Bokeh Tutorial

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

04. Data Sources and Transformations

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

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

(https://bokeh.pydata.org)

Overview

We've seen how Bokeh can work well with Python lists, NumPy arrays, Pandas series, etc. At lower levels, these inputs are converted to a Bokeh `ColumnDataSource`. This data type is the central data source object used throughout Bokeh. Although Bokeh often creates them for us transparently, there are times when it is useful to create them explicitly.

In later sections we will see features like hover tooltips, computed transforms, and CustomJS interactions that make use of the `ColumnDataSource`, so let's take a quick look now.

Creating with Python Dicts

The `ColumnDataSource` can be imported from `bokeh.models`:

In [3]:
```
from bokeh.models import ColumnDataSource
```

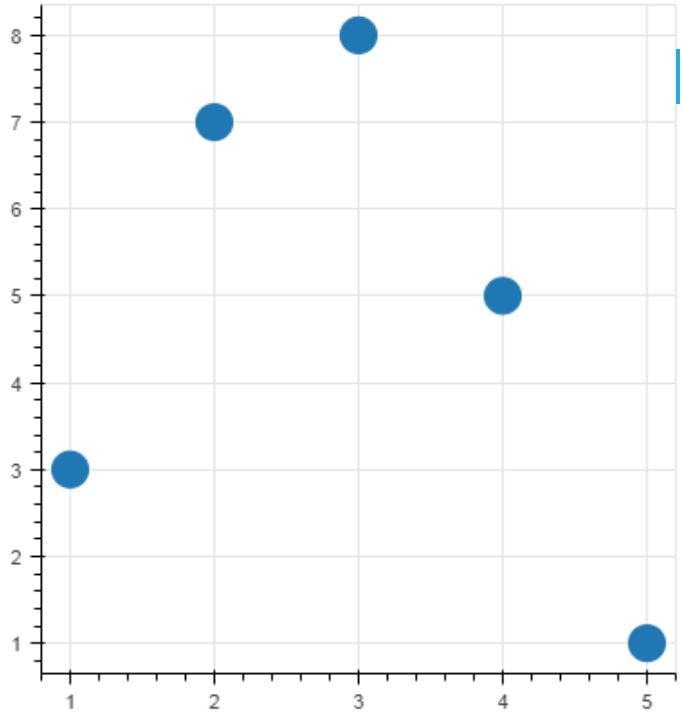
The `ColumnDataSource` is a mapping of column names (strings) to sequences of values. Here is a simple example. The mapping is provided by passing a Python `dict` with string keys and simple Python lists as values. The values could also be NumPy arrays, or Pandas sequences.

NOTE: ALL the columns in a `ColumnDataSource` must always be the SAME length.

In [4]:
```
source = ColumnDataSource(data={
    'x' : [1, 2, 3, 4, 5],
    'y' : [3, 7, 8, 5, 1],
})
```

Up until now we have called functions like `p.circle` by passing in literal lists or arrays of data directly, when we do this, Bokeh creates a `ColumnDataSource` for us, automatically. But it is possible to specify a `ColumnDataSource` explicitly by passing it as the `source` argument to a glyph method. Whenever we do this, if we want a property (like `"x"` or `"y"` or `"fill_color"`) to have a sequence of values, we pass the ***name of the column*** that we would like to use for a property:

In [5]:
```
p = figure(plot_width=400, plot_height=400)
p.circle('x', 'y', size=20, source=source)
show(p)
```



(htt

In [6]:
```
# Exercise: create a column data source with NumPy arrays as column values and plot it
```

Creating with Pandas DataFrames

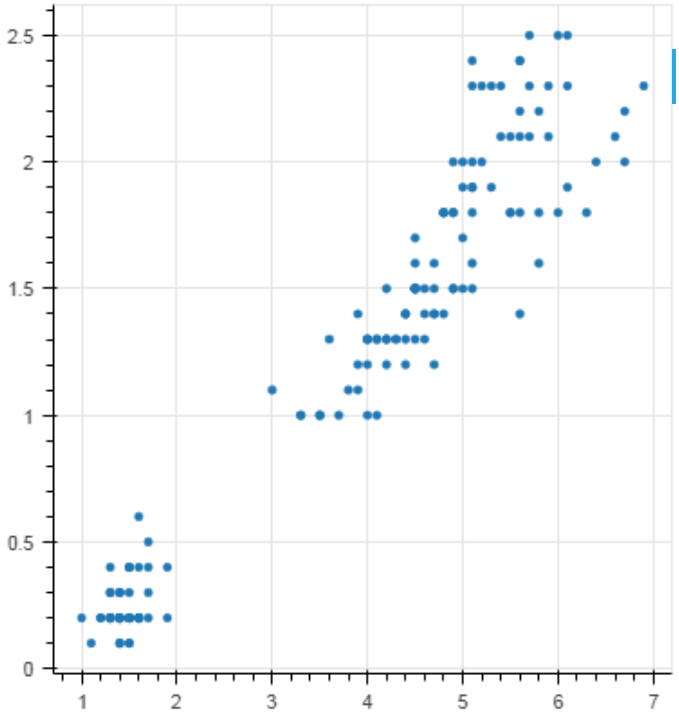
It's also simple to create `ColumnDataSource` objects directly from Pandas data frames. To do this, just pass the data frame to `ColumnDataSource` when you create it:

In [6]:
```
from bokeh.sampledata.iris import flowers as df

source = ColumnDataSource(df)
```

Now we can use it as we did above by passing the column names to glhph methods:

In [7]:
```
p = figure(plot_width=400, plot_height=400)
p.circle('petal_length', 'petal_width', source=source)
show(p)
```



(htt

In [8]:
```
# Exercise: create a column data source with the autompg sample data frame and plot it

from bokeh.sampledata.autompg import autompg_clean as df
```

In [ ]:
```

```