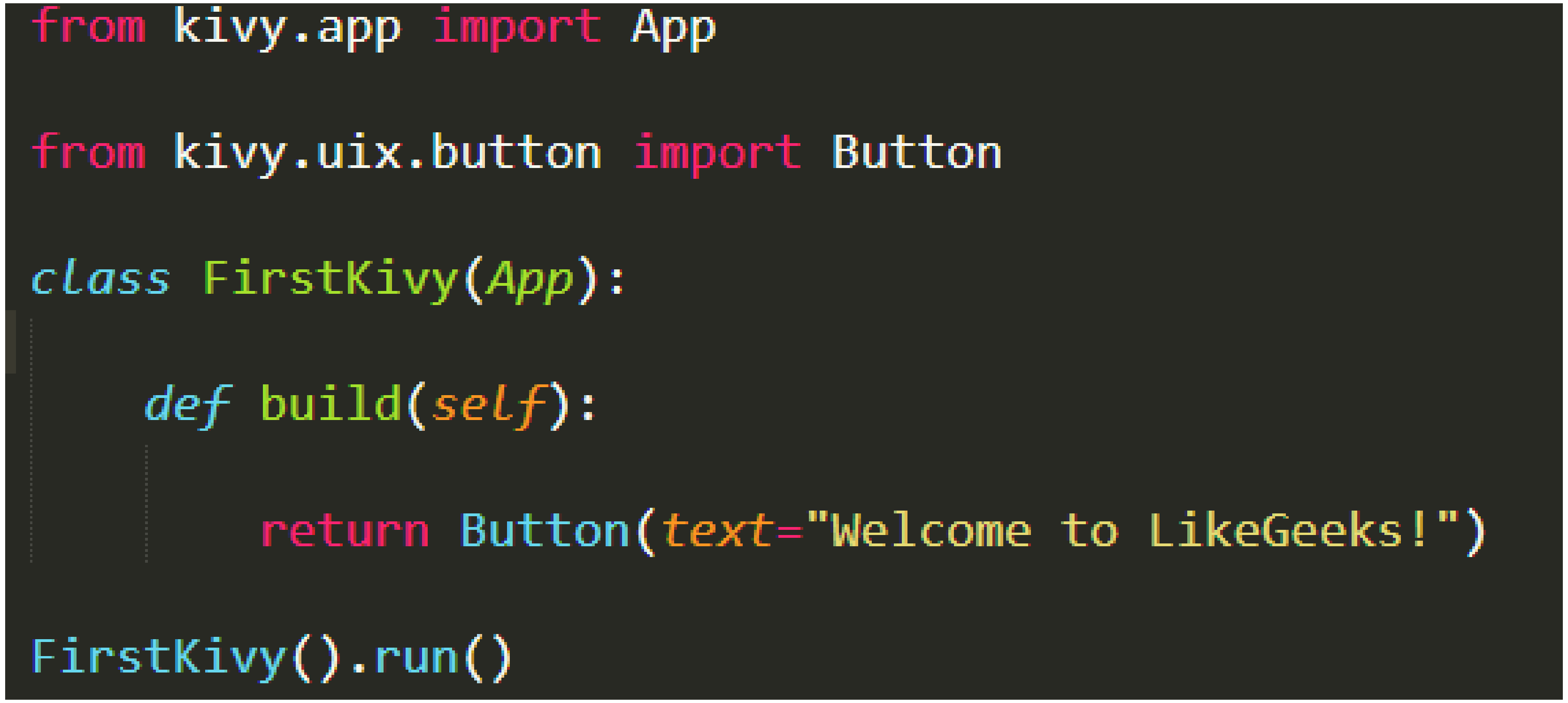
Kivy Button

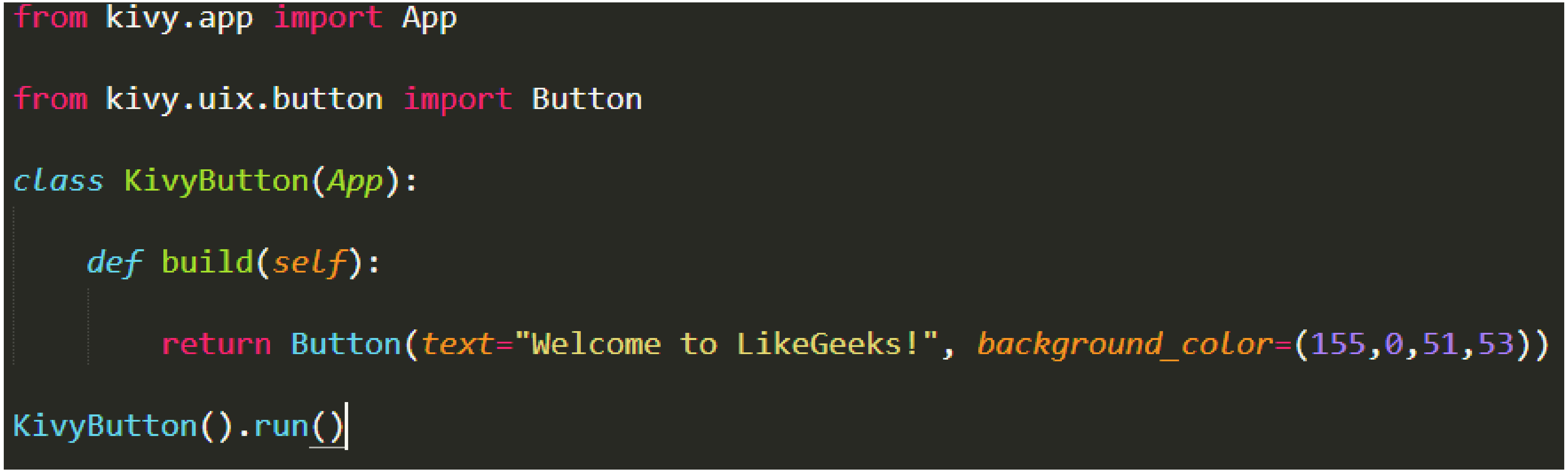
В этом разделе вы узнаете, как создать кнопку, изменить цвет кнопки, включить / отключить, как добавить изображение на кнопку и как изменить размер и положение кнопки.



```
from kivy.app import App

from kivy.uix.button import Button

class FirstKivy(App):

    def build(self):

        return Button(text="Welcome to LikeGeeks!")

FirstKivy().run()
```

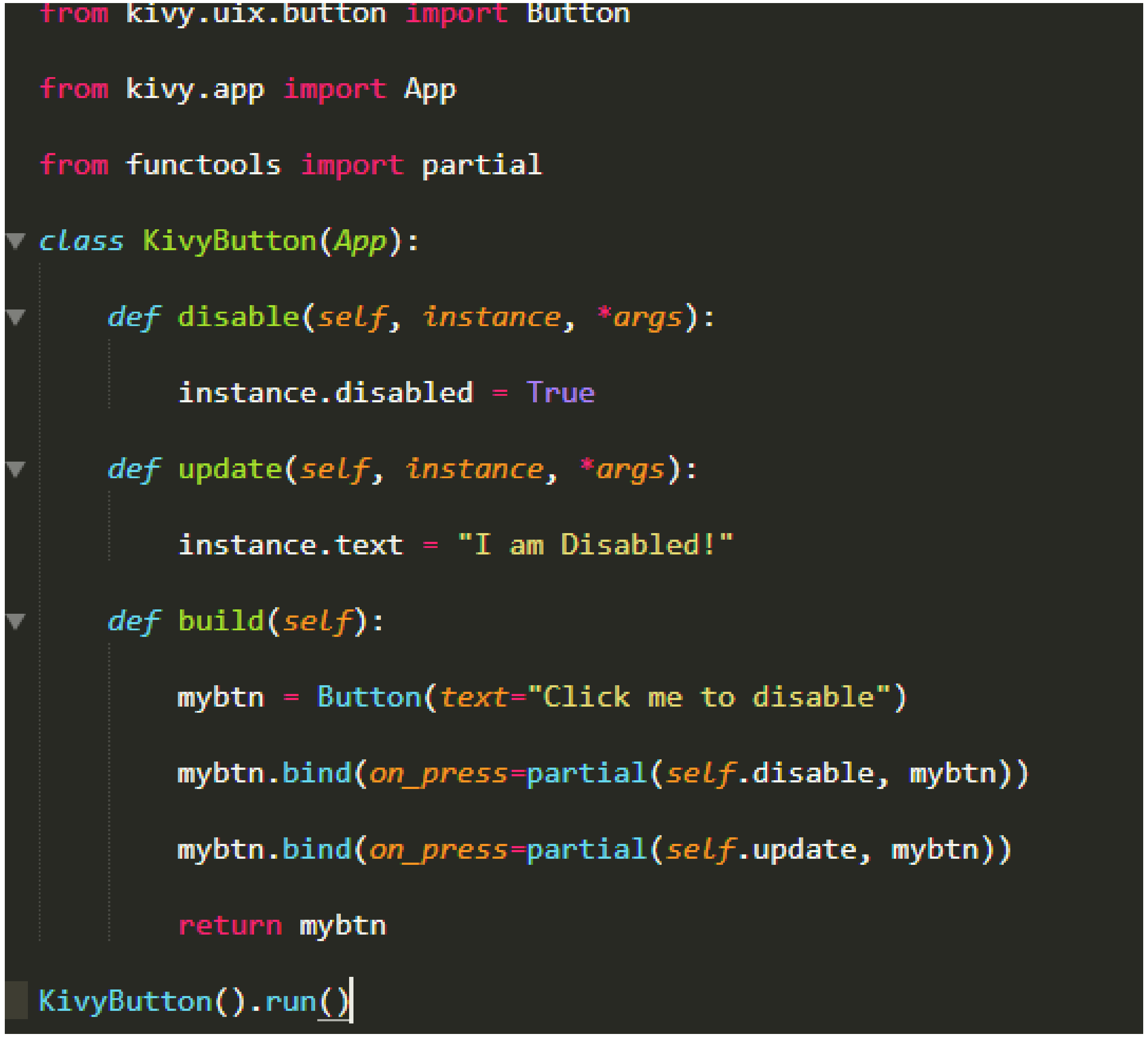
Change color of Kivy button

Цвет по умолчанию для кнопки Kivy - серый. Цвет можно изменить, указав свойство background_color в формате (r, g, b, a). Код демонстрируется ниже:



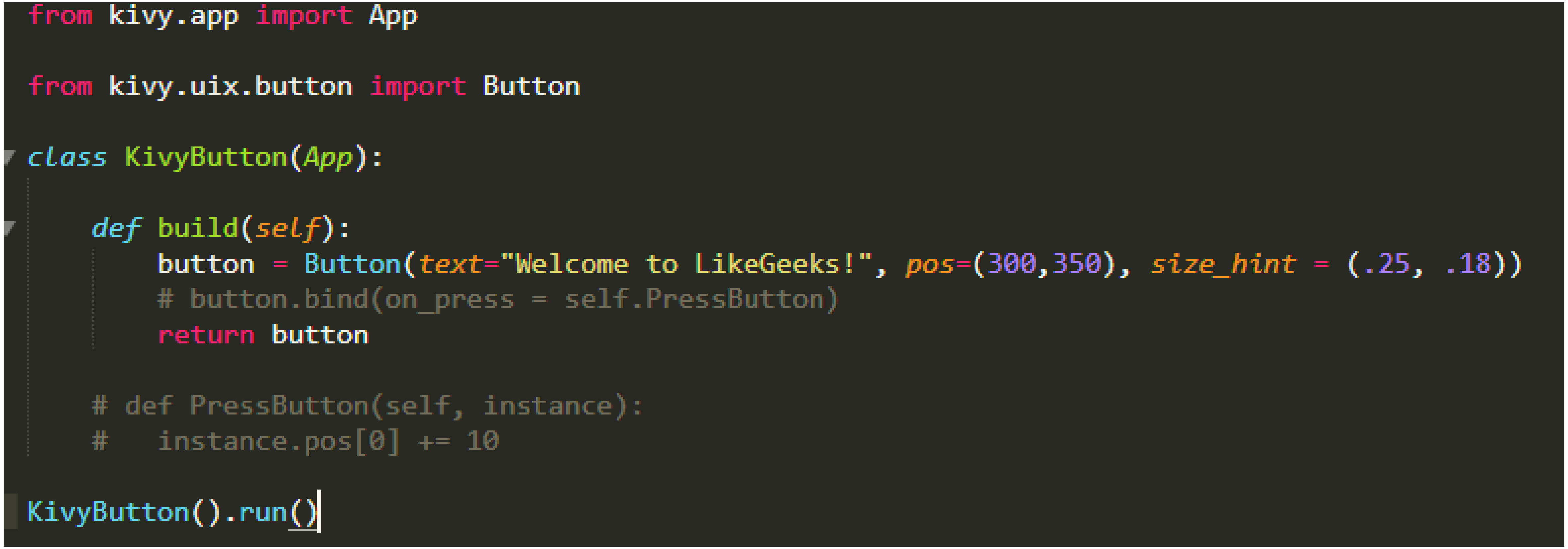
```
from kivy.app import App

from kivy.uix.button import Button

class KivyButton(App):

    def build(self):

        return Button(text="Welcome to LikeGeeks!", background_color=(155,0,51,53))

KivyButton().run()
```

Disable Kivy Button



```
from kivy.uix.button import Button

from kivy.app import App

from functools import partial

class KivyButton(App):

    def disable(self, instance, *args):

        instance.disabled = True

    def update(self, instance, *args):

        instance.text = "I am Disabled!"

    def build(self):

        mybtn = Button(text="Click me to disable")

        mybtn.bind(on_press=partial(self.disable, mybtn))

        mybtn.bind(on_press=partial(self.update, mybtn))

        return mybtn

KivyButton().run()
```

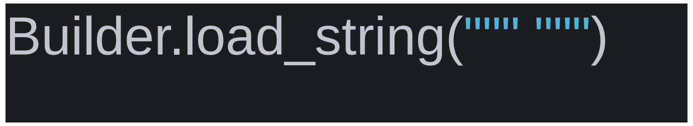
Change the size and position



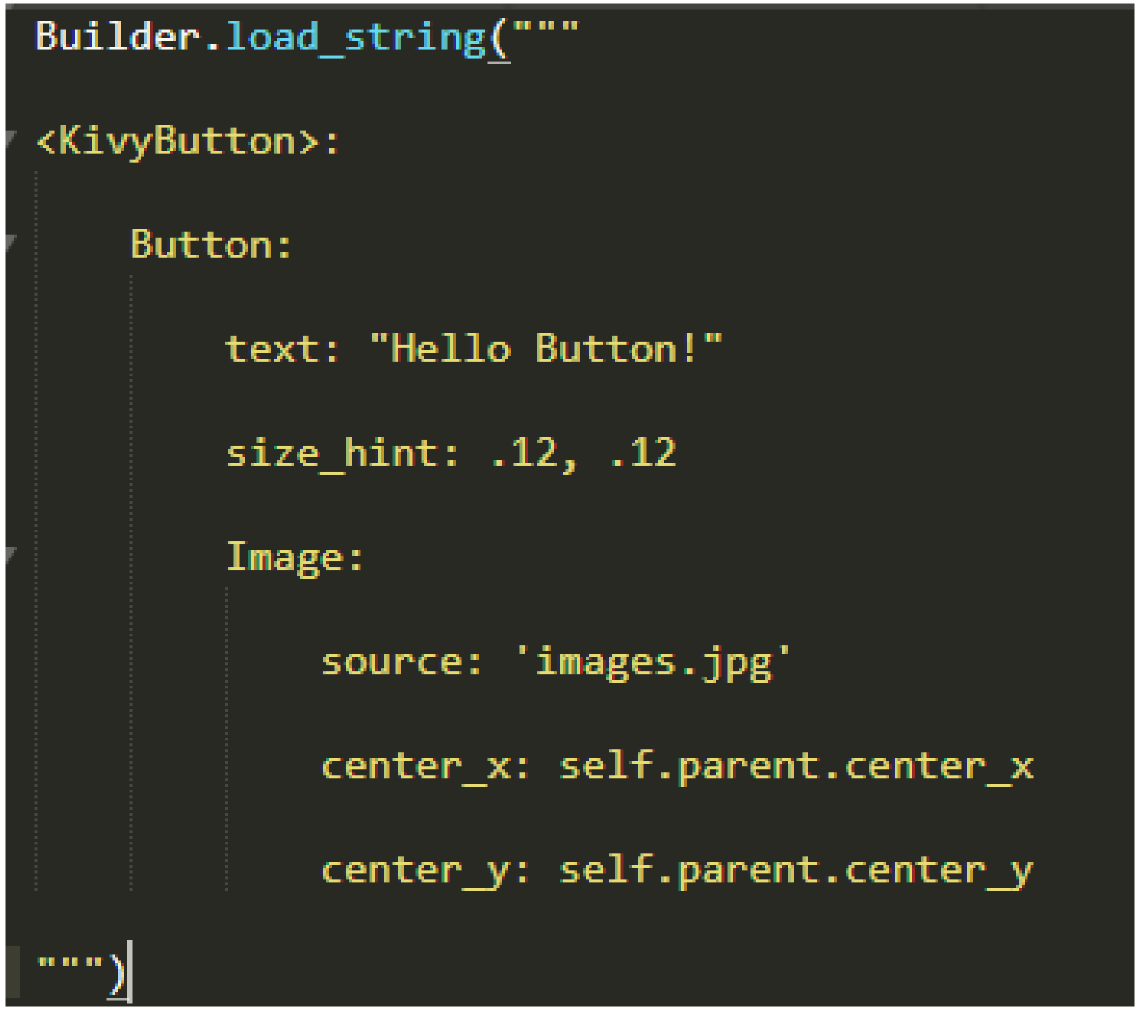
```
from kivy.app import App

from kivy.uix.button import Button

class KivyButton(App):

    def build(self):
        button = Button(text="Welcome to LikeGeeks!", pos=(300,350), size_hint = (.25, .18))
        # button.bind(on_press = self.PressButton)
        return button

    # def PressButton(self, instance):
    #   instance.pos[0] += 10

KivyButton().run()
```

Image in Kivy Button

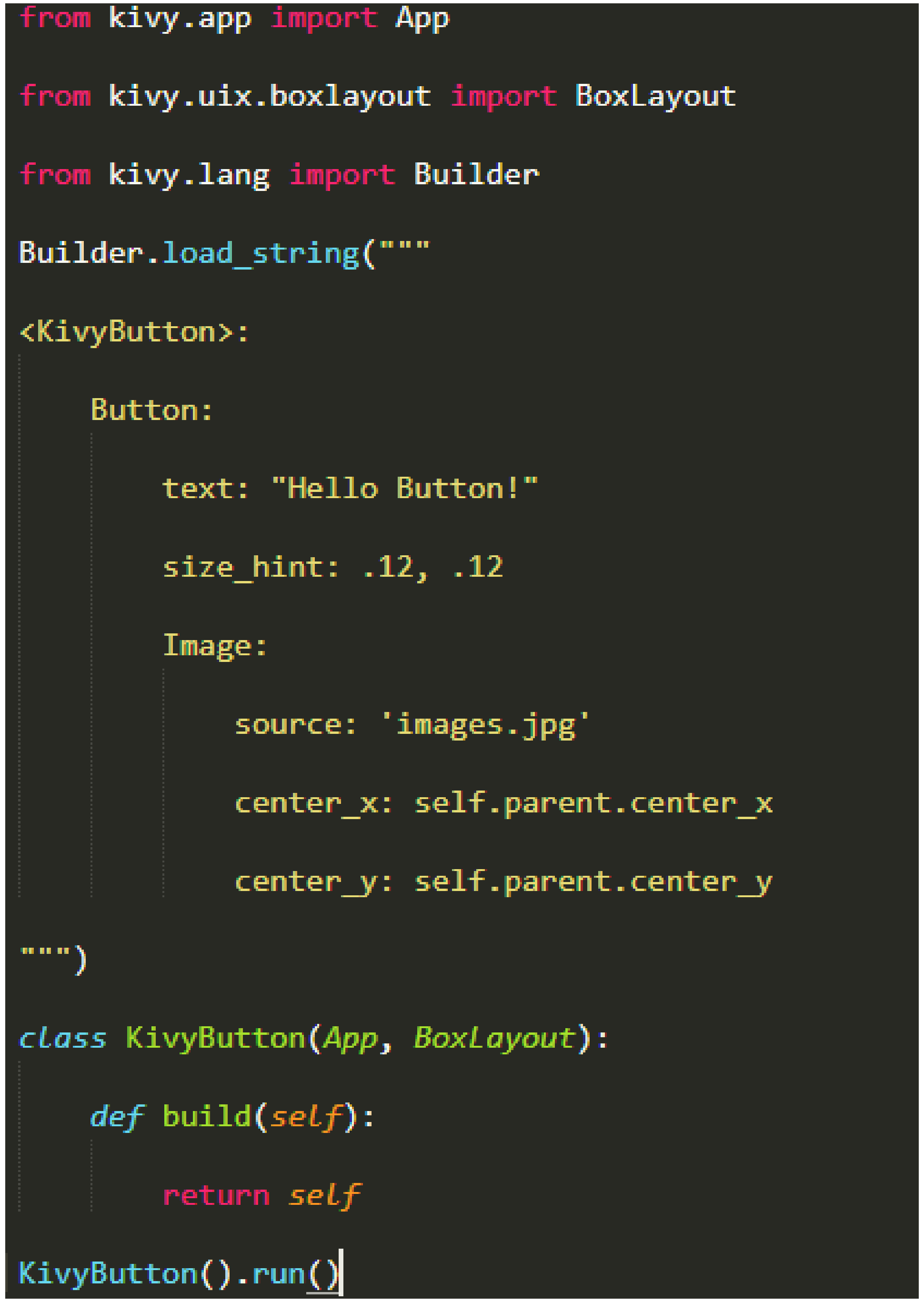
Load Kv string or file



```
Builder.load_string(""" """)
```



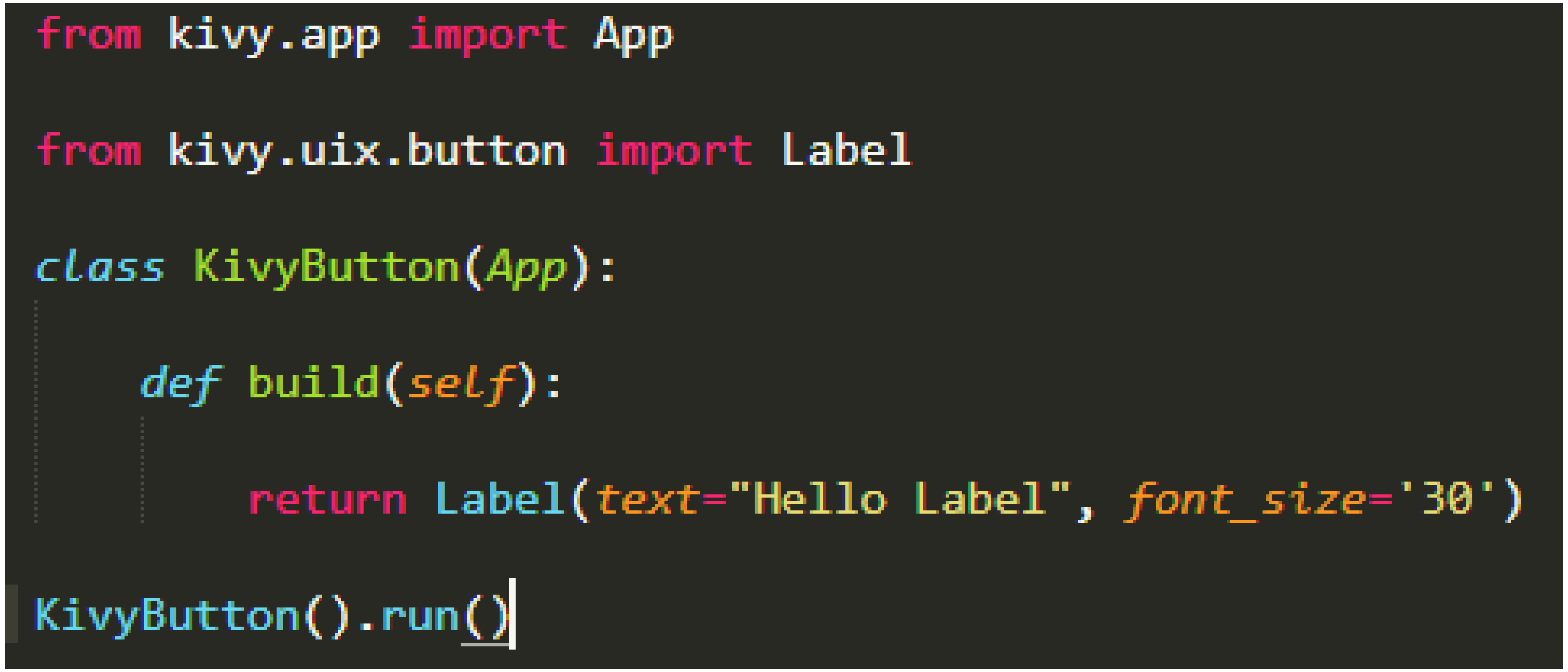
```
Builder.load_string("""
<KivyButton>:
    Button:
        text: "Hello Button!"
        size_hint: .12, .12
        Image:
            source: 'images.jpg'
            center_x: self.parent.center_x
            center_y: self.parent.center_y
""")
```



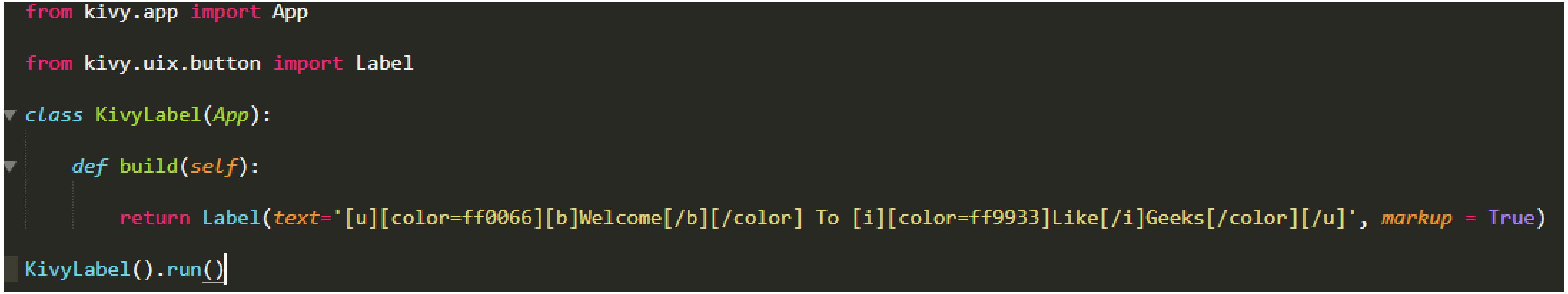
```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.lang import Builder
Builder.load_string("""
<KivyButton>:
    Button:
        text: "Hello Button!"
        size_hint: .12, .12
        Image:
            source: 'images.jpg'
            center_x: self.parent.center_x
            center_y: self.parent.center_y
""")
class KivyButton(App, BoxLayout):
    def build(self):
        return self
KivyButton().run()
```

Kivy Label

Change the font size



```python
from kivy.app import App

from kivy.uix.button import Label

class KivyButton(App):

    def build(self):

        return Label(text="Hello Label", font_size='30')

KivyButton().run()
```

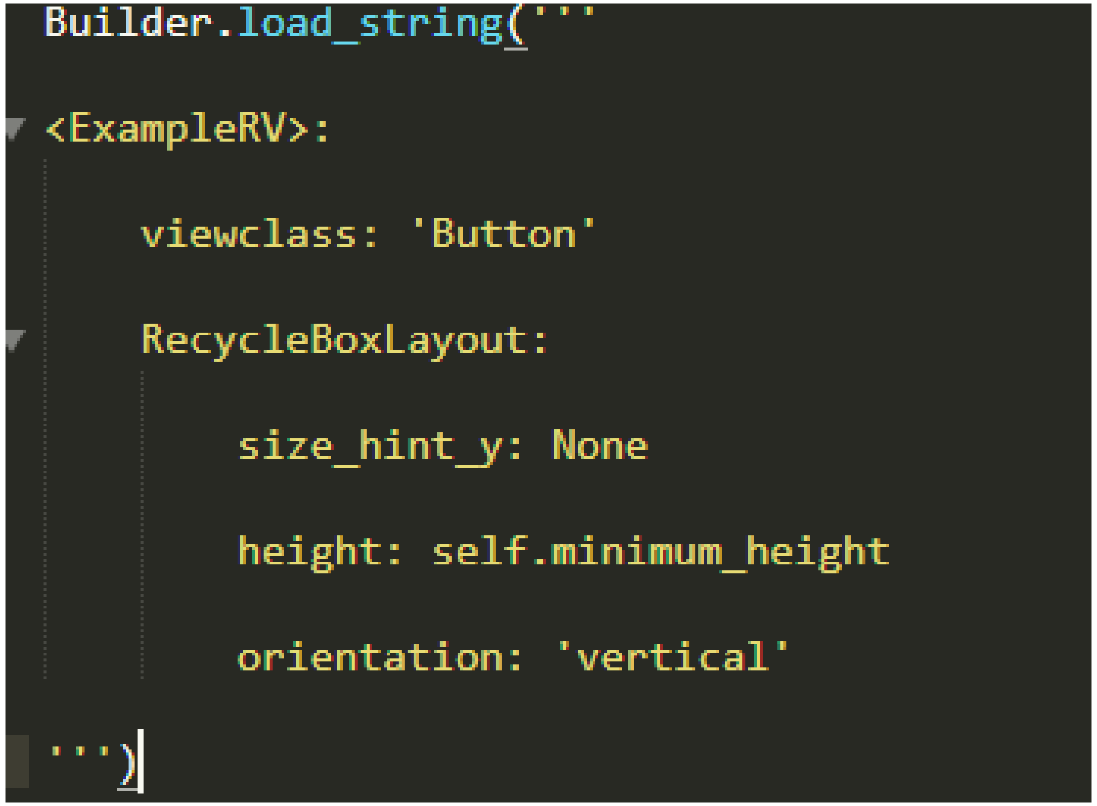


```python
from kivy.app import App

from kivy.uix.button import Label

class KivyLabel(App):

    def build(self):

        return Label(text='[u][color=ff0066][b]Welcome[/b][/color] To [i][color=ff9933]Like[/i]Geeks[/color][/u]', markup = True)

KivyLabel().run()
```

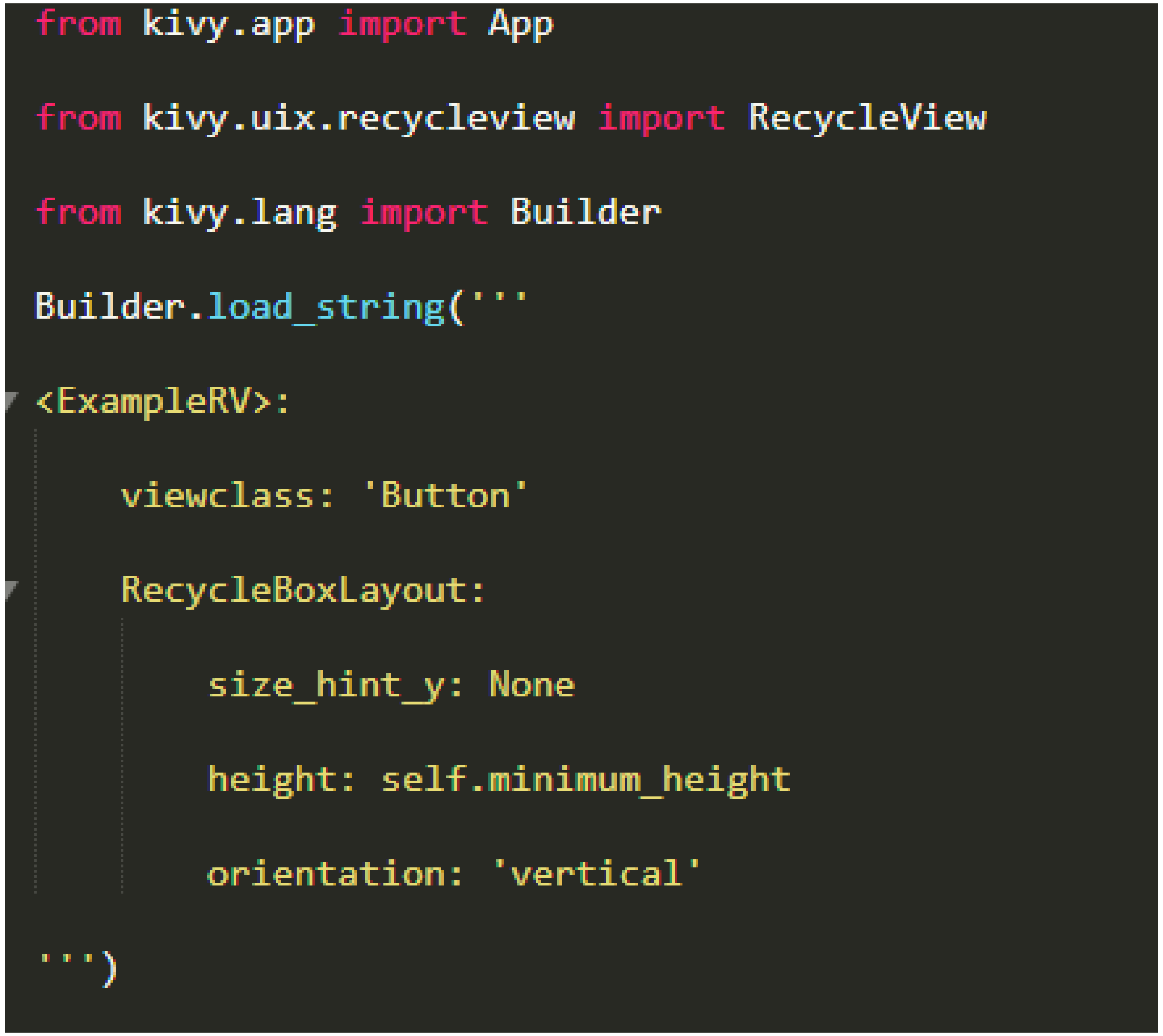
Kivy RecycleView

Before start coding, there are two main concepts to focus on:
1.View Holder which holds a view and helps the recycling.
2.The adapter which is used to adapt the data to display in the list.

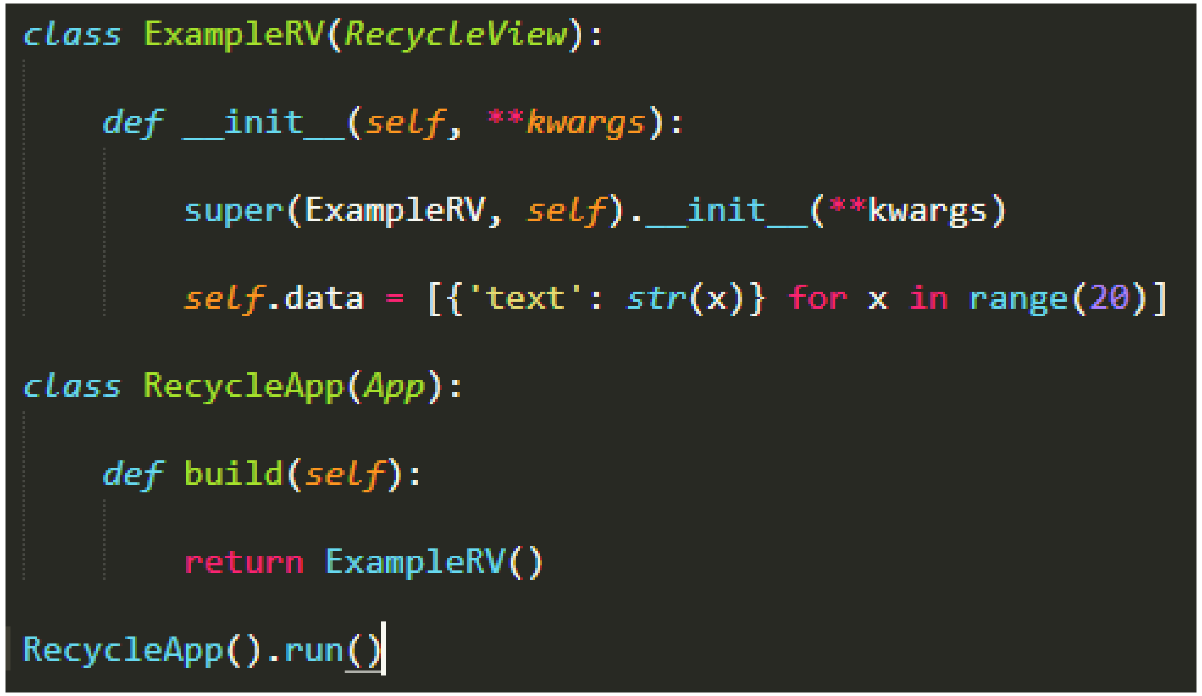
```
from kivy.uix.recycleview import RecycleView
```



```
Builder.load_string('''

<ExampleRV>:

    viewclass: 'Button'

    RecycleBoxLayout:

        size_hint_y: None

        height: self.minimum_height

        orientation: 'vertical'

''')
```

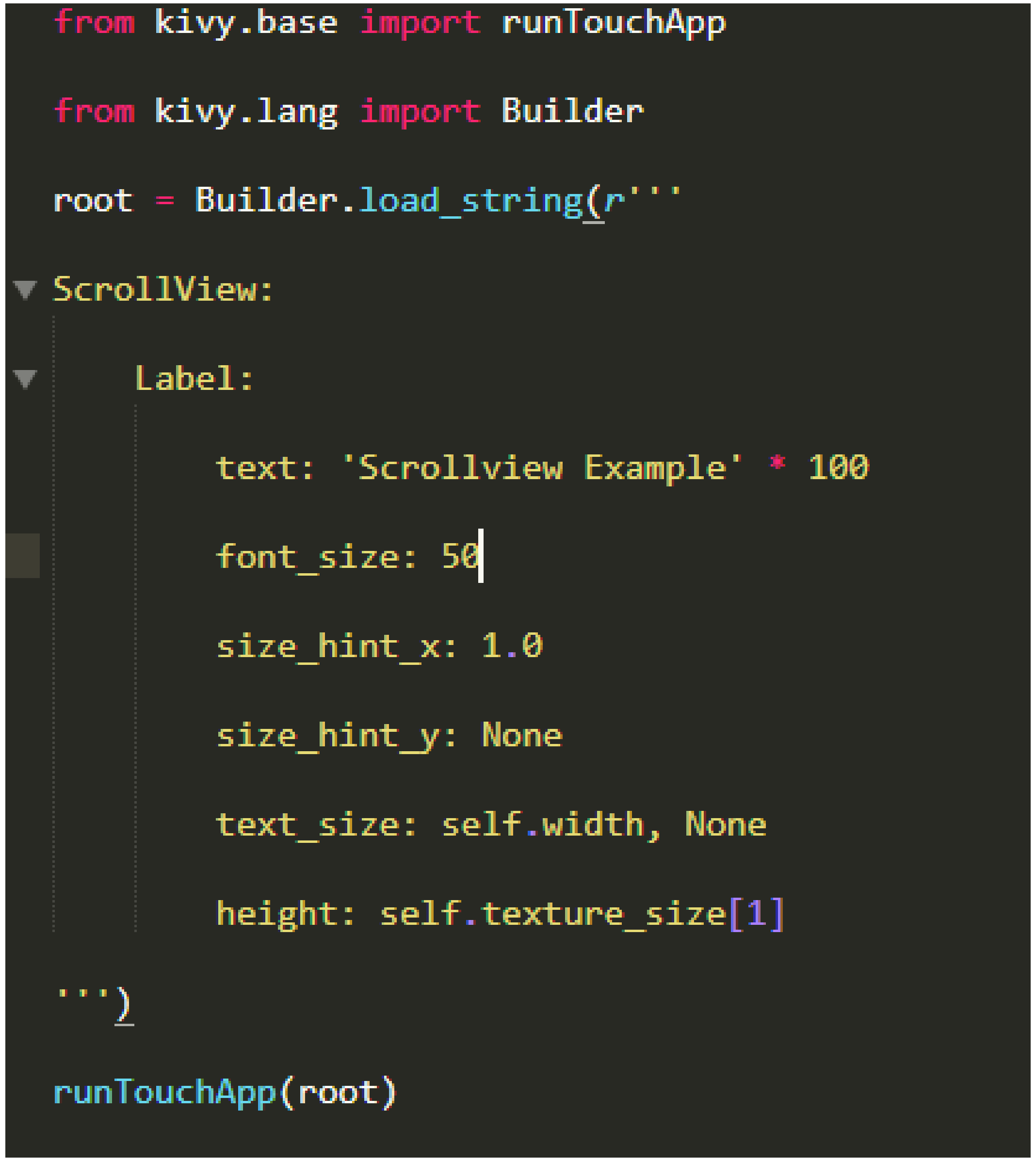


```
from kivy.app import App

from kivy.uix.recycleview import RecycleView

from kivy.lang import Builder

Builder.load_string('''

<ExampleRV>:

    viewclass: 'Button'

    RecycleBoxLayout:

        size_hint_y: None

        height: self.minimum_height

        orientation: 'vertical'

''')
```



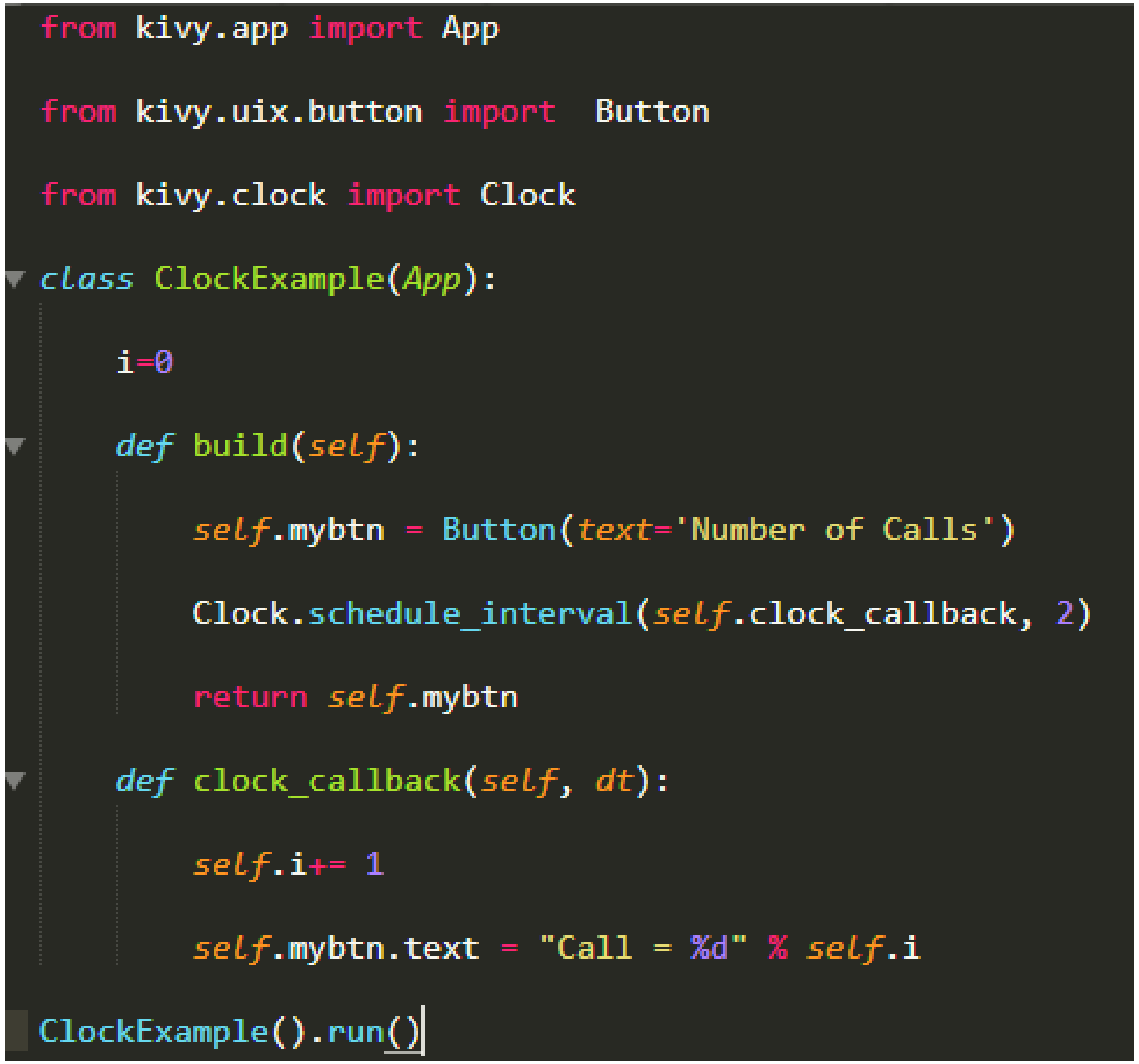
```
class ExampleRV(RecycleView):

    def __init__(self, **kwargs):

        super(ExampleRV, self).__init__(**kwargs)

        self.data = [{'text': str(x)} for x in range(20)]

class RecycleApp(App):

    def build(self):

        return ExampleRV()

RecycleApp().run()
```

Kivy ScrollView



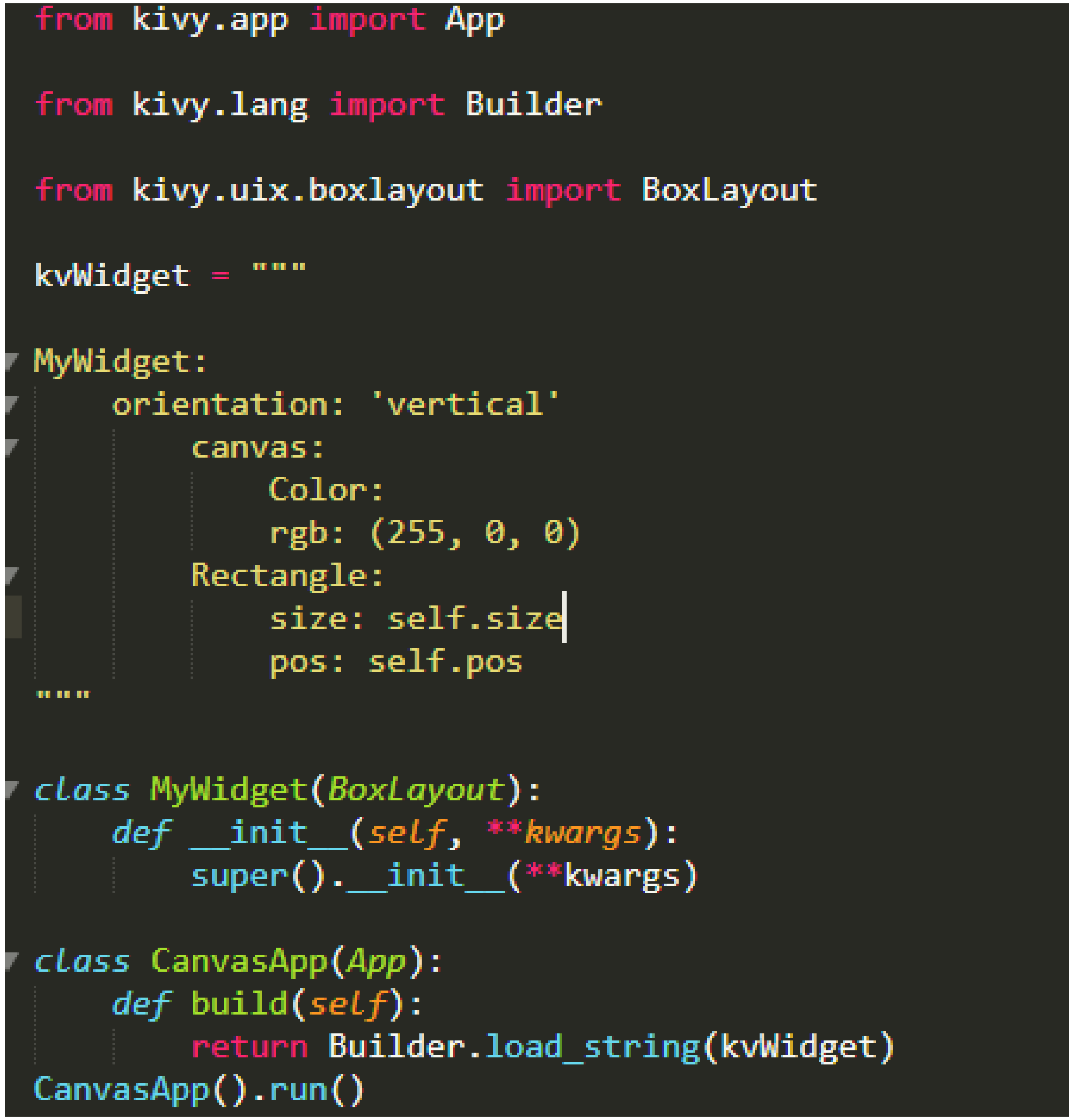
```
from kivy.base import runTouchApp

from kivy.lang import Builder

root = Builder.load_string(r'''

ScrollView:

    Label:

        text: 'Scrollview Example' * 100

        font_size: 50

        size_hint_x: 1.0

        size_hint_y: None

        text_size: self.width, None

        height: self.texture_size[1]

''')

runTouchApp(root)
```

Kivy Clock

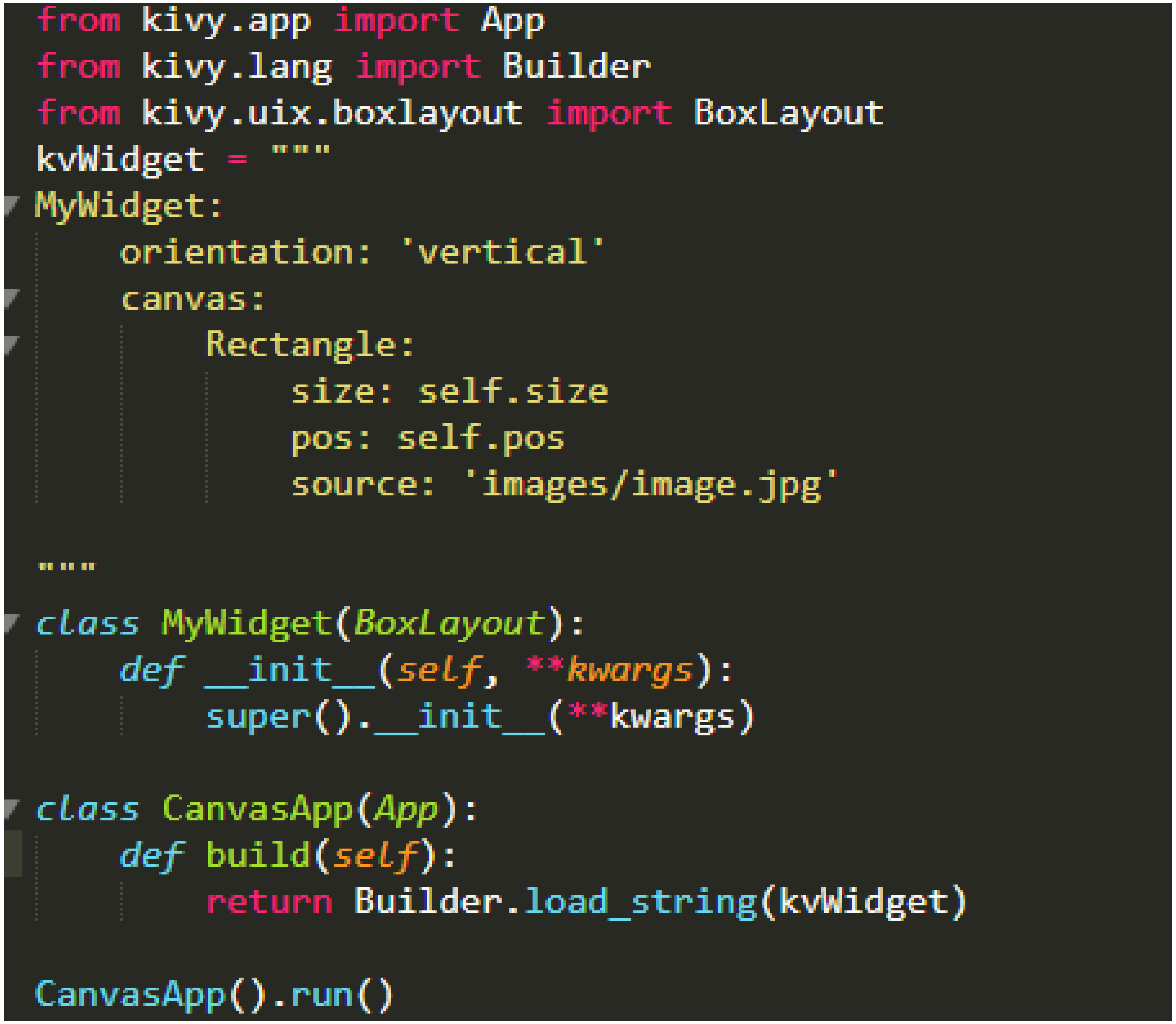


```
from kivy.app import App

from kivy.uix.button import  Button

from kivy.clock import Clock

class ClockExample(App):

    i=0

    def build(self):

        self.mybtn = Button(text='Number of Calls')

        Clock.schedule_interval(self.clock_callback, 2)

        return self.mybtn

    def clock_callback(self, dt):

        self.i+= 1

        self.mybtn.text = "Call = %d" % self.i

ClockExample().run()
```

Kivy Canvas



```python
from kivy.app import App

from kivy.lang import Builder

from kivy.uix.boxlayout import BoxLayout

kvWidget = """

MyWidget:
    orientation: 'vertical'
        canvas:
            Color:
            rgb: (255, 0, 0)
        Rectangle:
            size: self.size
            pos: self.pos
"""

class MyWidget(BoxLayout):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)

class CanvasApp(App):
    def build(self):
        return Builder.load_string(kvWidget)
CanvasApp().run()
```



```python
from kivy.app import App
from kivy.lang import Builder
from kivy.uix.boxlayout import BoxLayout
kvWidget = """
MyWidget:
    orientation: 'vertical'
    canvas:
        Rectangle:
            size: self.size
            pos: self.pos
            source: 'images/image.jpg'

"""
class MyWidget(BoxLayout):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)

class CanvasApp(App):
    def build(self):
        return Builder.load_string(kvWidget)

CanvasApp().run()
```