Python course materials

# Advanced Widget List

This notebook is an extension of **Widget List**, describing even more of the GUI widgets available!

import ipywidgets as widgets

### Output

The Output widget can capture and display stdout, stderr and [rich output generated by IPython](http://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html#module-IPython.display). After the widget is created, direct output to it using a context manager.

out = widgets.Output()  
out

You can print text to the output area as shown below.

with out:  
 for i in range(10):  
 print(i, 'Hello world!')

Rich material can also be directed to the output area. Anything which displays nicely in a Jupyter notebook will also display well in the Output widget.

from IPython.display import YouTubeVideo  
with out:  
 display(YouTubeVideo('eWzY2nGfkXk'))

### Play (Animation) widget

The Play widget is useful to perform animations by iterating on a sequence of integers with a certain speed. The value of the slider below is linked to the player.

play = widgets.Play(  
 # interval=10,  
 value=50,  
 min=0,  
 max=100,  
 step=1,  
 description="Press play",  
 disabled=False  
)  
slider = widgets.IntSlider()  
widgets.jslink((play, 'value'), (slider, 'value'))  
widgets.HBox([play, slider])

### Date picker

The date picker widget works in Chrome and IE Edge, but does not currently work in Firefox or Safari because they do not support the HTML date input field.

widgets.DatePicker(  
 description='Pick a Date',  
 disabled=False  
)

### Color picker

widgets.ColorPicker(  
 concise=False,  
 description='Pick a color',  
 value='blue',  
 disabled=False  
)

### Controller

The Controller allows a game controller to be used as an input device.

widgets.Controller(  
 index=0,  
)

### Container/Layout widgets

These widgets are used to hold other widgets, called children. Each has a children property that may be set either when the widget is created or later.

### Box

items = [widgets.Label(str(i)) for i in range(4)]  
widgets.Box(items)

### HBox

items = [widgets.Label(str(i)) for i in range(4)]  
widgets.HBox(items)

### VBox

items = [widgets.Label(str(i)) for i in range(4)]  
left\_box = widgets.VBox([items[0], items[1]])  
right\_box = widgets.VBox([items[2], items[3]])  
widgets.HBox([left\_box, right\_box])

### Accordion

accordion = widgets.Accordion(children=[widgets.IntSlider(), widgets.Text()])  
accordion.set\_title(0, 'Slider')  
accordion.set\_title(1, 'Text')  
accordion

### Tabs

In this example the children are set after the tab is created. Titles for the tabes are set in the same way they are for Accordion.

tab\_contents = ['P0', 'P1', 'P2', 'P3', 'P4']  
children = [widgets.Text(description=name) for name in tab\_contents]  
tab = widgets.Tab()  
tab.children = children  
for i in range(len(children)):  
 tab.set\_title(i, str(i))  
tab

### Accordion and Tab use selected\_index, not value

Unlike the rest of the widgets discussed earlier, the container widgets Accordion and Tab update their selected\_index attribute when the user changes which accordion or tab is selected. That means that you can both see what the user is doing *and* programmatically set what the user sees by setting the value of selected\_index.

Setting selected\_index = None closes all of the accordions or deselects all tabs.

In the cells below try displaying or setting the selected\_index of the tab and/or accordion.

tab.selected\_index = 3

accordion.selected\_index = None

### Nesting tabs and accordions

Tabs and accordions can be nested as deeply as you want. If you have a few minutes, try nesting a few accordions or putting an accordion inside a tab or a tab inside an accordion.

The example below makes a couple of tabs with an accordion children in one of them

tab\_nest = widgets.Tab()  
tab\_nest.children = [accordion, accordion]  
tab\_nest.set\_title(0, 'An accordion')  
tab\_nest.set\_title(1, 'Copy of the accordion')  
tab\_nest

# Conclusion

Use this as a further reference for yourself!