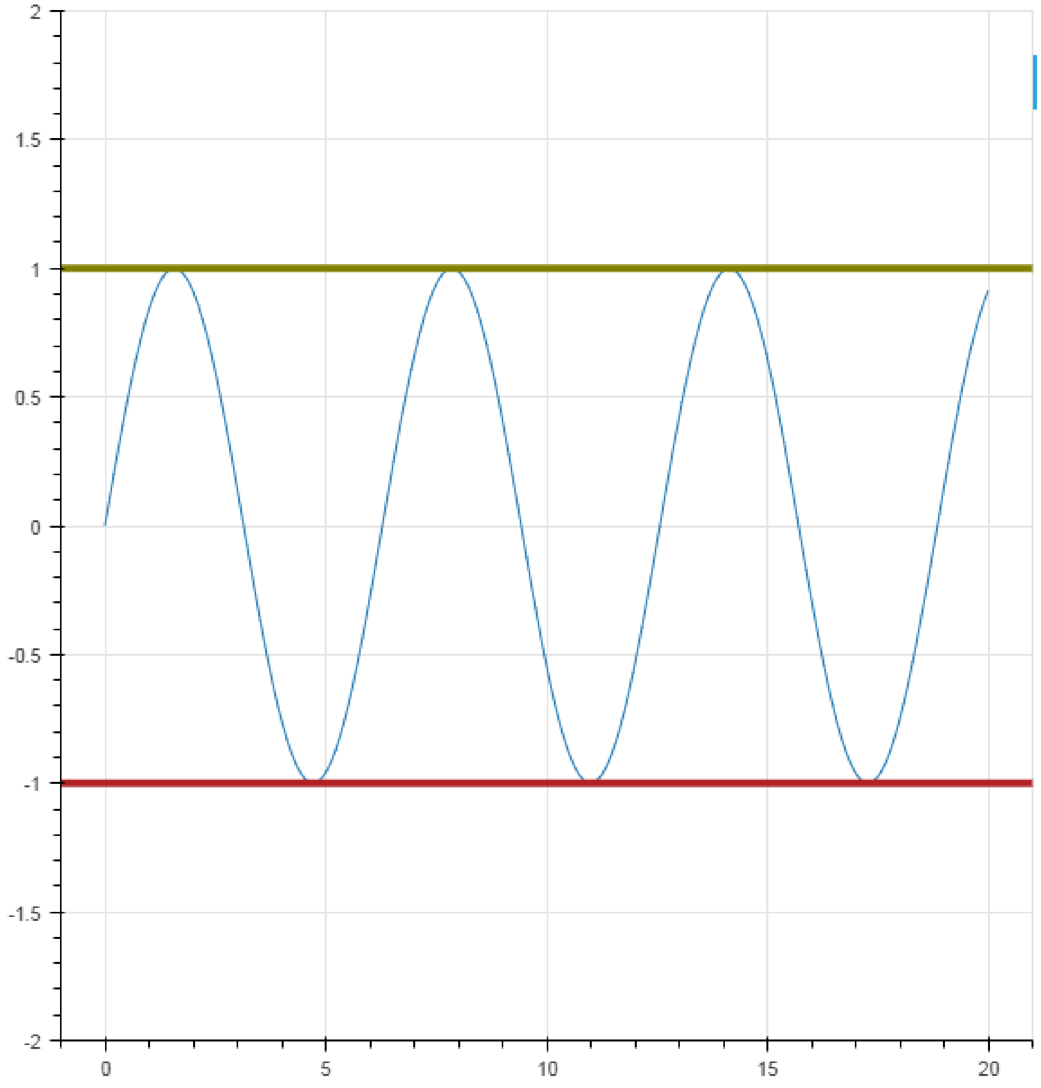
Overview

Sometimes we want to add visual cues (boundary lines, shaded regions, labels and arrows, etc.) to our plots to call out some feature or other. Bokeh has several annotation types available for uses like this. Typically to add annotations we create the "low level" annotation object directly, and add it to our `Plot` , `Figure` or `Chart` using `add_layout` . Let's take a look at some specific examples.

Spans

`Spans` are "infinite" vertical or horizonal lines. When creating them, you specify the `dimension` that should be spanned (i.e., `width` or `height`), any visual line properties for the appearance, and the location along the dimension where the line should be drawn. Let's look at an example that adds two horizontal spans to a simple plot:

In [3]:
```
import numpy as np
from bokeh.models.annotations import Span

x = np.linspace(0, 20, 200)
y = np.sin(x)

p = figure(y_range=(-2, 2))
p.line(x, y)

upper = Span(location=1, dimension='width', line_color='olive', line_width=4)
p.add_layout(upper)

lower = Span(location=-1, dimension='width', line_color='firebrick', line_width=4)
p.add_layout(lower)

show(p)
```



(htt

In [4]:
```
# Exercise:
```

Box Annotations

Sometimes you might want to call out some region of the plot by drawing a shaded box. This can be done with the `BoxAnnotation`, which is configured with the coordinate properties:

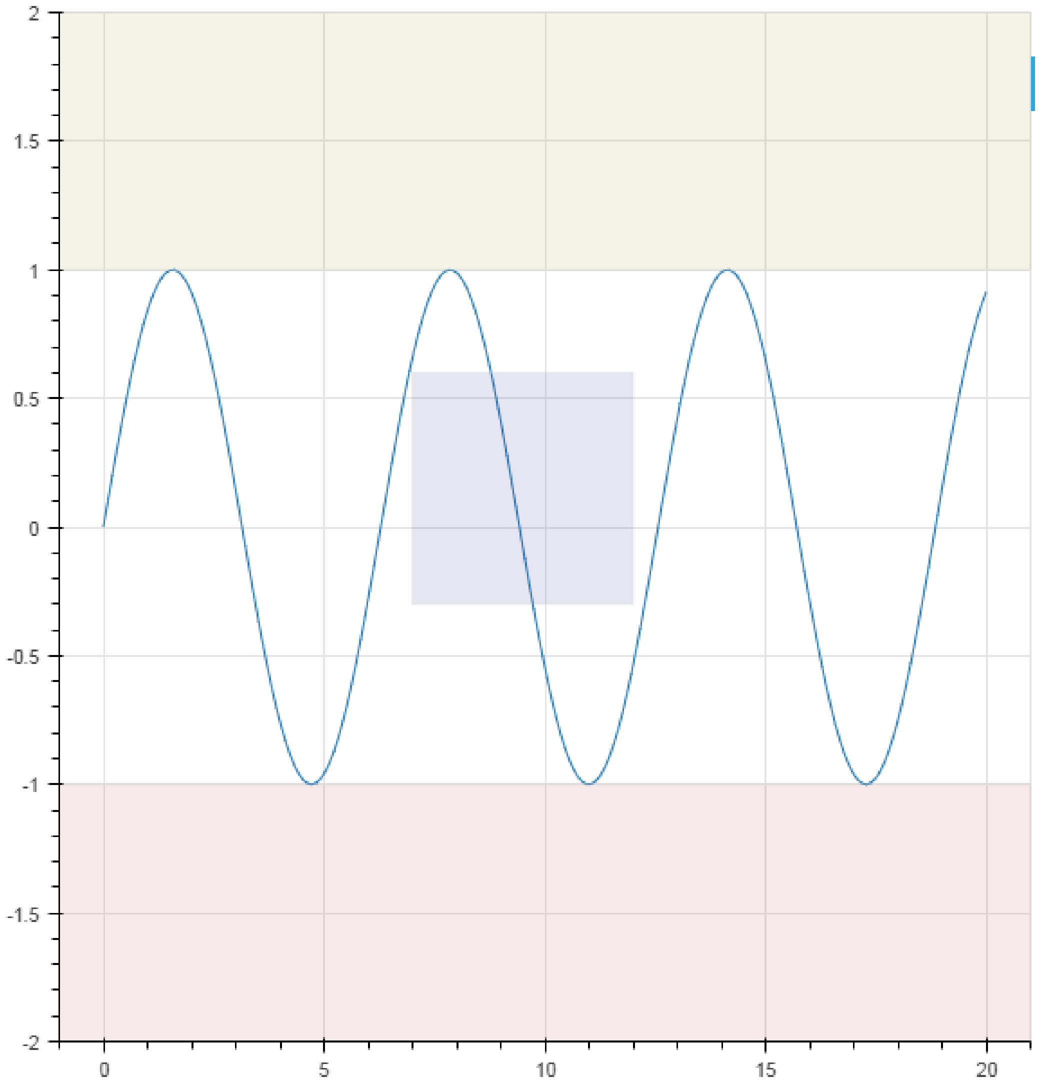
- `top`
- `left`
- `bottom`
- `right`

as well as any visual line or fill properties to control the appearance.

"Infinite" boxes can be made by leaving any of the coordinates unspecified. E.g., if `top` is not given, the box will always extend to the top of the plot area, regardless of any panning or zooming that happens.

Let's take a look at an example that adds a few shaded boxes to a plot:

In [4]:
```
import numpy as np
from bokeh.models.annotations import BoxAnnotation

x = np.linspace(0, 20, 200)
y = np.sin(x)

p = figure(y_range=(-2, 2))
p.line(x, y)

# region that always fills the top of the plot
upper = BoxAnnotation(bottom=1, fill_alpha=0.1, fill_color='olive')
p.add_layout(upper)

# region that always fills the bottom of the plot
lower = BoxAnnotation(top=-1, fill_alpha=0.1, fill_color='firebrick')
p.add_layout(lower)

# a finite region
center = BoxAnnotation(top=0.6, bottom=-0.3, left=7, right=12, fill_alpha=0.1, fill_color='n
p.add_layout(center)

show(p)
```



(htt

In [6]:
```
# Exercise:
```

Label

The `Label` annotation allows you to easily attach single text labels to plots. The position and text to display are configured as `x`, `y`, and `text`:

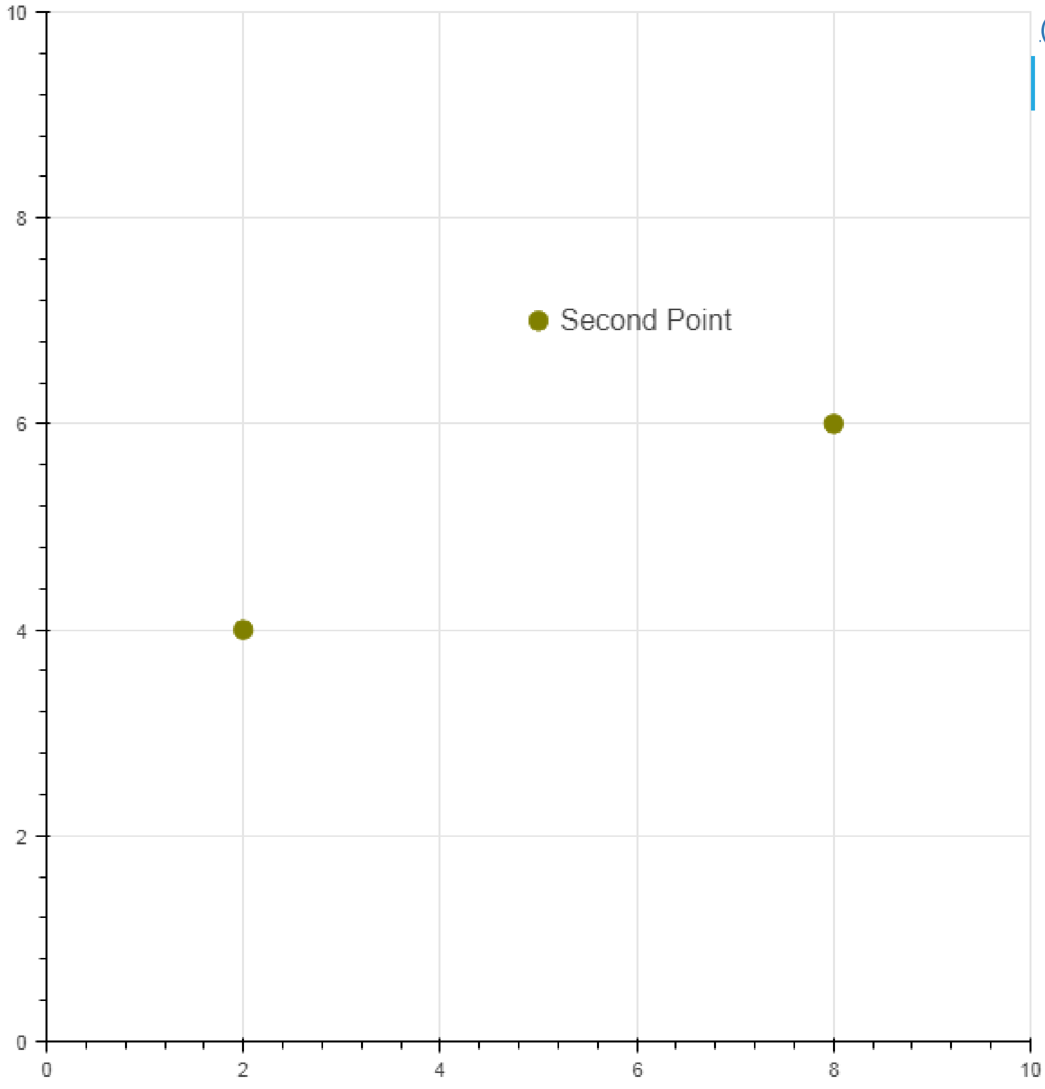
```
Label(x=10, y=5, text="Some Label")
```

By default the units are in "data space" but `x_units` and `y_units` maybe set to `"screen"` to position the label relative to the canvas. Labels can also accept `x_offset` and `y_offset` to offset the final position from `x` and `y` by a given screen space distance.

`Label` objects also have standard text, line (`border_line`) and fill (`background_fill`) properties. The line and fill properties apply to a bounding box around the text:

```
Label(x=10, y=5, text="Some Label", text_font_size="12pt",
      border_line_color="red", background_fill_color="blue")
```

In [5]:
```
from bokeh.models.annotations import Label
from bokeh.plotting import figure

p = figure(x_range=(0,10), y_range=(0,10))
p.circle([2, 5, 8], [4, 7, 6], color="olive", size=10)

label = Label(x=5, y=7, x_offset=12, text="Second Point", text_baseline="middle")
p.add_layout(label)

show(p)
```



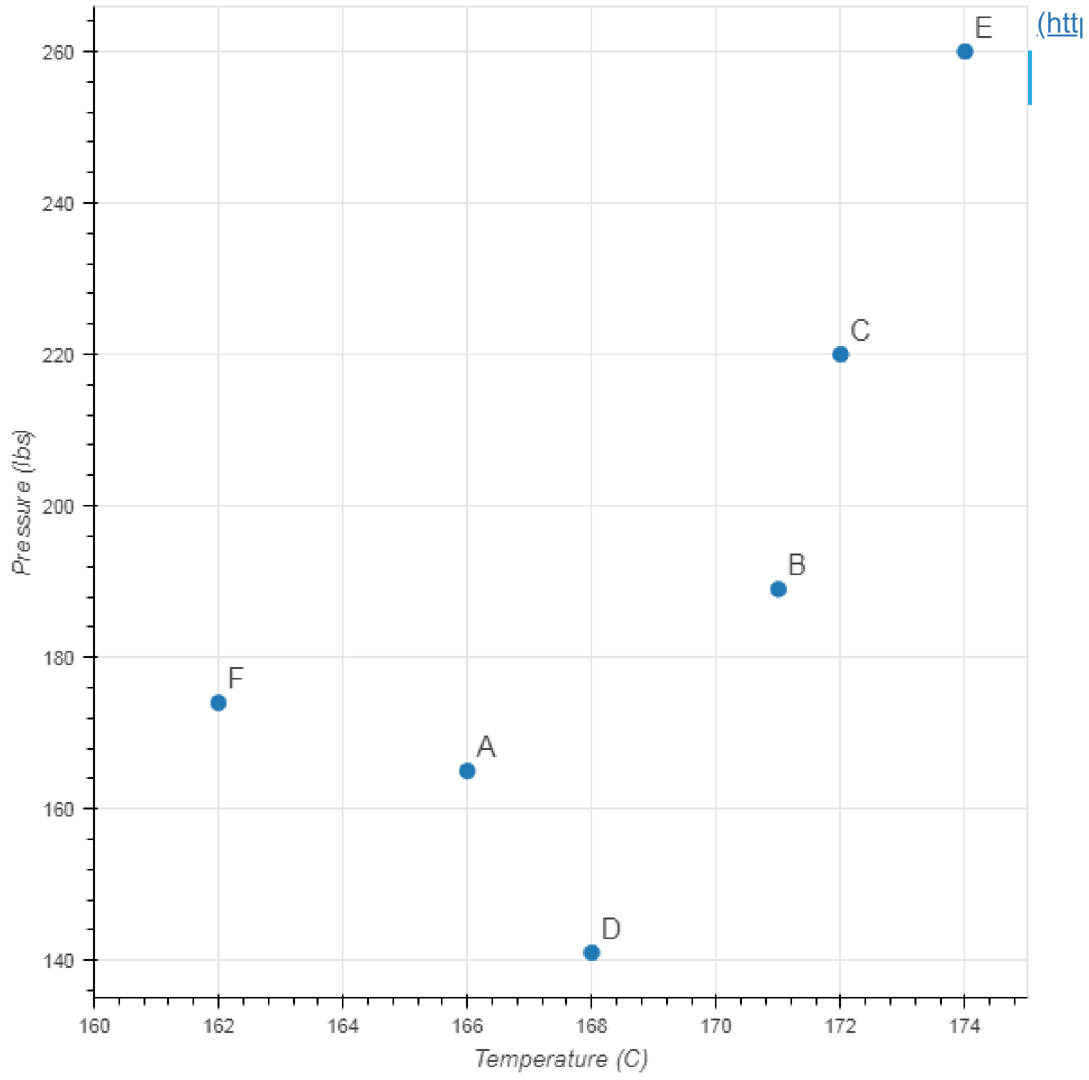
(htt

In [8]:
```
# EXERCISE: experiment with Label
```

LabelSet

The `LabelSet` annotation allows you to create many labels at once, for instance if you want to label an entire set of scatter markers. They are similar to `Label`, but they can also accept a `ColumnDataSource` as the `source` property, and then `x` and `y` may refer to columns in the data source, e.g. `x="col2"` (but may also still be fixed values, e.g. `x=10`).

In [6]:
```python
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, LabelSet


source = ColumnDataSource(data=dict(
    temp=[166, 171, 172, 168, 174, 162],
    pressure=[165, 189, 220, 141, 260, 174],
    names=['A', 'B', 'C', 'D', 'E', 'F']))

p = figure(x_range=(160, 175))
p.scatter(x='temp', y='pressure', size=8, source=source)
p.xaxis.axis_label = 'Temperature (C)'
p.yaxis.axis_label = 'Pressure (lbs)'

labels = LabelSet(x='temp', y='pressure', text='names', level='glyph',
                  x_offset=5, y_offset=5, source=source, render_mode='canvas')


p.add_layout(labels)

show(p)
```



In [10]:
```python
# EXERCISE: experiment with LabelSet
```

Arrows

The `Arrow` annotation allows you to "point" at different things on your plot, and can be especially useful in conjuction with labels.

For example, to create an arrow that points from `(0,0)` to `(1,1)` :

```
p.add_layout(Arrow(x_start=0, y_start=0, x_end=1, y_end=0))
```

This arrow will have the default `OpenHead` (http://bokeh.pydata.org/en/latest/docs/reference/models/arrow_heads.html#bokeh.models.arrow_heads.OpenHea arrow head at the end of the arrow. Other kinds of arrow heads include `NormalHead` (http://bokeh.pydata.org/en/latest/docs/reference/models/arrow_heads.html#bokeh.models.arrow_heads.NormalHe and `VeeHead` (http://bokeh.pydata.org/en/latest/docs/reference/models/arrow_heads.html#bokeh.models.arrow_heads.VeeHead) The arrow head type can be controlled by setting the `start` and `end` properties of `Arrow` objects:

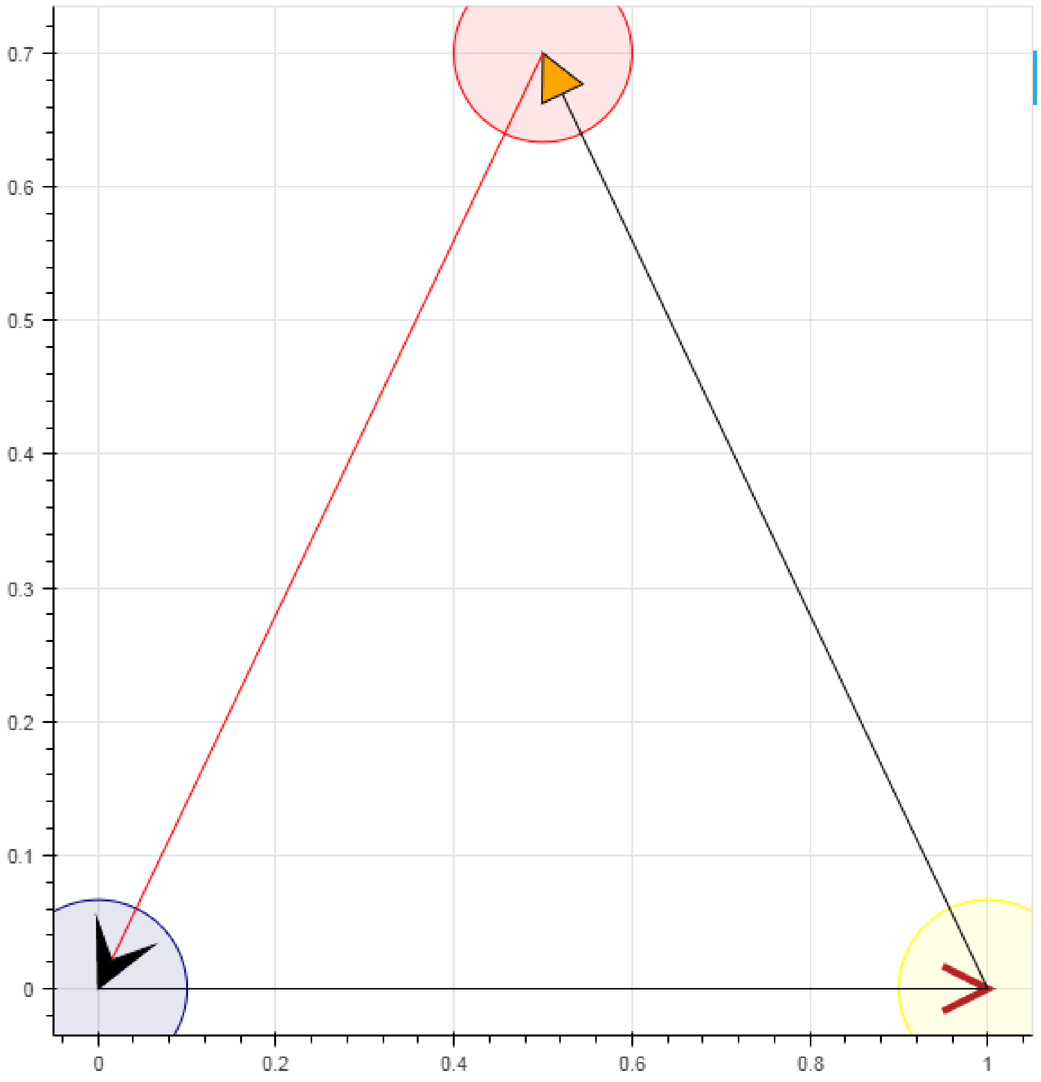
```
p.add_layout(Arrow(start=OpenHead(), end=VeeHead(),
                   x_start=0, y_start=0, x_end=1, y_end=0))
```

This will create a double-ended arrow with an "open" head at the start, and a "vee" head at the end. Arrowheads have the standard set of line and fill properties to control their appearance. As an example

```
OpenHead(line_color="firebrick", line_width=4)
```

The code and plot below shows several of these configurations together.

In [7]:
```
from bokeh.models.annotations import Arrow
from bokeh.models.arrow_heads import OpenHead, NormalHead, VeeHead

p = figure(plot_width=600, plot_height=600)

p.circle(x=[0, 1, 0.5], y=[0, 0, 0.7], radius=0.1,
         color=["navy", "yellow", "red"], fill_alpha=0.1)

p.add_layout(Arrow(end=OpenHead(line_color="firebrick", line_width=4),
                   x_start=0, y_start=0, x_end=1, y_end=0))

p.add_layout(Arrow(end=NormalHead(fill_color="orange"),
                   x_start=1, y_start=0, x_end=0.5, y_end=0.7))

p.add_layout(Arrow(end=VeeHead(size=35), line_color="red",
                   x_start=0.5, y_start=0.7, x_end=0, y_end=0))

show(p)
```



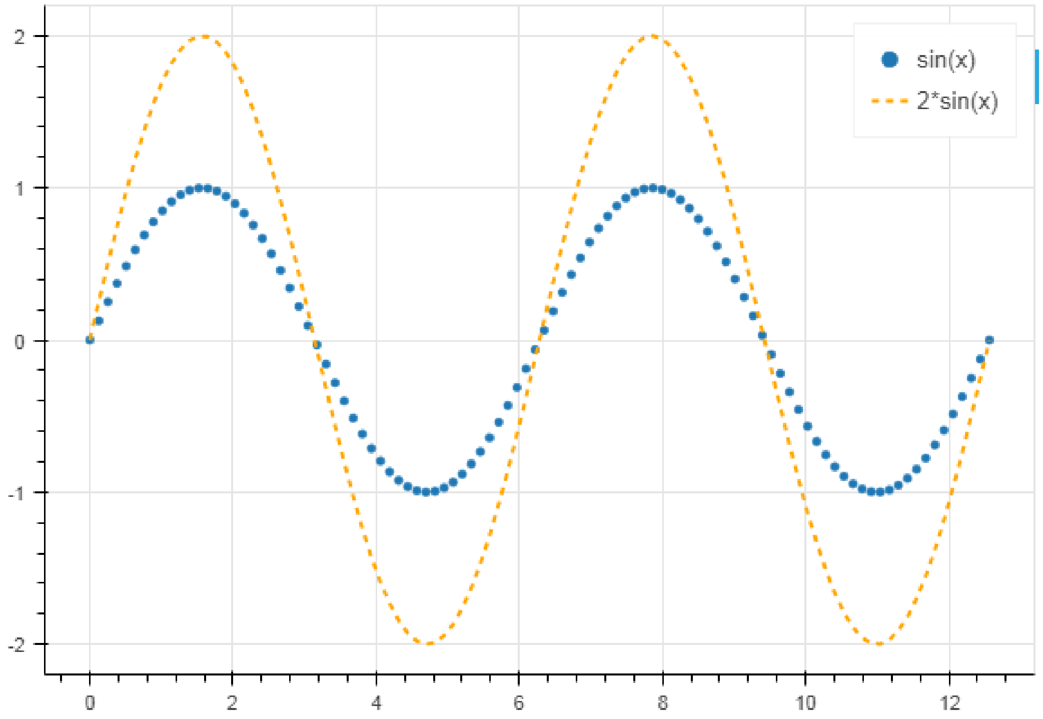
(htt

In [12]:
```
# EXERCISE: experiment with arrows and arrow heads
```

Legends

Simple Legends

In [8]:
```
import numpy as np

x = np.linspace(0, 4*np.pi, 100)
y = np.sin(x)

p = figure(height=400)

p.circle(x, y, legend="sin(x)")
p.line(x, 2*y, legend="2*sin(x)", line_dash=[4, 4], line_color="orange", line_width=2)

show(p)
```



(htt

Compound legends

Sometimes, two (or more) different glyphs are used with a single data source. In this case, you can make compound legends by specifying the same legend argument to multiple glyph methods when creating a plot, as in:

```
p.circle(x, y, legend="sin(x)")
p.line(x, y, legend="sin(x)", line_dash=[4, 4], line_color="orange", line_width=2)
```

In [14]:
```
# EXERCISE:
# (1) Try making a compound legend
# (2) Try moving the legend using p.legend.location.  Possible values are listed in bokeh.co
```

Legend Orientation and Positioning

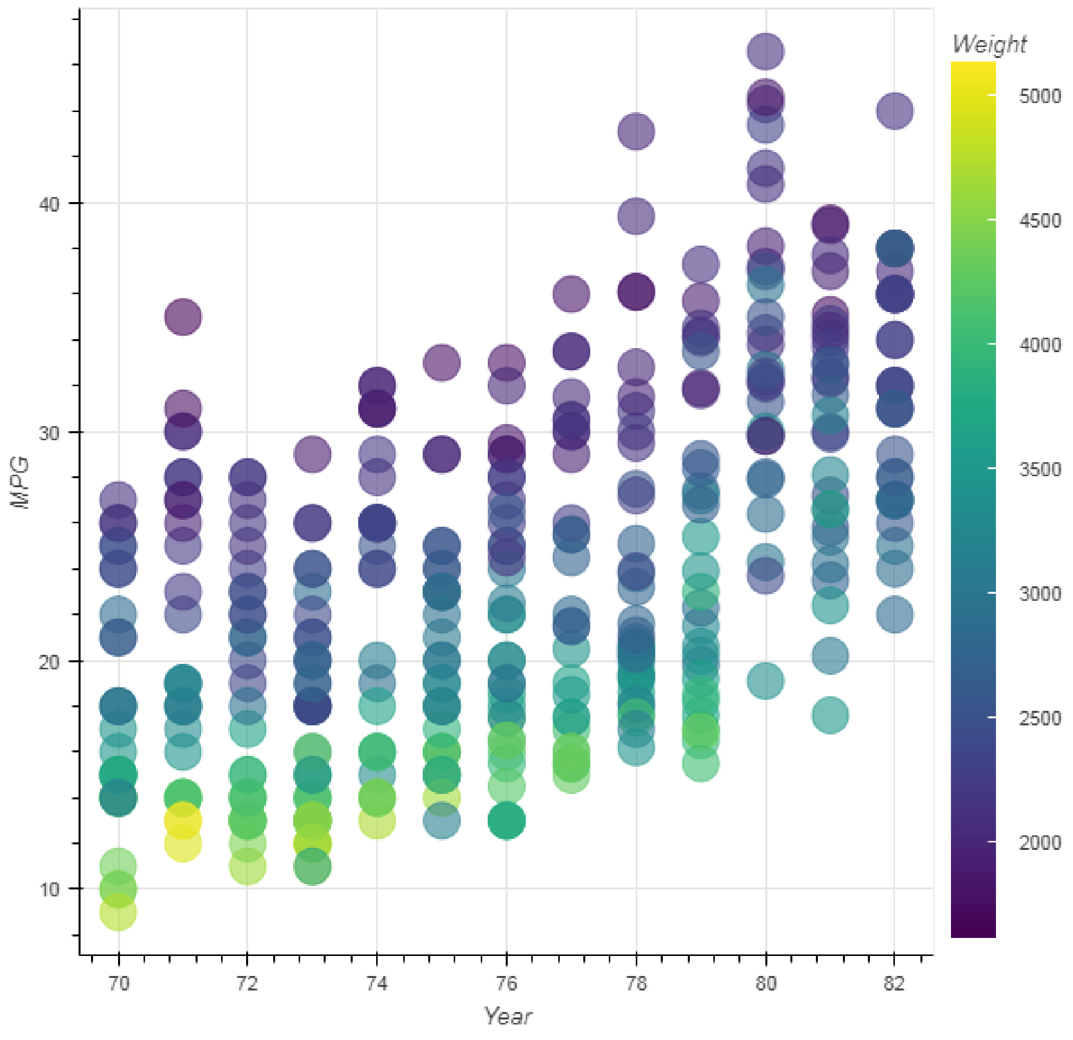
In [15]:
```
# Exercise
```

Color bars

In [9]:
```
from bokeh.sampledata.autompg import autompg
from bokeh.models import LinearColorMapper, ColorBar
from bokeh.palettes import Viridis256

source = ColumnDataSource(autompg)
color_mapper = LinearColorMapper(palette=Viridis256, low=autompg.weight.min(), high=autompg.

p = figure(x_axis_label='Year', y_axis_label='MPG', tools='', toolbar_location='above')
p.circle(x='yr', y='mpg', color={'field': 'weight', 'transform': color_mapper}, size=20, alp

color_bar = ColorBar(color_mapper=color_mapper, label_standoff=12, location=(0,0), title='We
p.add_layout(color_bar, 'right')

show(p)
```

```
/srv/conda/lib/python3.6/site-packages/bokeh/core/json_encoder.py:80: FutureWarning: Conve
rsion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In
future, it will be treated as `np.float64 == np.dtype(float).type`.
  elif np.issubdtype(type(obj), np.float):
```

(ht


Weight
MPG
Year