Para carregar um ou vários arquivos .kv

from kivy.lang import Builder

para cusar o arquivo .kv basta criar uma variável, e atribuir a ela o Builder.load_file() com o nome do arquivo .kv como argumento da função!

root = Builder.load_file(self.filename)

depois adicione a variável criada ao gerenciador de layout que estiver sendo atribuido!

Window.add_widget(root)

Classes dinamicas na linguagem kv

<nome@herança1+herança2>:

```
<botão@ButtonBehavior+Label>:
	canvas.before:
		color:
			rgba: 0.1, 0.5, 0.7, 1
		Ellipse:
			pos: self.pos
			size: self.height, self.height

		Ellipse:
			pos: self.x + self.width – self.height, self.y
			size: self.height, self.height

		Rectangle:
			pos: self.x+self.height/2, self.y
			size: self.width – self.height, self.heigth
```

Classes dinamicas na linguagem python

```
from kivy.uix.behaviors.button import ButtonBehavior
from kivy.graphics import Collor, Ellipse, Rectangle
from kivy.label import Label

class botão(ButtonBehavior, Label):
	def __init__(self, **kwargs):
		super(botão, self).__init__(**kwagrs)
		with self.canvas.before:
			Collor(rgba=(0.1, 0.5, 0.7, 1))
			Ellipse(size=(self.height, self.height),
					pos=self.pos)
			Ellipse(size=(self.height, self.height),
					pos=(self.x + self.width – self.height, self.y))
			Rectangle(size=(self.width-self.height, self.height),
					pos=(self.x + self.height / 2.0, self.y))
```

para redesenhar os botoes vamos dividir em funções para melhor controle

```
from kivy.uix.behaviors.button import ButtonBehavior
```

```
from kivy.graphics import Collor, Ellipse, Rectangle
from kivy.label import Label

class botão(ButtonBehavior, Label):
        def __init__(self, **kwargs):
                ‘’’ Executar quando iniciar ’’’

                super(botão, self).__init__(**kwagrs)
                self.Atualizar()

        def on_pos(self, *args):
                ‘’’ Executar quando mudar a posição‘’’

                self.Atualizar()

        def on_size(self, *args):
                ‘’’ Executar quando mudar de tamanho‘’’

                self.Atualizar()


        def Atualizar(self, *args):
                ‘’’ Executar para redesenhar as instruções graficas
                nos widgets e janelas‘’’

                self.canvas.before.clear() # acrescentado para limpar a tela  antes de redesenhar a tela

                with self.canvas.before:
                        Collor(rgba=(0.1, 0.5, 0.7, 1))
                        Ellipse(size=(self.height, self.height),
                                        pos=self.pos)
                        Ellipse(size=(self.height, self.height),
                                        pos=(self.x + self.width – self.height, self.y))
                        Rectangle(size=(self.width-self.height, self.height),
                                        pos=(self.x + self.height / 2.0, self.y))
```

Criando o ActionBar do Android

para criarmos um actionBar vamos para a tela que desejamos adicionar o ActionBar
e adicionar os códigos a seguir, isto na linguagem kv ok!

Pra fazer o menu em cima, basta colocálo como primeira opção após o gerenciador de layout!

```
<Tela>:
        BoxLayout:

                orientation:’vertical’

                ActionBar:

                        ActionView: # todo ActionBar deve conter um ActionView
```

```
                ActionPrevious:

                    title: ‘Tela’
                    on_release:app.root.current = ‘menu’

                ActionButton:

                    text:’Sair’
                    on_release:app.stop()

                ActionSeparator:

                ActionButton:

                    text:’Menu’
                    on_release:app.root.current = ‘menu’
```

Incluindo outros arquivos kv em um arquivo kv

```
#:kivy 1.8.0
#button.kv

<SpecialButton>:
    canvas:
        Color:
            rgba: 1.0, 0.0, 0.0, 1.0
        Rectangle:
            pos: self.pos
            size: (self.size[0]/4, self.size[1]/4)
```

no outro arquivo kv

```
#:include button.kv

<CustomLayout>:
    SpecialButton:
        text: 'Includes!'
```