SECTION 21: GUIs - 45 minutes, 7 parts

5/7 List of Possible Widgets

- -> a reference list of all the widgets available
- -> this is the third notebook
- -> there is a complete list of all the GUI widgets available
- -> he prints these out using an iteration list

- **-> there are 10 numeric widgets**
  - -> the integer slider
  - -> there are multiple different parameters for this (arguments)
  - -> if it has an orientation
  - -> the slider can also be named

- **-> the float slider**
  - -> the float
  - -> the step size can be a float
  - -> these can be displayed vertically

- **-> a range slider**
  - -> this can be used to grab a range

- **-> the float range slider**
  - -> this also shows the range

- **-> intprogress**
  - -> this looks like a loading bar which updates as a program runs
  - -> we also have a barstyle
  - -> there are different options for this argument
  - -> we also have bootstrap for this

- **-> floatprogress**
  - -> this also shows us if it is loading or not

- **-> bountedinttext**
  - -> numerical integer text which imposes some limit on the data
- -> bounded float text
- -> integer text and floating text

- **-> then boolean widgets**
  - -> three widgets which show boolean values
  - -> these have a toggle value
  - -> the checkbox

Widget List

PIERIAN DATA

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.

```
In [6]: import ipywidgets as widgets

        # Show all available widgets!
        for item in widgets.Widget.widget_types.items():
            print(item[0])
```

```
Jupyter.Checkbox
Jupyter.ToggleButton
Jupyter.Valid
```

```
In [8]: import ipywidgets as widgets

        # Show all available widgets!
        for item in widgets.Widget.widget_types.items():
            print(item[0])
```

```
Jupyter.Checkbox
Jupyter.ToggleButton
Jupyter.Valid
Jupyter.ButtonStyle
Jupyter.Button
Jupyter.Box
Jupyter.VBox
Jupyter.HBox
Jupyter.FloatText
Jupyter.BoundedFloatText
Jupyter.FloatSlider
Jupyter.FloatProgress
```

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

```
In [ ]: widgets.IntSlider(
            value=7,
            min=0,
            max=10,
            step=1,
            description='Test:',
            disabled=False,
            continuous_update=False,
            orientation='horizontal',
            readout=True,
            readout_format='d'
```

- -> then valid <- a read-only indicator
- -> to show something (if it is False)

- **-> selection widgets**
  - -> to select from a list
  - -> options as a parameter to this
  - -> three drop down options
  - -> we can also use a dictionary as opposed to a list

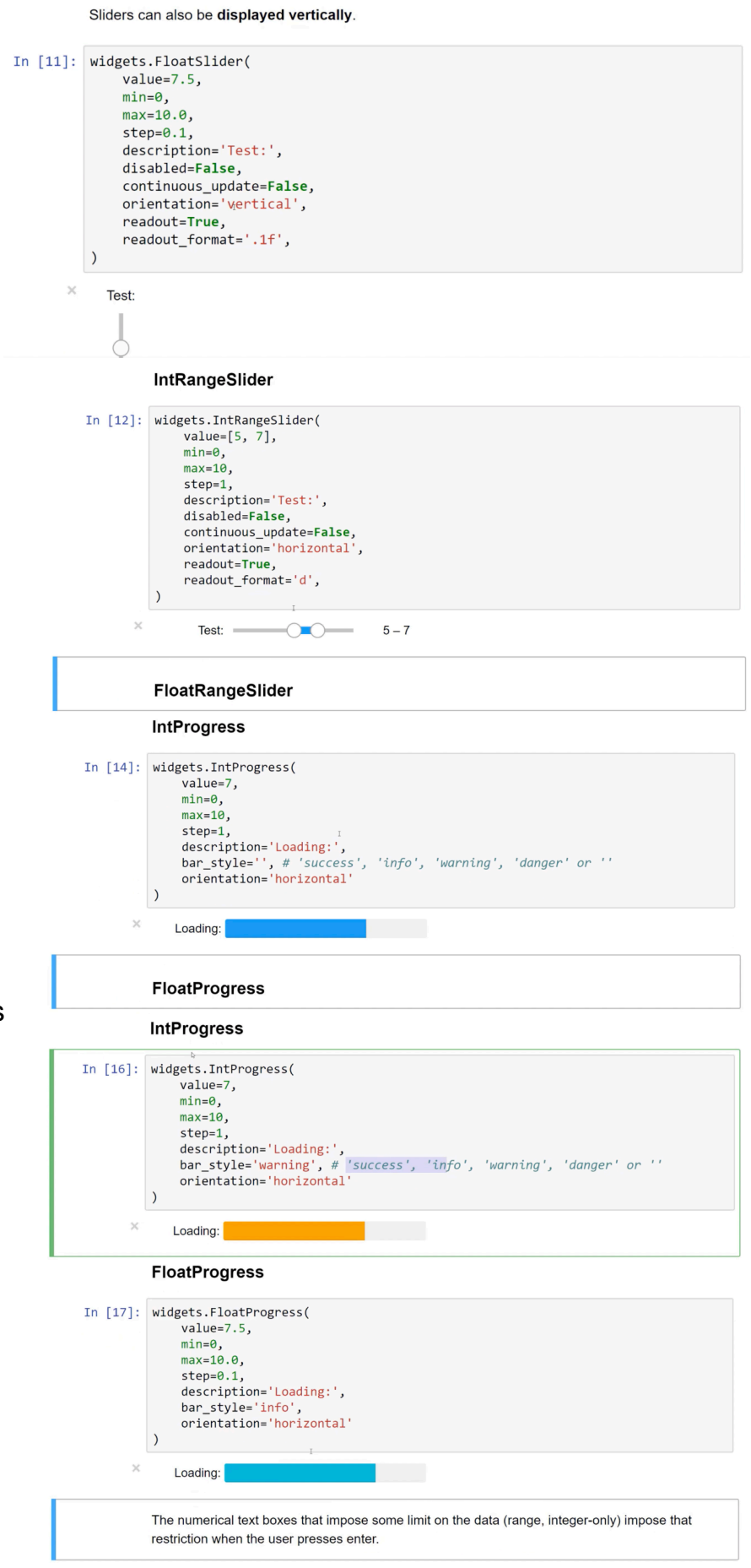
  - **-> dropdown**
    - -> to link up to the functions
    - -> dictionaries instead of lists
    - -> the output corresponds to the value, as opposed to the key
    - -> select
    - -> selection slider
      - -> this returns different options the more we slide
      - -> toggle buttons
      - -> we can toggle different values there

  - **-> select multiple**
    - -> fruits
  - **-> string widgets**
    - -> typing out different elements
  - **-> textarea**

  - **-> label**
    - -> custom widgets next to a controller
    - -> LaTeX in one of the arguments for this

  - **-> HTML**
    - -> putting in HTML and HTML Math

  - **-> Image**
    - -> this can be connected into an image which we would see
    - -> we would connect the image and see it there
    - -> we also have button
  - -> this is a list of all of the widgets
  - -> widgets in the next lecture



BoundedIntText

```
In [18]: widgets.BoundedIntText(
             value=7,
             min=0,
             max=10,
             step=1,
             description='Text:',
             disabled=False
         )
```

Text: 8

BoundedFloatText

```
In [19]: widgets.BoundedFloatText(
             value=7.5,
             min=0,
             max=10.0,
             step=0.1,
             description='Text:',
             disabled=False
         )
```

Text: 7.5

IntText

```
In [ ]: widgets.IntText(
            value=7,
            description='Any:',
            disabled=False
        )
```

Any: 7

FloatText

```
In [21]: widgets.FloatText(
             value=7.5,
             description='Any:',
             disabled=False
         )
```

Any: 7.5

Boolean widgets

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

ToggleButton

```
In [23]: widgets.ToggleButton(
             value=False,
             description='Click me',
             disabled=False,
             button_style='success', # 'success', 'info', 'warning', 'danger' or ''
             tooltip='Description',
             icon='check'
         )
```

✔ Click me

Checkbox

```
In [24]: widgets.Checkbox(
             value=False,
             description='Check me',
             disabled=False
         )
```

Check me ☐

Valid

The valid widget provides a read-only indicator.

```
In [26]: widgets.Valid(
             value=True,
             description='Valid!',
         )
```

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

```
In [27]: widgets.Dropdown(
             options=['1', '2', '3'],
             value='2',
             description='Number:',
             disabled=False,
         )
```

Number: 1

The following is also valid:

In [28]:
```
widgets.Dropdown(
    options={'One': 1, 'Two': 2, 'Three': 3},
    value=2,
    description='Number:',
)
```

Number: One

RadioButtons

In [29]:
```
widgets.RadioButtons(
    options=['pepperoni', 'pineapple', 'anchovies'],
    # value='pineapple',
    description='Pizza topping:',
    disabled=False
)
```

Pizza topping: pepperoni
pineapple
anchovies

Select

In [30]:
```
widgets.Select(
    options=['Linux', 'Windows', 'OSX'],
    value='OSX',
    # rows=10,
    description='OS:',
    disabled=False
)
```

OS: Windows

SelectionSlider

In [31]:
```
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
)
```

I like my eggs ... over easy

ToggleButtons

In [33]:
```
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 fa
    # icons=['check'] * 3
)
```

Speed: Slow Regular Fast

SelectMultiple

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

In [34]:
```
widgets.SelectMultiple(
    options=['Apples', 'Oranges', 'Pears'],
    value=['Oranges'],
    # rows=10,
    description='Fruits',
    disabled=False
)
```

Fruits
Apples
Oranges
Pears

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

In [35]:
```
widgets.Text(
    value='Hello World',
    placeholder='Type something',
    description='String:',
    disabled=False
)
```

String: Hello World

Textarea

In [36]:
```
widgets.Textarea(
    value='Hello World',
    placeholder='Type something',
    description='String:',
    disabled=False
)
```

String: Hello World

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.

In [37]:
```
widgets.HBox([widgets.Label(value="The $m$ in $E=mc^2$:"), widgets.FloatSlider()])
```

The m in E: 14.20

HTML

In [38]:
```
widgets.HTML(
    value="Hello <b>World</b>",
    placeholder='Some HTML',
    description='Some HTML',
)
```

Some HTML Hello **World**

HTML Math

In [39]:
```
widgets.HTMLMath(
    value=r"Some math and <i>HTML</i>: \(x^2\) and $$\frac{x+1}{x-1}$$",
    placeholder='Some HTML',
    description='Some HTML',
)
```

Some HTML Some math and *HTML*: x^2 and

$$\frac{x+1}{x-1}$$

Image

In [40]:
```
# file = open("images/WidgetArch.png", "rb")
# image = file.read()
# widgets.Image(
#     value=image,
#     format='png',
#     width=300,
#     height=400,
# )
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-40-c5950c5ba743> in <module>()
----> 1 file = open("images/WidgetArch.png", "rb")
      2 image = file.read()
      3 widgets.Image(
      4     value=image,
      5     format='png',

FileNotFoundError: [Errno 2] No such file or directory: 'images/WidgetArch.png'
```

Button

In [42]:
```
widgets.Button(
    description='Click me',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Click me',
    icon='check'
)
```

✔ Click me