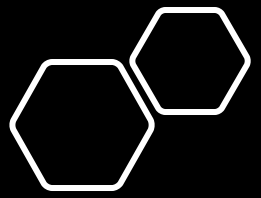
Creating Buttons and Triggering Events

In this kivy tutorial I will go over how to create buttons and trigger events when those buttons are clicked. I will also talk about creating multiple grid layouts to better display our widgets.



Importing Modules

Before we can start we need to import the following modules from Kivy.

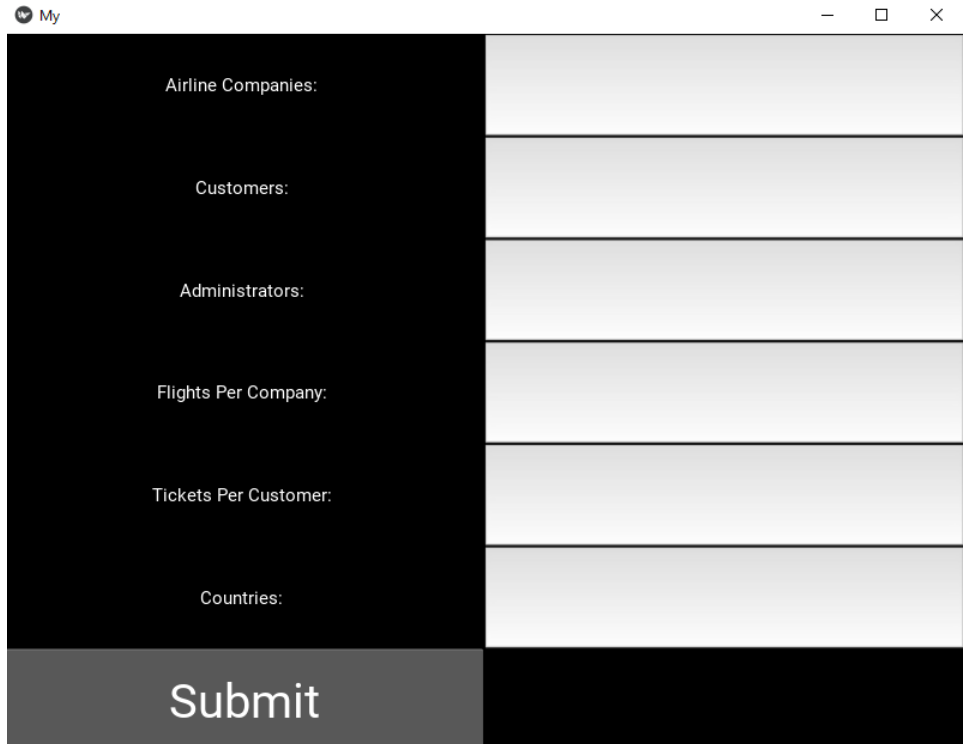
```
from kivy.uix.button import Button
```

Creating a Button

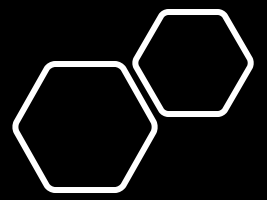
Creating a button can be done in a similar way to creating a text input box. To do so we simply declare a variable to hold our button and then add that to the grid layout.

```
self.submit = Button(text="Submit", font_size=40)
self.add_widget(self.submit)
# Adding this inside the __init__ of the class will create a button for us
```

Now when we run the program we should see a button appearing in our GUI.



However, notice that the button is not centered in our window! This is not ideal and we must do something to fix this.

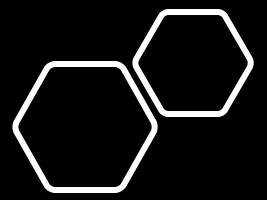


Multiple Layouts

The reason why our button is not centered is because we are using one grid layout that has 2 rows. This means when we add a button it is slotted in the first column of the forth row. A clever way of fixing this is by creating multiple grid layouts.

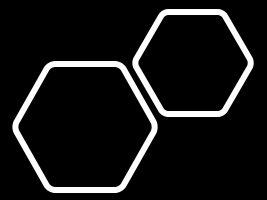
We can add all of the labels and text input boxes into a layout with two columns and then add that layout into another layout that only contains one column. We would also add our button into the layout with only one column so that it spans the entire bottom of the screen. Essentially we are going to be using two grid layouts and placing one inside of the other.

To do this we need to create another grid layout inside of our class and call it **inside** and add the appropriate widgets to it.



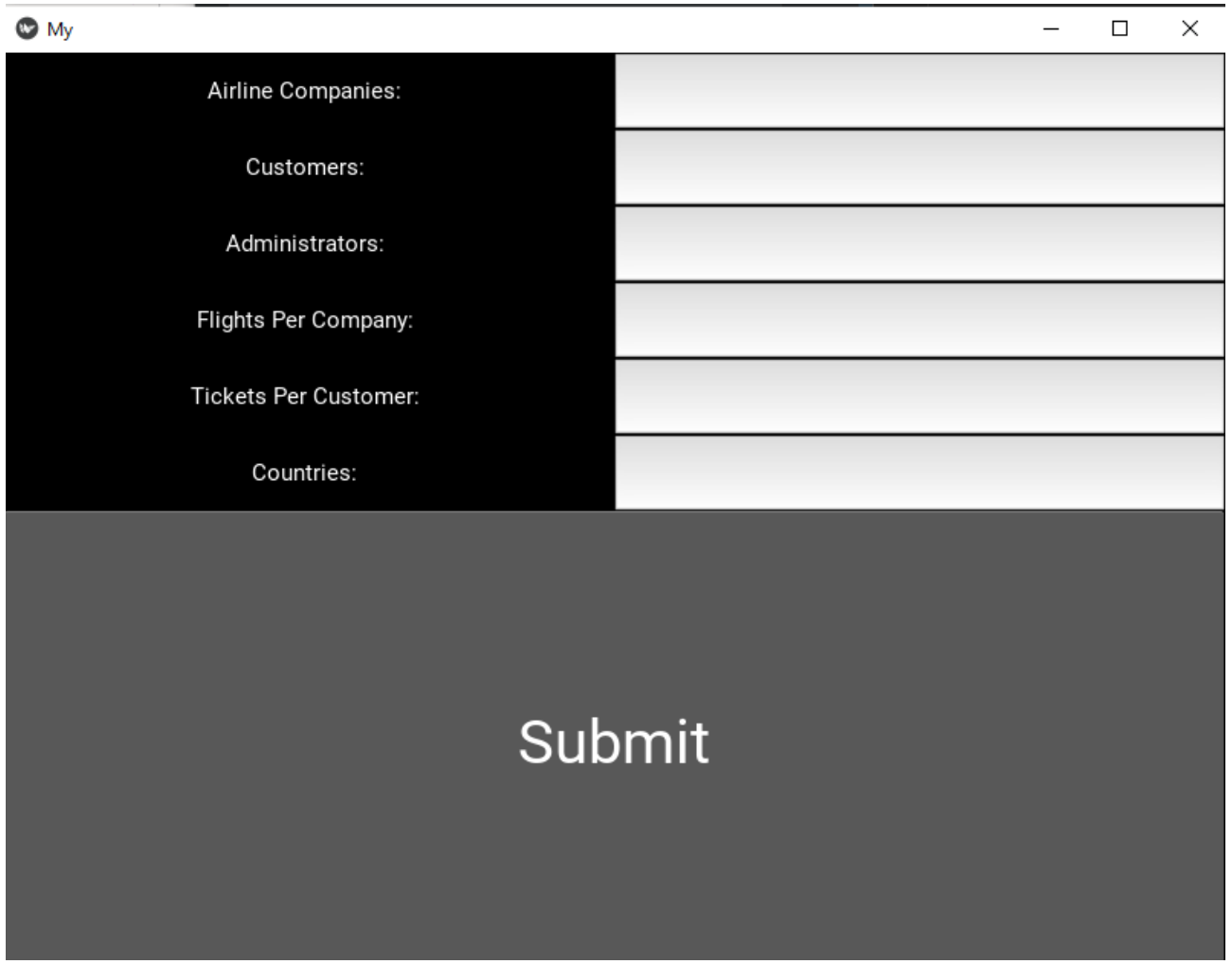
Multiple Layouts

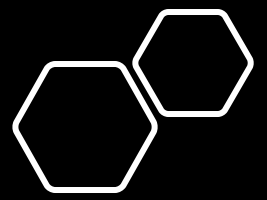
```
class MyGrid(GridLayout):
    def __init__(self, **kwargs):
        super(MyGrid, self).__init__(**kwargs)
        self.cols = 1 # Set columns for main layout

        self.inside = GridLayout() # Create a new grid layout
        self.inside.cols = 2 # set columns for the new grid layout

        # ALL OF THESE ARE APART OF THE (INTERIOR)NEW LAYOUT
        self.inside.add_widget(Label(text="Airline Companies: "))
        self.airline_companies = TextInput(multiline=False)
        self.inside.add_widget(self.airline_companies)

        self.inside.add_widget(Label(text="Customers: "))
        self.customers = TextInput(multiline=False)
        self.inside.add_widget(self.customers)

        self.inside.add_widget(Label(text="Administrators: "))
        self.administrators = TextInput(multiline=False)
        self.inside.add_widget(self.administrators)

        self.inside.add_widget(Label(text="Flights Per Company: "))
        self.flights_per_company = TextInput(multiline=False)
        self.inside.add_widget(self.flights_per_company)

        self.inside.add_widget(Label(text="Tickets Per Customer: "))
        self.tickets_per_customer = TextInput(multiline=False)
        self.inside.add_widget(self.tickets_per_customer)

        self.inside.add_widget(Label(text="Countries: "))
        self.countries = TextInput(multiline=False)
        self.inside.add_widget(self.countries)

        # ------------------------------------------------

        self.add_widget(self.inside)  # Add the interior layout to the main

        self.submit = Button(text="Submit", font_size=40)

        self.add_widget(self.submit)  # Add the button to the main layout
```



Multiple Layouts

Now when we run the program our button is centered nicely!



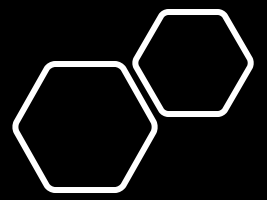


Binding Button Functions

Now that we have our button added we need to add some functionality to it. In my case when the button is clicked I'd like to collect all of the information from the form and print it to the console. I'd also like to clear the form so that it can be used again.

To accomplish this we need to do something called **binding**. We are going to bind a button to a function/method that we create so that when that button is clicked the function will run. To bind a button we use the following:

```
        # -----------------------------------------------

        self.add_widget(self.inside)  # Add the interior layout to the main

        self.submit = Button(text="Submit", font_size=40)

        self.submit.bind(on_press=self.pressed)
        # pressed is the name of the method we want to run
        # submit is clearly the name of our button

        self.add_widget(self.submit)  # Add the button to the main layout
```



Binding Button Functions

Now since we've bind our button to the method pressed we need to create that method. Inside of our class we will create the method pressed that contains one parameter. It is important that you include this.
To get the value of each of the text inputs in our form we can simply access the attribute **.text**. Similarly to change the text.

```
    # This goes inside our class MyGrid
    def pressed(self, instance):
        # Get the value of all of the tex inputs
        airline_companies = self.airline_companies.text
        customers = self.customers.text
        administrators = self.administrators.text
        flights_per_company = self.flights_per_company.text
        tickets_per_customer = self.tickets_per_customer.text
        countries = self.countries.text

        # print the values to the console
        print("Airline Companies:", airline_companies,
              "Customers:", customers,
              "Administrators:", administrators,
              "Flights Per Company:", flights_per_company,
              "Tickets Per Customer:", tickets_per_customer,
              "Countries:", countries)

        # Reset text to blank in each text input
        self.airline_companies.text = ""
        self.customers.text = ""
        self.administrators.text = ""
        self.flights_per_company.text = ""
        self.tickets_per_customer.text = ""
        self.countries.text = ""
```

If we run the program and click the button we should see that the text inputs get cleared and our input gets printed to the console.