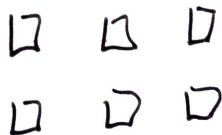


Free code camps - Kivy tutorial

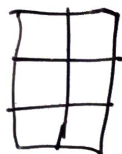
BoxLayout



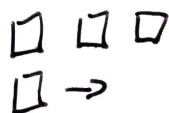
AnchorLayout



GridLayout



StackLayout



ScrollView



PageLayout



FloatLayout

RelativeLayout

ScatterLayout

Button:

Size: "40dp", "40dp"

- 40x40 size of finger
- dp - density independent pixels

Label:

color: 1,0,0,1

rgba 0.0 - 1.0

Layout (python code)

class BoxLayoutExample (BoxLayout):

def __init__(self, **kwargs):

call super

create widget

b1 = Button(text="A")

self.add_widget(b1)

self.orientation = "vertical"
or
"horizontal" (default)

Layout (ku file)

<BoxLayoutExample>:

orientation: "vertical"

Button:

text: "A"

Button:

text: "B"

size_hint: 0.5, 0.5

None, None

pos_hint: {"x": 0.5} #

for vertical: x, center-x, right

for horizontal: y, center-y, top eg None, 0.6

Need to use
size_hint as
a percent
Percent



use this to
override to
have a fixed
size

can mix

use width:
height:

Embed Layout (ku)

BoxLayout:

Button:



BoxLayout:

Button:



AnchorLayout

<AnchorLayoutExample>:

anchor_x: "center" # or left, right

anchor_y: "center" # or bottom, top

... continued

→ Box Layout:

Size-hint: 0, 2, 0, 2


Button:


—


Button:

—

GridLayout

<GridLayoutExample@GridLayout>:  Shortcut, no need to define class in main.py
rows: int
cols: int

MyBoxLayout:  Embed custom layout
Button:

<MyBoxLayout>:  Defining layout configuration
Button...

StackLayout

<StackLayoutExample>:
orientation: "lr-tb"
swap either ↗ ↘

Properties in code
b = Button (size=(dp(w), dp(h)))
↑
from kivy.metrics import dp

padding: (left, top, right, bottom)
spacing: "10dp", "10dp"
 x y

Scrolling

<ScrollView Example @ ScrollView>:

MyLayout:

size_hint: 1, None

height: self.minimum_height

PageLayout

<PageLayout Example @ PageLayout>:

OtherLayout:

Yet Another Layout:

:

widgets

...
Button:

layout
method
↓

on_press: root.on_button_click()

Label:

text: root.my_text

font_name: "assets/fonts/Led.ttf"

font_size: "80dp"

color: 1, 0.5, 1, 1 #rgba

```
class Layout(...):  
    def on_button_click  
    ...
```

```
class Layout(...):  
    my_text =  
        StringProperty(  
# from kivy.properties  
import StringProp
```

ToggleButton:

on_state: root.on_toggle_state(self)

Button:

disabled: not root.enabled

the toggle
button object
↓

```
def .. (self, widget,  
    widget.state  
    "normal", "down"  
    (off), (on))
```

```
class ... = BooleanProp
```

Switch:

on-active: root.on-switch-active(self)

Slider:

min: 0

max: 100

value: 50

orientation: "vertical"

on-value: root.on-slider-value(self)

* Better Syntax ...

Slider:

id: my-slider

Label:

↖ b.ha the value
using it.

text: str(int(my-slider.value))

Progressbar:

Max: 100

value: int

* no minimum,
always 0

TextInput:

text: str

multiline: bool

on-text-validate: root.on-text-validate(self)

Note: Check out Screen Manager
Action bar

Image:

Source: "assets/images/cake.jpg"

allow_stretch: True
Keep_ratio: False

when given space larger than the image

Canvas

<CanvasExample @ AnyWidget>:

Canvas:

Color:

rgb: 1, 0, 0

Line:

circle: (200, 200, 100)

width: 2.

Line:

ellipse: (500, 300, 100, 200)

width: 2.

Rectangle:

pos: 200, 500

Size: 200, 100

Best practice
use
dp(200), dp(100)

Define var in tv

#: set var-name value

...

Rectangle:

pos: self.Center_x - size/2

* *
get container
widget properties

Line:

* points: $(x_1, y_1, x_2, y_2, [x_n, y_n])$

* width: int

* one of

* circle: (center-x, center-y, radius)

* ellipse: (c-x, c-y, w, h)

* rectangle: (x, y, w, h)

Color:

rgb: float, float, float

rgba: float, float, float, float

Line in code

Rectangle

from kivy.graphics.vertex_instructions import Line
"context_instructions import color"

with self.canvas:

Line(points=(100, 100, 200, 100))

Animation

def -- init -- (...):

from kivy.properties

seconds

... Clock.schedule_interval (self.update, 1)

Delay function call

seconds

Clock.schedule_once(func, 3)

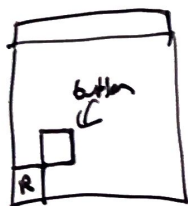
canvas.before: ← draws rect before
default graphics
Rectangle:

* Important
Button:

Canvas:

Rectangle:

pos: 100, 100



Canvas is always relative to (0,0)

RelativeLayout:

Canvas:

Rectangle:

pos: 100, 100



<Example @ widget>:

BoxLayout:

size: root.size

← widgets default to 100x100
to make it go screen-size...

BoxLayout:

widget:

Canvas:

Rectangle:

pos: self.pos

if the widget is not at (0,0)

you need to move the canvas
to the widget's position

OR

← replace widget with
RelativeLayout

class ...

x = NumericProperty(0)

...

def on_x(self, widget, value):

auto-generated on-change method

.kv file
line 17 → x: int

Set window size

from kivy.config import Config

Config.set("graphics", "width", "200")

Config.set("graphics", "height", "200")

Before every Py!

Keypress

init:

self.keyboard = Window.request_keyboard(

self.keyboard_closed,

self

self.keyboard, Keycodes, text, modifiers)

.. .. bind(on_key_down = func)

.. .. bind(on_key_up = func)

self, kb, keycode)

Touch

def on_touch_down(self, touch):

touch.x

touch.y

def on_touch_up(self, touch):

def is_desktop(self):

return platform in ("linux", "win", "mac", "android", "ios", "unknown")

Kivy Common Imports

from

Kivy.app

Kivy.graphics.context_instructions

• vertex_instructions

Kivy.metrics

Kivy.properties

Kivy.uix.anchorlayout

• boxlayout

• button

• gridlayout

• scrollview

• stacklayout

• widget

Kivy.config

Kivy.core.window

Kivy.utils

Kivy.lang

Kivy.core.audio

import

App

Color

Ellipse, Line, Rectangle, ~~Point~~

dp

StringProperty, NumericProperty

Clock

AnchorLayout

BoxLayout

Button

GridLayout

ScrollView

StackLayout

Widget

Config

Window

platform

Builder

SoundLoader

Importing from other files

menu.py

```
class NewWidget(widget):  
    pass
```

menu.kv

```
<NewWidget>:
```

...

import to .kv

```
#import menu menu
```

other.py

```
Builder.load_file("menu.kv")
```


Changing bg colour

Button:

background-normal: ""

background-color: 1, 0.5, 0.5, 0.5

Image of button bg
when not pressed



Sound

```
self.sound = SoundLoader.load("file.wav")
```

```
self.sound.play()
```

```
self.sound.stop()
```