Python course materials

# Widget List

This lecture will serve as a reference for widgets, providing a list of the GUI widgets available!

## Complete list

For a complete list of the GUI widgets available to you, you can list the registered widget types. Widget is the base class.

import ipywidgets as widgets  
  
# Show all available widgets!  
for item in widgets.Widget.widget\_types.items():  
 print(item[0][2][:-5])

## Numeric widgets

There are 10 widgets distributed with IPython that are designed to display numeric values. Widgets exist for displaying integers and floats, both bounded and unbounded. The integer widgets share a similar naming scheme to their floating point counterparts. By replacing Float with Int in the widget name, you can find the Integer equivalent.

### IntSlider

widgets.IntSlider(  
 value=7,  
 min=0,  
 max=10,  
 step=1,  
 description='Test:',  
 disabled=False,  
 continuous\_update=False,  
 orientation='horizontal',  
 readout=True,  
 readout\_format='d'  
)

### FloatSlider

widgets.FloatSlider(  
 value=7.5,  
 min=0,  
 max=10.0,  
 step=0.1,  
 description='Test:',  
 disabled=False,  
 continuous\_update=False,  
 orientation='horizontal',  
 readout=True,  
 readout\_format='.1f',  
)

Sliders can also be **displayed vertically**.

widgets.FloatSlider(  
 value=7.5,  
 min=0,  
 max=10.0,  
 step=0.1,  
 description='Test:',  
 disabled=False,  
 continuous\_update=False,  
 orientation='vertical',  
 readout=True,  
 readout\_format='.1f',  
)

### IntRangeSlider

widgets.IntRangeSlider(  
 value=[5, 7],  
 min=0,  
 max=10,  
 step=1,  
 description='Test:',  
 disabled=False,  
 continuous\_update=False,  
 orientation='horizontal',  
 readout=True,  
 readout\_format='d',  
)

### FloatRangeSlider

widgets.FloatRangeSlider(  
 value=[5, 7.5],  
 min=0,  
 max=10.0,  
 step=0.1,  
 description='Test:',  
 disabled=False,  
 continuous\_update=False,  
 orientation='horizontal',  
 readout=True,  
 readout\_format='.1f',  
)

### IntProgress

widgets.IntProgress(  
 value=7,  
 min=0,  
 max=10,  
 step=1,  
 description='Loading:',  
 bar\_style='', # 'success', 'info', 'warning', 'danger' or ''  
 orientation='horizontal'  
)

### FloatProgress

widgets.FloatProgress(  
 value=7.5,  
 min=0,  
 max=10.0,  
 step=0.1,  
 description='Loading:',  
 bar\_style='info',  
 orientation='horizontal'  
)

The numerical text boxes that impose some limit on the data (range, integer-only) impose that restriction when the user presses enter.

### BoundedIntText

widgets.BoundedIntText(  
 value=7,  
 min=0,  
 max=10,  
 step=1,  
 description='Text:',  
 disabled=False  
)

### BoundedFloatText

widgets.BoundedFloatText(  
 value=7.5,  
 min=0,  
 max=10.0,  
 step=0.1,  
 description='Text:',  
 disabled=False  
)

### IntText

widgets.IntText(  
 value=7,  
 description='Any:',  
 disabled=False  
)

### FloatText

widgets.FloatText(  
 value=7.5,  
 description='Any:',  
 disabled=False  
)

## Boolean widgets

There are three widgets that are designed to display a boolean value.

### ToggleButton

widgets.ToggleButton(  
 value=False,  
 description='Click me',  
 disabled=False,  
 button\_style='', # 'success', 'info', 'warning', 'danger' or ''  
 tooltip='Description',  
 icon='check'  
)

### Checkbox

widgets.Checkbox(  
 value=False,  
 description='Check me',  
 disabled=False  
)

### Valid

The valid widget provides a read-only indicator.

widgets.Valid(  
 value=False,  
 description='Valid!',  
)

## Selection widgets

There are several widgets that can be used to display single selection lists, and two that can be used to select multiple values. All inherit from the same base class. You can specify the **enumeration of selectable options by passing a list** (options are either (label, value) pairs, or simply values for which the labels are derived by calling str). You can **also specify the enumeration as a dictionary**, in which case the **keys will be used as the item displayed** in the list and the corresponding **value will be used** when an item is selected (in this case, since dictionaries are unordered, the displayed order of items in the widget is unspecified).

### Dropdown

widgets.Dropdown(  
 options=['1', '2', '3'],  
 value='2',  
 description='Number:',  
 disabled=False,  
)

The following is also valid:

widgets.Dropdown(  
 options={'One': 1, 'Two': 2, 'Three': 3},  
 value=2,  
 description='Number:',  
)

### RadioButtons

widgets.RadioButtons(  
 options=['pepperoni', 'pineapple', 'anchovies'],  
 # value='pineapple',  
 description='Pizza topping:',  
 disabled=False  
)

### Select

widgets.Select(  
 options=['Linux', 'Windows', 'OSX'],  
 value='OSX',  
 # rows=10,  
 description='OS:',  
 disabled=False  
)

### SelectionSlider

widgets.SelectionSlider(  
 options=['scrambled', 'sunny side up', 'poached', 'over easy'],  
 value='sunny side up',  
 description='I like my eggs ...',  
 disabled=False,  
 continuous\_update=False,  
 orientation='horizontal',  
 readout=True  
)

### SelectionRangeSlider

The value, index, and label keys are 2-tuples of the min and max values selected. The options must be nonempty.

import datetime  
dates = [datetime.date(2015,i,1) for i in range(1,13)]  
options = [(i.strftime('%b'), i) for i in dates]  
widgets.SelectionRangeSlider(  
 options=options,  
 index=(0,11),  
 description='Months (2015)',  
 disabled=False  
)

### ToggleButtons

widgets.ToggleButtons(  
 options=['Slow', 'Regular', 'Fast'],  
 description='Speed:',  
 disabled=False,  
 button\_style='', # 'success', 'info', 'warning', 'danger' or ''  
 tooltips=['Description of slow', 'Description of regular', 'Description of fast'],  
 # icons=['check'] \* 3  
)

### SelectMultiple

Multiple values can be selected with shift and/or ctrl (or command) pressed and mouse clicks or arrow keys.

widgets.SelectMultiple(  
 options=['Apples', 'Oranges', 'Pears'],  
 value=['Oranges'],  
 # rows=10,  
 description='Fruits',  
 disabled=False  
)

## String widgets

There are several widgets that can be used to display a string value. The Text and Textarea widgets accept input. The HTML and HTMLMath widgets display a string as HTML (HTMLMath also renders math). The Label widget can be used to construct a custom control label.

### Text

widgets.Text(  
 value='Hello World',  
 placeholder='Type something',  
 description='String:',  
 disabled=False  
)

### Textarea

widgets.Textarea(  
 value='Hello World',  
 placeholder='Type something',  
 description='String:',  
 disabled=False  
)

### Label

The Label widget is useful if you need to build a custom description next to a control using similar styling to the built-in control descriptions.

widgets.HBox([widgets.Label(value="The $m$ in $E=mc^2$:"), widgets.FloatSlider()])

### HTML

widgets.HTML(  
 value="Hello <b>World</b>",  
 placeholder='Some HTML',  
 description='Some HTML',  
)

### HTML Math

widgets.HTMLMath(  
 value=r"Some math and <i>HTML</i>: \(x^2\) and $$\frac{x+1}{x-1}$$",  
 placeholder='Some HTML',  
 description='Some HTML',  
)

### Image

file = open("images/WidgetArch.png", "rb")  
image = file.read()  
widgets.Image(  
 value=image,  
 format='png',  
 width=300,  
 height=400,  
)

## Button

widgets.Button(  
 description='Click me',  
 disabled=False,  
 button\_style='', # 'success', 'info', 'warning', 'danger' or ''  
 tooltip='Click me',  
 icon='check'  
)

# Conclusion

Even more widgets are described in the notebook **Widget List - Advanced**. Use these as a future reference for yourself!