Widget Basics

In this lecture we will continue to build off our understanding of **interact** and **interactive** to begin using full widgets!

What are widgets?

Widgets are eventful python objects that have a representation in the browser, often as a control like a slider, textbox, etc.

What can they be used for?

You can use widgets to build **interactive GUIs** for your notebooks.
You can also use widgets to **synchronize stateful and stateless information** between Python and JavaScript.

Using widgets

To use the widget framework, you need to import `ipywidgets`.

In [1]:

```
import ipywidgets as widgets
```

repr

Widgets have their own display `repr` which allows them to be displayed using IPython's display framework. Constructing and returning an `IntSlider` automatically displays the widget (as seen below). Widgets are displayed inside the output area below the code cell. Clearing cell output will also remove the widget.

In [2]:

```
widgets.IntSlider()
```

display()

You can also explicitly display the widget using `display(...)`.

In [3]:

```
from IPython.display import display
w = widgets.IntSlider()
display(w)
```

Multiple display() calls

If you display the same widget twice, the displayed instances in the front-end will remain in sync with each other. Try dragging the slider below and watch the slider above.

In [4]:

```
display(w)
```

Closing widgets

You can close a widget by calling its `close()` method.

In [5]:

```
display(w)
```

In [6]:

```
w.close()
```

Widget properties

All of the IPython widgets share a similar naming scheme. To read the value of a widget, you can query its `value` property.

In [7]:

```
w = widgets.IntSlider()
display(w)
```

In [8]:

```
w.value
```

Out[8]:

```
0
```

Similarly, to set a widget's value, you can set its `value` property.

In [9]:

```
w.value = 100
```

Keys

In addition to `value`, most widgets share `keys`, `description`, and `disabled`. To see the entire list of synchronized, stateful properties of any specific widget, you can query the `keys` property.

In [10]:

```
w.keys
```

Out[10]:

```
['_dom_classes',
 '_model_module',
 '_model_module_version',
 '_model_name',
 '_view_count',
 '_view_module',
 '_view_module_version',
 '_view_name',
 'continuous_update',
 'description',
 'disabled',
 'layout',
 'max',
 'min',
 'orientation',
 'readout',
 'readout_format',
 'step',
 'style',
 'value']
```

Shorthand for setting the initial values of widget properties

While creating a widget, you can set some or all of the initial values of that widget by defining them as keyword arguments in the widget's constructor (as seen below).

In [12]:

```
widgets.Text(value='Hello World!', disabled=True)
```

Linking two similar widgets

If you need to display the same value two different ways, you'll have to use two different widgets. Instead of attempting to manually synchronize the values of the two widgets, you can use the `link` or `jslink` function to link two properties together (the difference between these is discussed in Widget Events). Below, the values of two widgets are linked together.

In [13]:

```
a = widgets.FloatText()
b = widgets.FloatSlider()
display(a,b)

mylink = widgets.jslink((a, 'value'), (b, 'value'))
```

Unlinking widgets

Unlinking the widgets is simple. All you have to do is call `.unlink` on the link object. Try changing one of the widgets above after unlinking to see that they can be independently changed.

In [14]:

```
mylink.unlink()
```

Conclusion

You should now be beginning to have an understanding of how Widgets can interact with each other and how you can begin to specify widget details.