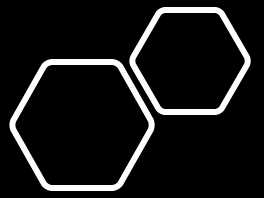
The kv Design Language (.kv File)

This is a tutorial for the kv design language. The kv design language allows for separation between design and logic. You can specify all of the widgets and elements in your application in one file and code the logic in your python script.

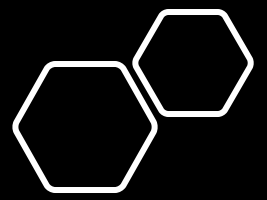


The kv Design Language

Up until this point we have been adding widgets to our window/app using python code. This is acceptable and works fine however there is a much easier and nicer way to do this.
Kivy has something called the kv design language.

You think of it as a language similar to HTML and CSS where it is responsible for styling and adding elements to the display but does not handle any logic. In this tutorial I will show you how to create a .kv file and create the graphical part of your application from within that file.

Note: We still need our python script to handle the logic and load the application.



Creating a .kv File

There are a few conventions we need to follow when creating a .kv file.

Naming: The name of your .kv file must follow the rules below in order for python/kivy to be able to see and load the file.

1. It must be all lowercase
2. It must match with the name of your main class. (The one that has the build method)
3. If the name of your main class ends in "app" (lowercase or uppercase) you must not include "app" in your file name.

File Location: The file must be in the same directory as your python script.

File Extension: The file must be saved as type *All Files and end with .kv

For example:

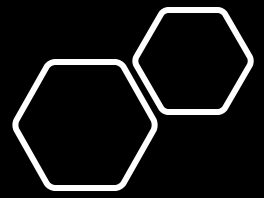
My main classes name is **MyApp**. Therefore I will name my file **my.kv**.

```python
import kivy
from kivy.app import App
from kivy.uix.label import Label
from kivy.uix.gridlayout import GridLayout
from kivy.uix.textinput import TextInput
from kivy.uix.button import Button
from kivy.uix.widget import Widget


class MyGrid(Widget):
    pass


class MyApp(App): # <- Main Class
    def build(self):
        return MyGrid()


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



Setting up The Python Script

Before continuing please import the following module from your python script.

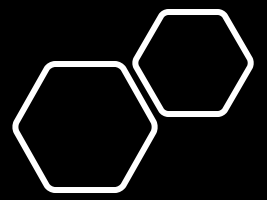
```
from kivy.uix.widget import Widget
```

The first thing we need to do to add widgets to our screen using .kv is set up an empty class in our python script. This class will simply inherit from Widget and will be what is used from within the kv file to add widgets.

```
class MyGrid(Widget):
    pass
```

Now ensure that your main classes build method returns an instance of MyGrid and we are ready to go.

```
class MyApp(App): # <- Main Class
    def build(self):
        return MyGrid()
```



Adding Widgets From a .kv File

Two important things to remember about .kv files:
- Capitals are Important
- Indentation is important

The first thing we do when writing in a .kv is declare the class we will be adding widgets to. In my case it's **MyGrid**.

```
# Filename: my.kv

<MyGrid>:
```

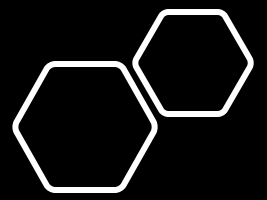
The next step is to define the widgets that we want to add to that class. Each widget also has several properties associated with it that we can change.

```
# Filename: my.kv

<MyGrid>:
    Label:
        text: "Name: "

    TextInput:
        multiline:False
```

You can see that the widgets that are inside of the MyGrid class are tabbed in once and the properties of those widgets are tabbed twice.



Creating a Form With a .kv File

The file below is the recreation of the form we made in tutorial 3, this time using a kv file. You can see that we have multiple grid layouts like before and that we add widgets to those layouts.

Notice that to make these widgets fill the entire screen we must change the **size** attribute. By making the size of the first grid layout **root.width, root.height** we are telling it to fill the size of the window dynamically.

```
# Filename: my.kv

<MyGrid>:
    GridLayout:
        cols:1
        size: root.width, root.height

        GridLayout:
            cols:2

            Label:
                text: "Airline Companies: "
            TextInput:
                multinline:False

            Label:
                text: "Customers: "
            TextInput:
                multiline:False

            Label:
                text: "Administrators: "
            TextInput:
                multinline:False

            Label:
                text: "Flights Per Company: "
            TextInput:
                multiline:False
            Label:
                text: "Tickets Per Customer: "
            TextInput:
                multinline:False

            Label:
                text: "Countries: "
            TextInput:
                multiline:False
        Button:
            text:"Submit"
            on_press: app.btn()
```

Creating a Form With a .kv File