Object Properties and .kv Continued

This python kivy tutorial covers object properties and continues to talk about the kivy .kv design language. Specifically mentioning how to reference objects from a .kv file in your python code.

Object Properties

After experimenting with the .kv language some of you may have asked the question: How do we access our elements (textinput, button etc.) from our python script? Well that is an excellent question and is what we have object properties for!

An object property allows us to create a reference to widgets from within a .kv file from our python script.

Modifying the .kv File

To set up object properties we need to create global variables from within our .kv file and assign these variables to the **id** property of specific widgets.

```
<MyGrid>:

 # Global variable name references the id name
    airline_companies: airline_companies
    customers: customers
    administrators: administrators
    flights_per_company: flights_per_company
    tickets_per_customer: tickets_per_customer
    countries: countries

    GridLayout:
        cols:1
        size: root.width - 200, root.height -200
        pos: 100, 100

        GridLayout:
            cols:2
            Label:
                text: "Airline Companies: "
            TextInput:
                id: airline_companies  # <- Add this
                multinline:False
            Label:
                text: "Customers: "
            TextInput:
                id: customers  # <- Add this
                multiline:False
            Label:
                text: "Administrators: "
            TextInput:
                id: administrators  # <- Add this
                multinline:False
            Label:
                text: "Flights Per Company: "
            TextInput:
                id: flights_per_company  # <- Add this
                multiline:False
            Label:
                text: "Tickets Per Customer: "
            TextInput:
                id: tickets_per_customer  # <- Add this
                multinline:False
            Label:
                text: "Countries: "
            TextInput:
                id: countries  # <- Add this
                multiline:False
        Button:
            text:"Submit"
```

Modifying the Python Script

The first thing we need to do to use an object property from within our python script is import the specific module.

```
from kivy.properties import ObjectProperty
```

Next we will define two object properties from within our class **MyGrid**.

```
class MyGrid(Widget):

    airline_companies = ObjectProperty(None)
    customers = ObjectProperty(None)
    administrators = ObjectProperty(None)
    flights_per_company = ObjectProperty(None)
    tickets_per_customer = ObjectProperty(None)
    countries = ObjectProperty(None)
```

We initialize the values as None to start as when we first create the class they will have no value.
Now if we want to access the value of the TextInput box with id airline_companies we can simply use self.airline_companies to do so.

Creating a Button On_Press

Like we had in previous tutorials we'd like our button to perform a function when it is clicked. To accomplish this we need to create a method inside of our **MyGrid** class that we can call each time our button is pressed. We will call this **btn()**.

```
# Goes inside MyGrid Class
def btn(self):
    print("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)
```

This method will print the contents of our form and clear both of the text input boxes.

Now we need to tell the button what it should call when it is pressed. To do this we will add the property **on_press** in our .kv file.

```
Button:
    text:"Submit"
    on_press: root.btn() # <- Add this
```

We use app.btn() to reference the class that contains our btn method.
Now we should have a functioning application that works the same as the one we created in tutorial #3.

However, we have successfully separated our logic from our styling and elements.

Full code main.py

```python
import kivy
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.properties import ObjectProperty


class MyGrid(Widget):

    airline_companies = ObjectProperty(None)
    customers = ObjectProperty(None)
    administrators = ObjectProperty(None)
    flights_per_company = ObjectProperty(None)
    tickets_per_customer = ObjectProperty(None)
    countries = ObjectProperty(None)

    def btn(self):
        print("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)

        self.airline_companies.text = ""
        self.customers.text = ""
        self.administrators.text = ""
        self.flights_per_company.text = ""
        self.tickets_per_customer.text = ""
        self.countries.text = ""

class MyApp(App):
    def build(self):
        return MyGrid()


if __name__ == "__main__":
    MyApp().run()
```