

## SECTION 21: GUIs - 45 minutes, 7 parts

### 3/7 Interact Functionality with GUIs

#### • building GUIs with Jupyter

- -> graphical user interfaces
- -> there are multiple libraries for this
- -> these have licensing issues
- -> styling widgets
- -> interactive elements
- -> keeping the lecture notebooks, for reference code
- -> interact functionality

#### • in an ipynb

- -> we can't see the sliders for this
- -> we can't execute the code for this
- -> this is under bonus material
- -> **the interact function**
  - -> he imports ipywidgets
  - -> he then imports it as ipywidgets
  - -> auto-generating the user interface control for a user function argument
  - -> then calling the function with the arguments which we can manipulate interactively
  - -> he defines a function
  - -> and then uses the name of that function as the argument of another, with a second argument

```
In [80]: def func(x):  
         return x
```

```
In [81]: interact(func,x=10)
```



```
Out[81]: <function __main__.func>
```

- -> this returns a slider with different inputs to the function and their outputs
- -> passing booleans into this returns a checkbox instead of a slider
- -> strings return an auto-textbox

- -> **this can also be done with a decorator**
  - -> decorators are @interact() in this case
  - -> the second element can be run



# Python GUIs With IPyWidgets

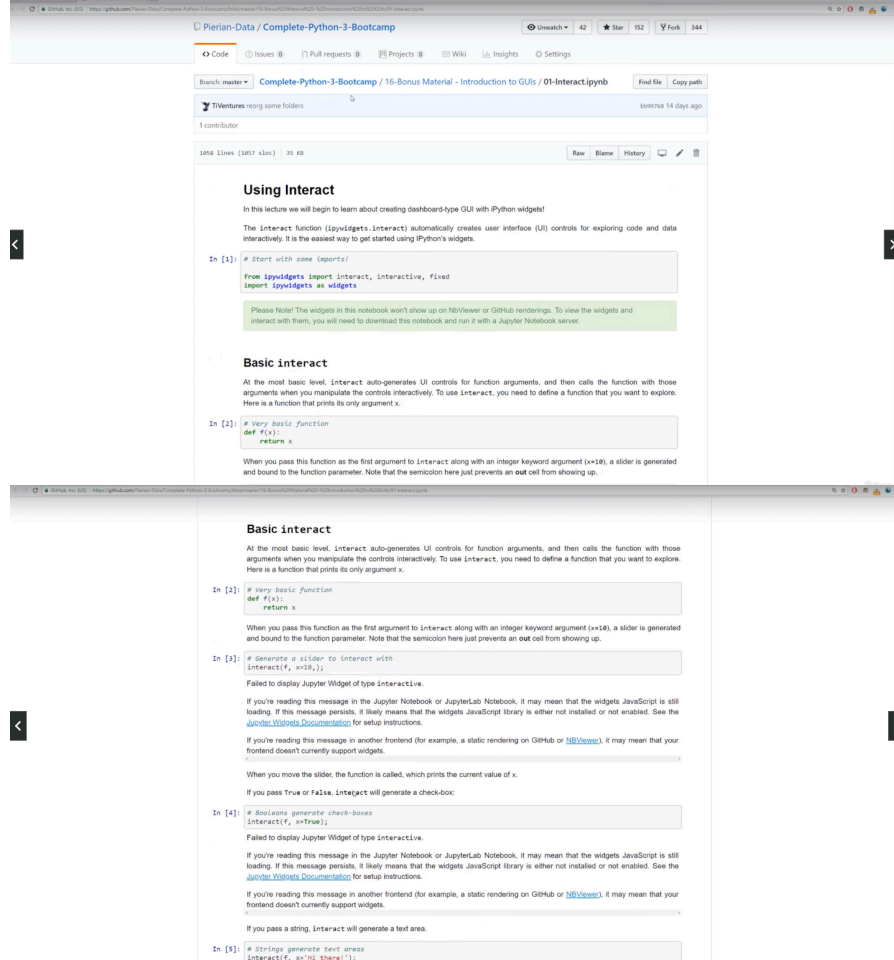
PIERIAN DATA



## Decorators

- We can build simple graphical user interfaces (GUI) with Jupyter!
- Let's explore how to build out these interactive elements.
- Keep the lecture notebooks handy, there is a lot of reference code and information there for you!

PIERIAN DATA



**Using Interact**

In this lecture we will begin to learn about creating dashboard-type GUI with Python widgets!

The `interact` function (`ipywidgets.interact`) automatically creates user interface (UI) controls for exploring code and data interactively. It is the easiest way to get started using Python's widgets.

```
In [1]: # Start with some imports!  
from ipywidgets import interact, interactive, fixed  
import ipywidgets as widgets
```

Please Note! The widgets in this notebook won't show up on NBViewer or GitHub renderings. To view the widgets and interact with them, you will need to download this notebook and run it with a Jupyter Notebook server.

**Basic interact**

At the most basic level, `interact` auto-generates UI controls for function arguments, and then calls the function with those arguments when you manipulate the controls interactively. To use `interact`, you need to define a function that you want to explore. Here is a function that prints its only argument `x`.

```
In [2]: # Very basic function  
def f(x):  
    return x
```

When you pass this function as the first argument to `interact` along with an integer keyword argument (`x=10`), a slider is generated and bound to the function parameter. Note that the semicolon here just prevents an out cell from showing up.

```
In [3]: # Generate a slider to interact with  
interact(f, x=10,)
```

Failed to display Jupyter Widget of type Interactive.

If you're reading this message in the Jupyter Notebook or JupyterLab Notebook, it may mean that the widgets JavaScript is still loading. If this message persists, it likely means that the widgets JavaScript library is either not installed or not enabled. See the [Jupyter Widgets Documentation](#) for setup instructions.

If you're reading this message in another frontend (for example, a static rendering on GitHub or [NBViewer](#)), it may mean that your frontend doesn't currently support widgets.

When you move the slider, the function is called, which prints the current value of `x`.

If you pass `True` or `False`, `interact` will generate a checkbox:

```
In [4]: # Booleans generate check-boxes  
interact(f, x=True)
```

Failed to display Jupyter Widget of type Interactive.

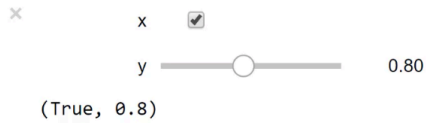
If you're reading this message in the Jupyter Notebook or JupyterLab Notebook, it may mean that the widgets JavaScript is still loading. If this message persists, it likely means that the widgets JavaScript library is either not installed or not enabled. See the [Jupyter Widgets Documentation](#) for setup instructions.

If you're reading this message in another frontend (for example, a static rendering on GitHub or [NBViewer](#)), it may mean that your frontend doesn't currently support widgets.

If you pass a string, `interact` will generate a text area.

```
In [5]: # Strings generate text areas  
interact(f, x='Hi there!')
```

```
In [88]: @interact(x=True,y=1.0)
def g(x,y):
    return (x,y)
```



## ○ -> widget abbreviations <- the .interact() method

- -> he gives more arguments to the widget
- -> this allows us to add different ranges to the slider
- -> there is an integer and float slider
- -> he is creating a minimum, maximum, step value and value
- -> there are ways to abbreviate all of this functionality

```
In [97]: interact(func,x=widgets.IntSlider(min=-100,max=100,step=1,value=0))
```



```
Out[97]: <function __main__.func>
```

```
In [99]: interact(func,x=(-10,10,1))
```



- -> min, max, step
- -> there is another function called the interact function
- -> the arguments of this are function and the second is in

```
In [100]: interact(func,x=(-10.0,10.0,.1))
```



in this case in the abbreviated format

```
In [101]: @interact(x=(0.0,20.0,0.5))
def h(x=5.0):
    return x
```



- -> this can also be done with the @interact decorator
- -> this is widget abbreviation <- start, stop, step
- -> he defines another function in this example

```
In [79]: from ipywidgets import interact,interactive,fixed
import ipywidgets as widgets
```

```
In [84]: def func(x):
    return x
```

```
In [85]: interact(func,x=10)
```



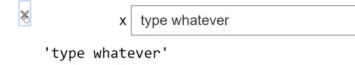
```
Out[85]: <function __main__.func>
```

```
In [ ]:
```

```
In [79]: from ipywidgets import interact,interactive,fixed
import ipywidgets as widgets
```

```
In [84]: def func(x):
    return x
```

```
In [87]: interact(func,x='Hello')
```



```
Out[87]: <function __main__.func>
```

```
In [ ]:
```

```
import ipywidgets as widgets
```

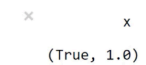
```
In [90]: def func(x):
    return x
```

```
In [91]: interact(func,x=fixed('Hello'))
```



```
Out[91]: <function __main__.func>
```

```
In [89]: @interact(x=True,y=fixed(1.0))
def g(x,y):
    return (x,y)
```



```
In [ ]:
```

```
In [97]: interact(func,x=widgets.IntSlider(min=-100,max=100,step=1,value=0))
```



```
Out[97]: <function __main__.func>
```

```
In [100]: interact(func,x=(-10.0,10.0,.1))
```



```
Out[100]: <function __main__.func>
```

```
In [ ]:
```

- -> the floating point slider
- -> increasing in increments of .1's
- -> we can also have drop down menus for this, with text to be entered
  - -> adding a list for this creates a drop down menu
- -> dictionaries with this
  - -> he passes a dictionary into this method
  - -> the output of this is equal to the value

## ○ -> the `interactive()` method

- -> **about this method**
  - -> to reuse the widgets we already have, or access the data that is bound to the user interface controls
  - -> the value of this isn't returned automatically
  - -> he imports modules
  - -> then defines a function
  - -> he uses `display()` in the definition for this
  - -> then runs `interactive()` on this
  - -> the type of this is an interactive object

## ▸ -> **the children of this attribute**

- -> this builds out a larger UI using the interactive functionality
- -> using `display` on this method prints two sliders
- -> this is to allow the user to see the inputs / outputs
- -> there is a port for the output of the function which can be removed by entering `interactive(.); <-` a semi-colon at the end of this

```

Out[100]: <function __main__.func>

In [101]: @interact(x=(0.0,20.0,0.5))
          def h(x=5.0):
              return x

x 12.00
12.0

In [103]: interact(func,x=['hello','option 2','option3'])

x option3
'option3'

Out[103]: <function __main__.func>

File Edit View Insert Cell Kernel Widgets Help Python 3

In [105]: from IPython.display import display
          def f(a,b):
              display(a+b)
              return a+b

In [106]: w = interactive(f,a=10,b=20)

In [107]: type(w)
Out[107]: ipywidgets.widgets.interaction.interactive

In [108]: w.children
Out[108]: (<ipywidgets.widgets.widget_int.IntSlider at 0x231d80574e0>,
<ipywidgets.widgets.widget_int.IntSlider at 0x231d80577f0>,
<ipywidgets.widgets.widget_output.Output at 0x231d8057438>)

In [109]: display(w)

a 13
b 43
56
  
```