Kivy Framework Cheat Sheet

If you are writing in Python 3

On the top of the python `main.py` file put:

```
        from __future__ import (absolute_import, division, print_function,
unicode_literals)
        from builtins import *
```

These two lines are not necessary but recommended, they will ensure that your code is Python 2 compatible, some packagers like Buildozer works better with Python 2

In all applications the following imports should be done:

```
        import kivy
        kivy.require('1.10.0')  # replace with your current kivy version !
        from kivy.app import App
```

Controling the initial window size of the kivy app:

Before the window creation:

```
        from kivy.config import Config
        Config.set('graphics', 'width', '200')
        Config.set('graphics', 'height', '200')     # put your desired
values instead of 200
```

Or dynamically After The window creation:

```
        from kivy.core.window import Window
        Window.size = (300, 100)
```

Typical Python `main.py` file:

```
        import kivy
        kivy.require('1.10.0')
        from kivy.app import App

        class TypicalApp(App):
            pass

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

A simple typical Kivy `typical.kv` file:

```
        #:kivy `1.10`   # kivy header line to declare the kivy version used

        BoxLayout:    # The root layout(widget), can be offcourse any other
type of layout
            someproperty:
            someproperty:
            someproperty:
            someproperty:

            SomeAnotherWidget:  #This is a direct child of the root
layout(widget)
                someproperty:
                someproperty:
                someproperty:
                someproperty:
                someproperty:


            SomeLayout: # this is another layout that will be included in
the first root layout, usually this is not needed
                someproperty:
                someproperty:
                someproperty:

                SomeWidget:     # this is a child of the second layout
                    someproperty:
                    someproperty:
                    someproperty:
```

A typical Kivy `typical.kv` file with a custom layout:

```
        #:kivy `1.10`   # kivy header line to declare the kivy version used

        TypicalScreen:    # The root widget which is a custom widget defined
below but only called beacuse of this line, i.e.: TypicalScreen:
        <TypicalScreen@BoxLayout>:
        ## Defining our custom widget, The `<` `>` means that this is a
class, the `@` means it inherits from the `BoxLayout` layout class,
the BoxLayout can be changed to any other layout or widget
        ## The above line is equivalent in python to :
        # class TypicalScreen(BoxLayout):
        #   def __init__(self,**kwargs):
        #       super(TypicalScreen, self).__init__(**kwargs)
```

```
        ## And offcourse this will mean the following change in the App
subclass:
        # class MyApp(App):
        #     def build(self):
        #         return MyScreen()
        ## Offcourse this will complicate things up, so subclassing the
widget in the kivy file as shown here is preferred
            someproperty:
            someproperty:
            someproperty:
            someproperty:

            SomeAnotherWidget:  #This is a direct child of the root
layout(widget)
                someproperty:
                someproperty:
                someproperty:
                someproperty:
                someproperty:


            SomeLayout: # this is another layout that will be included in
the first root layout, usually this is not needed
                someproperty:
                someproperty:
                someproperty:

                SomeWidget:     # this is a child of the second layout
                    someproperty:
                    someproperty:
                    someproperty:
```

Available layouts

And their respective import statemnts

```
        from kivy.uix.anchorlayout AnchorLayout

        from kivy.uix.floatlayout import FloatLayout

        from kivy.uix.gridlayout import GridLayout

        from kivy.uix.pagelayout import PageLayout

        from kivy.uix.relativelayout import RelativeLayout

        from kivy.uix.scatterlayout import ScatterLayout
```

```
from kivy.uix.stacklayout import StackLayout
```

> Tip: in any python file, make an import to one of the above layouts, lets' say AnchorLayout, Then in a new line type: `AnchorLayout.` then using the code completion feature in your text editor, view the available attributes in this class together with the description (documentation) available for it, the attribute name in any class is the same name that can be used in the `.kv` file, with this method you can understand a lot about every class in the kivy framework, also play around with the methods (functions) available in this class. This trick is one of the main methods that was used to make this cheat sheet.

Available Classical Widgets

And their respective import statemnts

```
from kivy.uix.label import Label

from kivy.uix.button import Button

from kivy.uix.checkbox

from kivy.uix.image

from kivy.uix.slider

from kivy.uix.progressbar

from kivy.uix.textinput

from kivy.uix.togglebutton

from kivy.uix.switch

from kivy.uix.video
```

Available Complex Widgets

And their respective import statemnts

```
from kivy.uix.bubble

from kivy.uix.dropdown

from kivy.uix.filechooser

from kivy.uix.popup
```

```
from kivy.uix.spinner

from kivy.uix.listview import ListView

from kivy.uix.tabbedpanel

from kivy.uix.videoplayer

from kivy.uix.vkeyboard
```

Behaviours Widgets

And their respective import statemnts

```
api-kivy.uix.scatter

api-kivy.uix.stencilview
```

Screen Manager

And their respective import statemnts

```
api-kivy.uix.screenmanager
```

Widget Examples:

GridLayout

- Python:

```
from kivy.uix.gridlayout import GridLayout
```

- Kivy:

```
GridLayout:
    spacing: 10     #Spacing between children i.e.: widgets, can be
: x,y (horizontal spacing, vertical spacing), or can be one argument
only, defaults to 0,0 if not added
    padding: 10     # Padding between the layout box and it's
children, can be 4 arguments (left,top,right,bottom), or
2(horizontal,vertical), or one only, defaults to 0,0,0,0 if not
added
    cols: 2     # Number of columns in the grid, defaults to 0
    rows: 2     # Number of rows in the grid, defaults to 0
```

```
            col_default_width: 0    # Default minimum size to use for a
column, defaults to 0
            row_default_height: 0   # Default minimum size to use for row,
defaults to 0
            col_force_default: False    # If True, ignore the width and
size_hint_x of the child and use the default column width, default
is False
            row_force_default: False    # If True, ignore the height and
size_hint_y of the child and use the default row height, default is
False
```

Scheduling events:

```
        #....
        from kivy.clock import Clock
        # Method 1:
        event = Clock.schedule_interval(function, interval)   # This method
will trigger the function every some interval (seconds)

        # But the function passed here must have a patameter of delta time
as an argument in its signature

        def function(dt):
            # some code
        # the dt parameter is not needed to be used in the function

        # Method 2:
        event = Clock.schedule_once(function, delay)    # This will trigger
function once only after some delay in seconds

        # Again this function should have 'dt' as an argument
        # Off course if this function is part of a class, it should look
like this:

        def function(self,dt):
            # some code

        # If at some condition or at some point in your code, you need to
disable the scheduled event, i.e. unschedule it, use:
        event.cancel()

        # Or:
        Clock.unschedule(event)

        # Or in the signature of the function itself, return False at the
desired cancelation condition
```

Controling the environment

```
        import os
        os.environ["KIVY_NO_CONSOLELOG"]= "1"   #This will prevent printing
logs to the console
        import kivy     # the previous line should be before importing kivy
```

Config file

Location: `C:\Users\Home_Folder\.kivy\config.ini`