Kivy in python:-

Program code1:-

HELLO WORLD:-

import kivy

from kivy.app import App

from kivy.uix.label import Label

kivy.require('1.11.1')

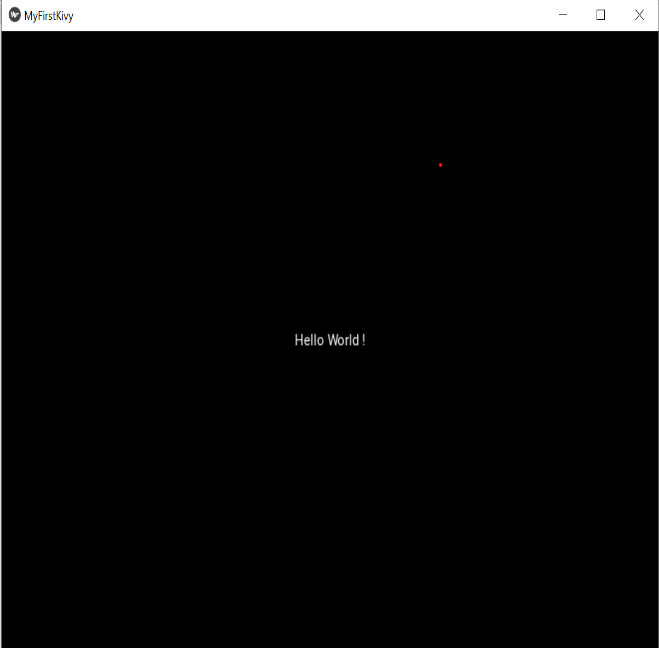
class MyFirstKivyApp(App):

    def build(self):

        return Label(text ="Hello World !")

MyFirstKivyApp().run()

Output:-



Programe 2:-

IMPORTING KIVY FILE .kv

Main file

import kivy

from kivy.app import App

kivy.require('1.11.1')

class kvfileApp(App):

    pass

kv = kvfileApp()

kv.run()

.kv file

Label:

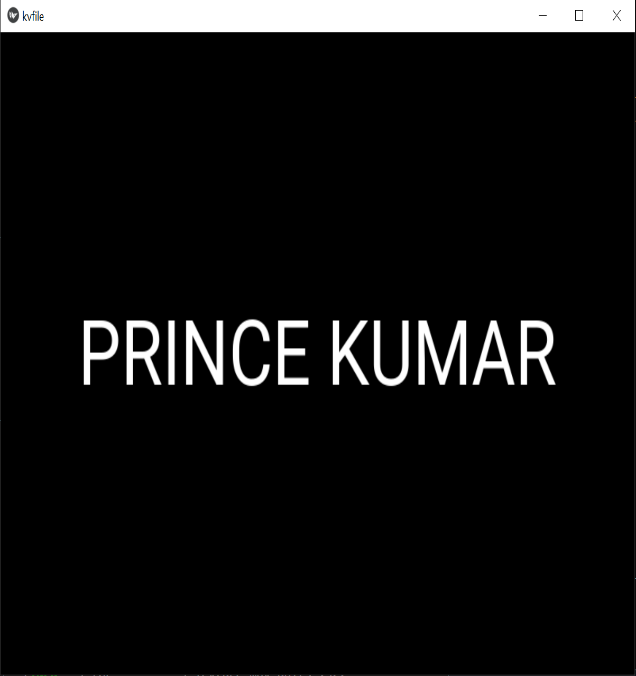
text:

"PRINCE KUMAR"

markup: True

font\_size: '64pt'

Output:-



LAYOUT:-

TYPES OF LAYOUT:-

Layouts are containers used to arrange widgets in a particular manner.

[AnchorLayout](https://kivy.org/doc/stable/api-kivy.uix.anchorlayout.html#module-kivy.uix.anchorlayout):

Widgets can be anchored to the ‘top’, ‘bottom’, ‘left’, ‘right’ or ‘center’.

[BoxLayout](https://kivy.org/doc/stable/api-kivy.uix.boxlayout.html#module-kivy.uix.boxlayout):

Widgets are arranged sequentially, in either a ‘vertical’ or a ‘horizontal’ orientation.

[FloatLayout](https://kivy.org/doc/stable/api-kivy.uix.floatlayout.html#module-kivy.uix.floatlayout):

Widgets are essentially unrestricted.

[RelativeLayout](https://kivy.org/doc/stable/api-kivy.uix.relativelayout.html#module-kivy.uix.relativelayout):

Child widgets are positioned relative to the layout.

[GridLayout](https://kivy.org/doc/stable/api-kivy.uix.gridlayout.html#module-kivy.uix.gridlayout):

Widgets are arranged in a grid defined by the *rows* and *cols* properties.

[PageLayout](https://kivy.org/doc/stable/api-kivy.uix.pagelayout.html#module-kivy.uix.pagelayout):

Used to create simple multi-page layouts, in a way that allows easy flipping from one page to another using borders.

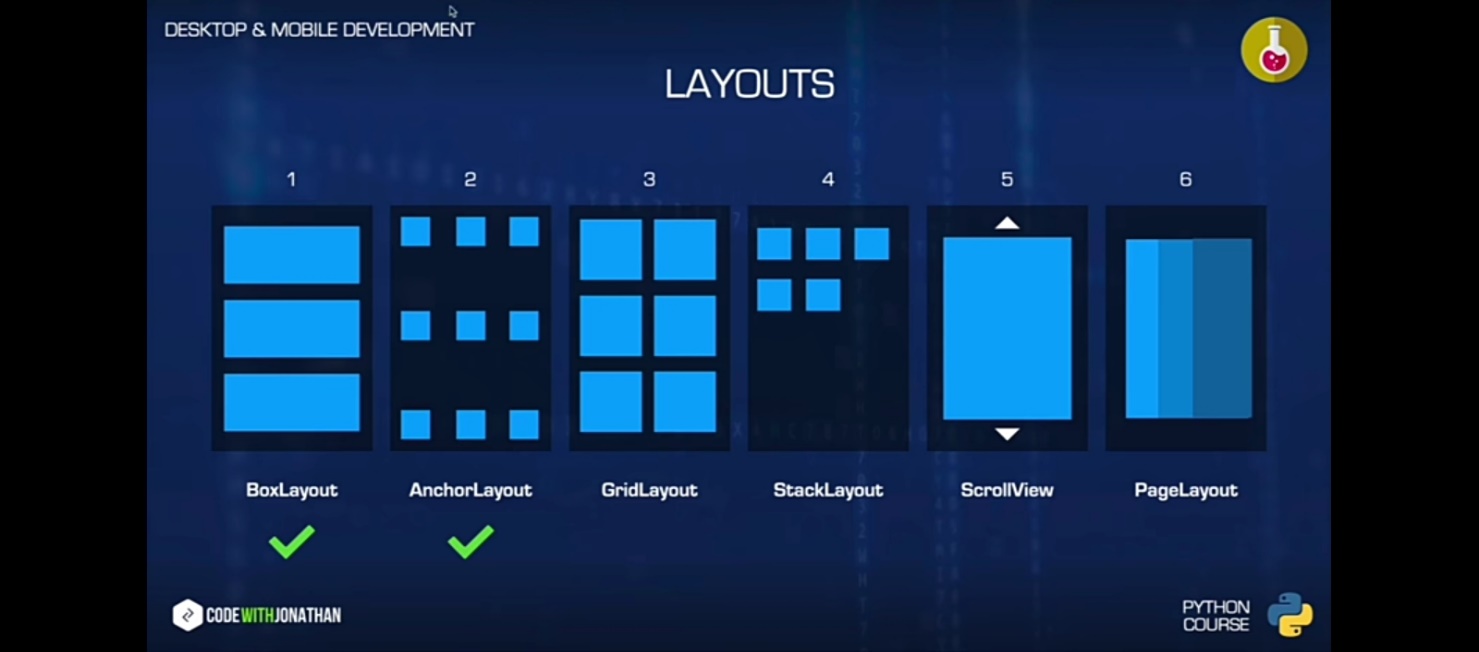
[ScatterLayout](https://kivy.org/doc/stable/api-kivy.uix.scatterlayout.html#module-kivy.uix.scatterlayout):

Widgets are positioned similarly to a RelativeLayout, but they can be translated, rotated and scaled.

[StackLayout](https://kivy.org/doc/stable/api-kivy.uix.stacklayout.html#module-kivy.uix.stacklayout):

Widgets are stacked in a *lr-tb* (left to right then top to bottom) or *tb-lr* order.

Diagram:-



LAYOUT PROGRAME:-

Programe code:-

from kivy.app import App

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.anchorlayout import AnchorLayout

from kivy.uix.gridlayout  import GridLayout

from kivy.uix.stacklayout import StackLayout

from kivy.uix.scrollview import ScrollView

from kivy.uix.pagelayout import PageLayout

from kivy.uix.button import Button

from kivy.uix.widget import Widget

class ScrLayout(ScrollView):

    pass

class StckLayout(StackLayout):

    def \_\_init\_\_(self, \*\*kwargs):

        super().\_\_init\_\_(\*\*kwargs)

        b=Button(text="Z",size\_hint=(.5,.5))

        self.add\_widget(b)

    pass

class GrdLayout(GridLayout):

    pass

class AnrLayout(AnchorLayout):

    pass

class BxLayout(BoxLayout):

    pass

class mainwidget(Widget):

    pass

class princeapp(App):

    pass

princeapp().run()

.Kv file:-

PgLayout:

<PgLayout@PageLayout>

StckLayout:

GrdLayout:

AnrLayout:

BxLayout:

mainwidget:

<ScrLayout@ScrollView>

StckLayout:

size\_hint:1,None

height:12000

<StckLayout>

#left-right top-bottom

orientation:"rl-tb"

padding:("40dp","40dp","40dp","40dp")

spacing:"20dp","20dp"

Button:

text:"A"

size\_hint:.5,.5

Button:

text:"B"

size\_hint:.5,.5

Button:

text:"C"

size\_hint:.5,.5

Button:

text:"D"

size\_hint:.5,.5

Button:

text:"E"

size\_hint:.5,.5

Button:

text:"F"

size\_hint:.5,.5

Button:

text:"G"

size\_hint:.5,.5

Button:

text:"H"

size\_hint:.5,.5

Button:

text:"I"

size\_hint:.5,.5

Button:

text:"J"

size\_hint:.5,.5

Button:

text:"K"

size\_hint:.5,.5

<GrdLayout>

# requires rows and cols

cols:3

rows:3

Button:

text:"A"

size\_hint:.5,1

BxLayout:

Button:

text:"C"

Button:

text:"D"

size\_hint:.5,1

Button:

text:"E"

AnrLayout:

Button:

text:"G"

size\_hint:.5,1

Button:

text:"H"

Button:

text:"I"

<AnrLayout>:

anchor\_x:"center"

# right,left,center

anchor\_y:"top"

# top,bottom,center

BoxLayout:

size\_hint:.2,.2

Button:

text:"A"

Button:

text:"B"

<BxLayout>

orientation:"horizontal"

Button:

text:"A"

size\_hint:None,None

width:"100dp"

height:"68dp"

pos\_hint:{"x":.1}

BoxLayout:

orientation:"vertical"

spacing:"10dp"

Button:

text:"B1"

Button:

text:"B2"

Button:

text:"B3"

Button:

text:"B4"

Button:

text:"C"

size\_hint:None,None

width:"100dp"

height:"70dp"

pos\_hint:{"center\_x":.7}

# for horizontal we use top and down insted of right and left

<mainwidget>

Button:

text: "Hello1"

size:"400dp","200dp"

pos:"100dp","200dp"

color: 0,0,1,1

Label:

text: "Hello2"

size:"100dp","80dp"

pos:"40dp","80dp"

color: 1,0,0,1

Box Layout:-

Main file:-

class BxLayout(BoxLayout):

    pass

.kv file:-

<BxLayout>

orientation:"horizontal"

Button:

text:"A"

size\_hint:None,None

width:"100dp"

height:"68dp"

pos\_hint:{"x":.1}

BoxLayout:

orientation:"vertical"

spacing:"10dp"

Button:

text:"B1"

Button:

text:"B2"

Button:

text:"B3"

Button:

text:"B4"

Button:

text:"C"

size\_hint:None,None

width:"100dp"

height:"70dp"

pos\_hint:{"center\_x":.7}

# for horizontal we use top and down insted of right and left

Anchor layout:-

class AnrLayout(AnchorLayout):

    pass

.kvfile:-

<AnrLayout>:

anchor\_x:"center"

# right,left,center

anchor\_y:"top"

# top,bottom,center

BoxLayout:

size\_hint:.2,.2

Button:

text:"A"

Button:

text:"B"

Grid layout:-

class GrdLayout(GridLayout):

    pass

.kvfile:-

<GrdLayout>

# requires rows and cols

cols:3

rows:3

Button:

text:"A"

size\_hint:.5,1

BxLayout:

Button:

text:"C"

Button:

text:"D"

size\_hint:.5,1

Button:

text:"E"

AnrLayout:

Button:

text:"G"

size\_hint:.5,1

Button:

text:"H"

Button:

text:"I"

Stack Layout:-

class StckLayout(StackLayout):

   def \_\_init\_\_(self, \*\*kwargs):

        super().\_\_init\_\_(\*\*kwargs)

        b=Button(text="Z",size\_hint=(.5,.5))

        self.add\_widget(b)

.kv file:-

<StckLayout>

#left-right top-bottom

orientation:"rl-tb"

padding:("40dp","40dp","40dp","40dp")

spacing:"20dp","20dp"

Button:

text:"A"

size\_hint:.5,.5

Button:

text:"B"

size\_hint:.5,.5

Button:

text:"C"

size\_hint:.5,.5

Button:

text:"D"

size\_hint:.5,.5

Button:

text:"E"

size\_hint:.5,.5

Button:

text:"F"

size\_hint:.5,.5

Button:

text:"G"

size\_hint:.5,.5

Scroll layout:-

class ScrLayout(ScrollView):

    pass

.kvfile:-

<ScrLayout>

StckLayout:

size\_hint:1,None

height:12000

Page Layout:-

class PgLayout(PageLayout):

    pass

.kvfile:-

<PgLayout>

StckLayout:

GrdLayout:

AnrLayout:

BxLayout:

mainwidget:

WIDGETS PROPERTIES:-



Programe code:-

class WidgetsExample(GridLayout):

    count=1

    count\_enabled=BooleanProperty(False)

    my\_text=StringProperty("1")

    slider\_value\_txt=StringProperty("Value")

    def on\_button\_click(self):

        print("Button clicked")

        if self.count\_enabled:

            self.count+=1

            self.my\_text=str(self.count)

    def on\_toggle\_button\_state(self,widget):

        print("toggle state:"+ widget.state)

        if widget.state=="normal":

            widget.text="OFF"

            self.count\_enabled=False

        else:

            widget.text="ON"

            self.count\_enabled=True

    def on\_switch\_active(self,widget):

        print("Switch" + str(widget.active))

    def on\_slider\_value(self,widget):

        print("Slider:"+ str(int(widget.value)))

        self.slider\_value\_txt=str(int(widget.value))

.Kv file:-

WidgetsExample:

<WidgetsExample>

cols:3

ToggleButton:

text:"OFF"

on\_state: root.on\_toggle\_button\_state(self)

size\_hint:None,1

width:"100dp"

Button:

text:"Count"

on\_press: root.on\_button\_click()

disabled: not root.count\_enabled

Label:

text: root.my\_text

font\_size:"80dp"

color:1,.5,1,1

Switch:

on\_active:root.on\_switch\_active(self)

size\_hint:None,1

width:"100dp"

Slider:

min:0

max:100

value:50

on\_value:root.on\_slider\_value(self)

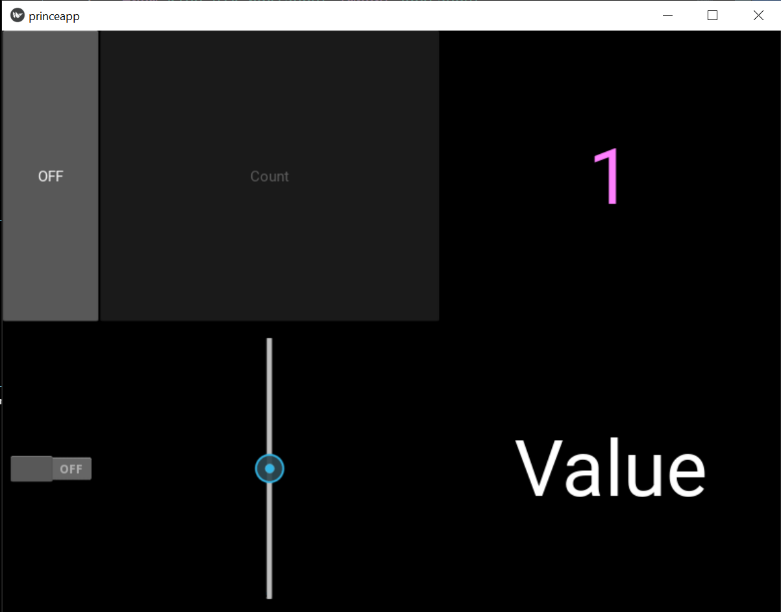
orientation:"vertical"

Label:

text:root.slider\_value\_txt

font\_size:"80dp"

Output:-



Button,toggle,Disabled:-

Programe code:-

class WidgetsExample(GridLayout):

    count=1

    count\_enabled=BooleanProperty(False)

    my\_text=StringProperty("1")

    slider\_value\_txt=StringProperty("Value")

    def on\_button\_click(self):

        print("Button clicked")

        if self.count\_enabled:

            self.count+=1

            self.my\_text=str(self.count)

    def on\_toggle\_button\_state(self,widget):

        print("toggle state:"+ widget.state)

        if widget.state=="normal":

            widget.text="OFF"

            self.count\_enabled=False

        else:

            widget.text="ON"

            self.count\_enabled=True

.kv file:-

WidgetsExample:

<WidgetsExample>

cols:3

ToggleButton:

text:"OFF"

on\_state: root.on\_toggle\_button\_state(self)

size\_hint:None,1

width:"100dp"

Button:

text:"Count"

on\_press: root.on\_button\_click()

disabled: not root.count\_enabled

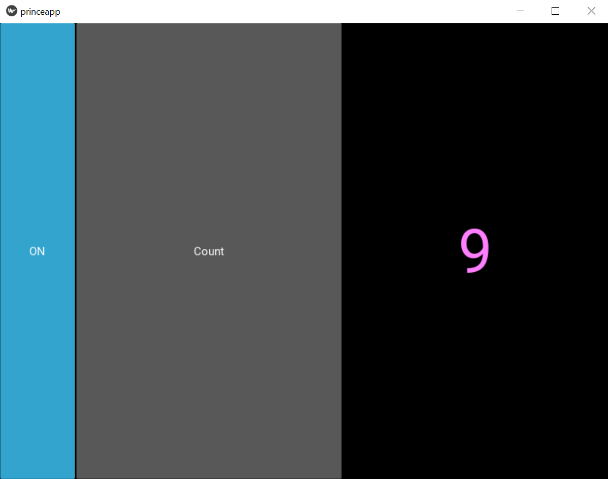
Label:

text: root.my\_text

font\_size:"80dp"

color:1,.5,1,1

Output:-



Switch,Slider,Progress:-

def on\_switch\_active(self,widget):

        print("Switch" + str(widget.active))

    def on\_slider\_value(self,widget):

        print("Slider:"+ str(int(widget.value)))

        self.slider\_value\_txt=str(int(widget.value))

.Kv File:-

Switch:

id:my\_switch

on\_active: root.on\_switch\_active(self)

size\_hint:None,1

width:"100dp"

active:False

Slider:

min:0

max:100

value:50

on\_value: root.on\_slider\_value(self)

orientation:"vertical"

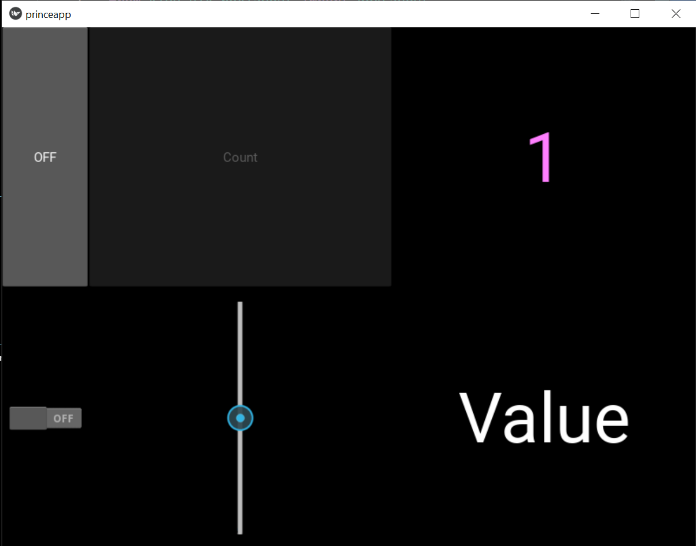
disabled: not my\_switch.active

Label:

text: root.slider\_value\_txt

font\_size:"80dp"

output:-



Adding progress bar:-

from kivy.app import App

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.anchorlayout import AnchorLayout

from kivy.properties import StringProperty, BooleanProperty

from kivy.uix.gridlayout  import GridLayout

from kivy.uix.stacklayout import StackLayout

from kivy.uix.scrollview import ScrollView

from kivy.uix.pagelayout import PageLayout

from kivy.uix.button import Button

from kivy.uix.widget import Widget

class WidgetsExample(GridLayout):

    count=1

    count\_enabled=BooleanProperty(False)

    my\_text=StringProperty("1")

    slider\_value\_txt=StringProperty("Value")

    def on\_button\_click(self):

        print("Button clicked")

        if self.count\_enabled:

            self.count+=1

            self.my\_text=str(self.count)

    def on\_toggle\_button\_state(self,widget):

        print("toggle state:"+ widget.state)

        if widget.state=="normal":

            widget.text="OFF"

            self.count\_enabled=False

        else:

            widget.text="ON"

            self.count\_enabled=True

    def on\_switch\_active(self,widget):

        print("Switch" + str(widget.active))

    #def on\_slider\_value(self,widget):

        #print("Slider:"+ str(int(widget.value)))

        #self.slider\_value\_txt=str(int(widget.value))

class ScrLayout(ScrollView):

    pass

class PgLayout(PageLayout):

    pass

class StckLayout(StackLayout):

   def \_\_init\_\_(self, \*\*kwargs):

        super().\_\_init\_\_(\*\*kwargs)

        b=Button(text="Z",size\_hint=(.5,.5))

        self.add\_widget(b)

class GrdLayout(GridLayout):

    pass

class AnrLayout(AnchorLayout):

    pass

class BxLayout(BoxLayout):

    pass

class mainwidget(Widget):

    pass

class princeapp(App):

    pass

princeapp().run()

.Kv file:-

WidgetsExample:

<WidgetsExample>

cols:3

ToggleButton:

text:"OFF"

on\_state: root.on\_toggle\_button\_state(self)

size\_hint:None,1

width:"100dp"

Button:

text:"Count"

on\_press: root.on\_button\_click()

disabled: not root.count\_enabled

Label:

text: root.my\_text

font\_size:"80dp"

color:1,.5,1,1

Switch:

id:my\_switch

on\_active: root.on\_switch\_active(self)

size\_hint:None,1

width:"100dp"

active:False

Slider:

id:my\_slider

min:0

max:100

value:50

#on\_value: root.on\_slider\_value(self)

orientation:"vertical"

disabled: not my\_switch.active

BoxLayout:

orientation:"vertical"

Label:

text: str(int(my\_slider.value))

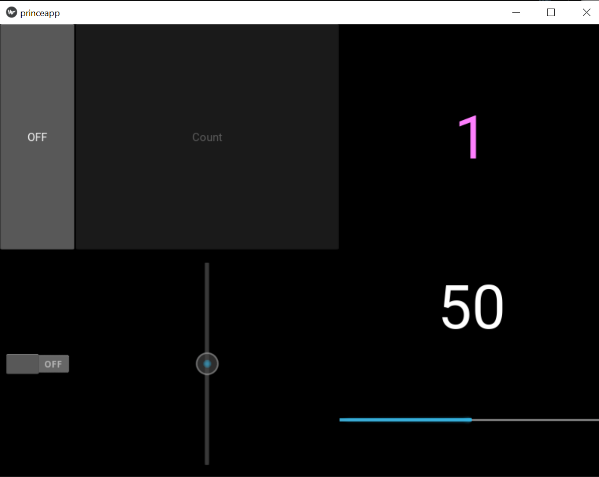
font\_size:"80dp"

ProgressBar:

max:100

value:my\_slider.value

Output:-



TextInput:-

Syntax:- TextInput:

def on\_text\_validate(self,widget):

        self.text\_input\_str=widget.text

.Kv file:-

TextInput:

id:my\_text\_input

size\_hint:None,1

width:"100dp"

text:"Prince"

multiline:False

on\_text\_validate:root.on\_text\_validate(self)

Label:

text:"Your Name is "+my\_text\_input.text

Output:

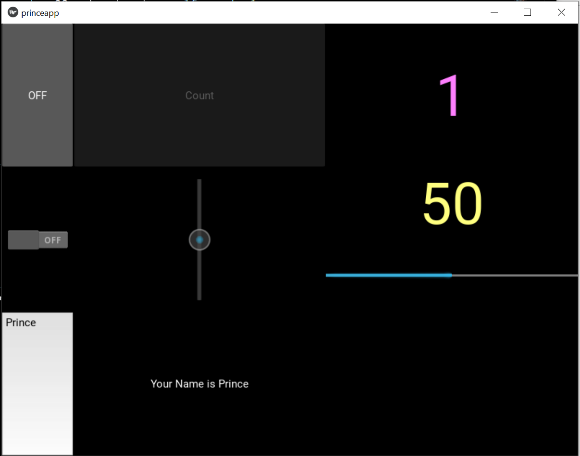


Image:-

.Kv class file:

ImageExample:

<ImageExample@GridLayout>

cols:3

Image:

source:"pri2.jpg"

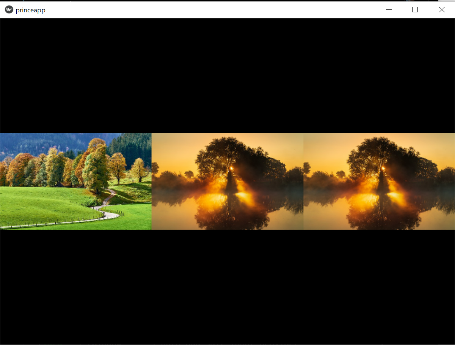
Image:

source:"pri1.jpg"

Image:

source:"pri1.jpg"

Output:-



Strechabel,ratio

.Kv file:

ImageExample:

<ImageExample@GridLayout>

cols:3

Image:

source:"pri2.jpg"

Image:

source:"pri1.jpg"

allow\_stretch:True

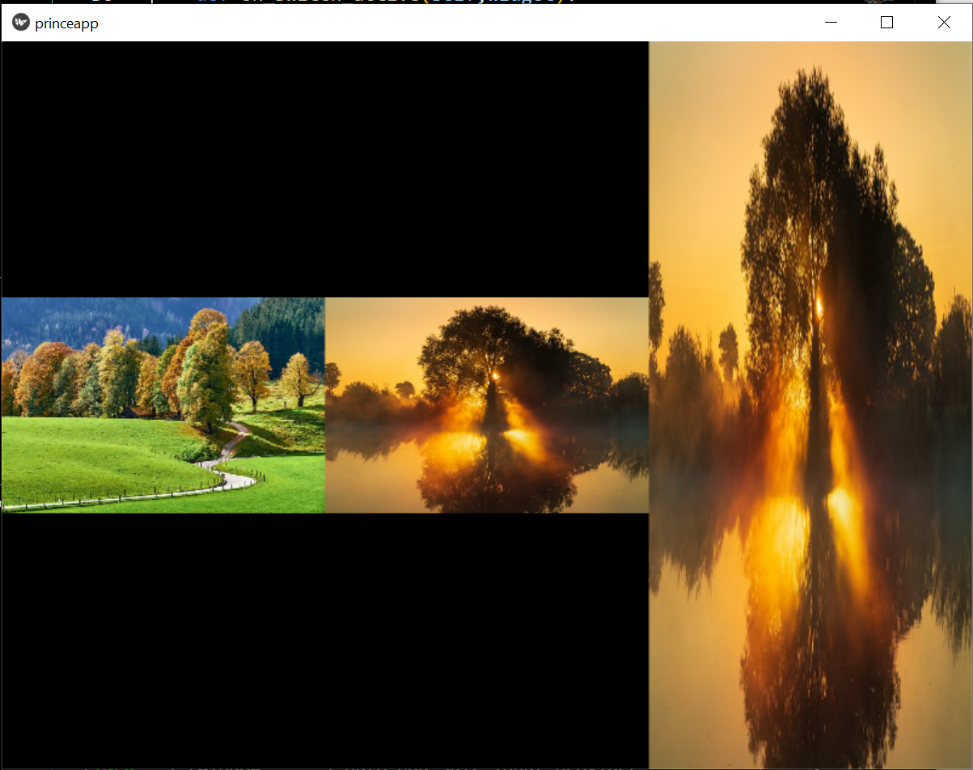
Image:

source:"pri1.jpg"

allow\_stretch:True

keep\_ratio:False

Output:



Canvas:

Line,ellipse,rectangle:

class CanvasExample(Widget):

    pass

.Kv file:-

CanvasExample:

<CanvasExample>

canvas:

Rectangle:

pos:self.center\_x-s/2,self.center\_y-s/2

size:s,s

Ellipse:

pos:200,500

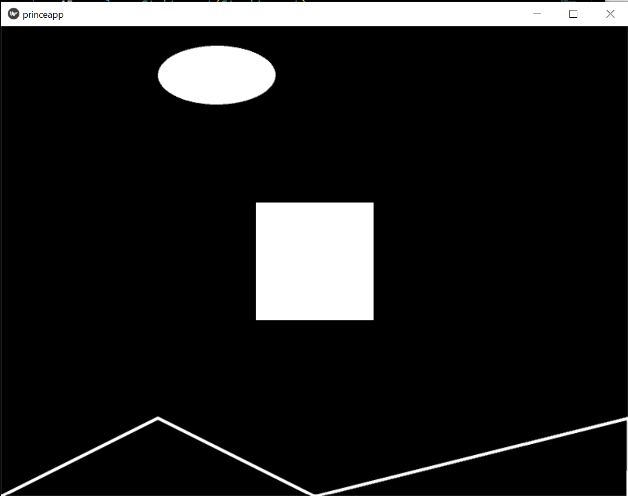
size:s,s/2

Line:

points:(0,0,self.width/4,100,self.width/2,0,self.width+3/4,100,self.width,0)

width:2

output:-



Line property:

class CanvasExample1(Widget):

    pass

.Kv file:

CanvasExample1:

<CanvasExample1>

canvas:

Line:

points:(0,self.height/2,self.width,self.height/2)

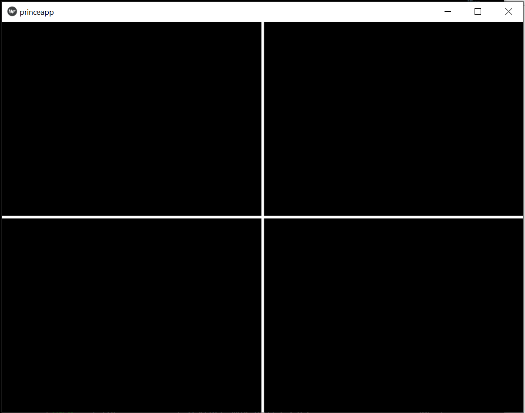
width:2

Line:

points:(self.width/2,0,self.width/2,self.height)

width:2

Output:



Circle,rectangle,eclipse using line property:

class CanvasExample3(Widget):

    pass

.Kv file:

CanvasExample3:

<CanvasExample3>

canvas:

Color:

rgba:1,0,0,.5

Line:

circle:(200,200,100)# center\_x,center\_y,radius

width:2

Line:

ellipse:(500,300,100,200)# center\_x,center\_y,radius\_x,radius\_y

width:4

Color:

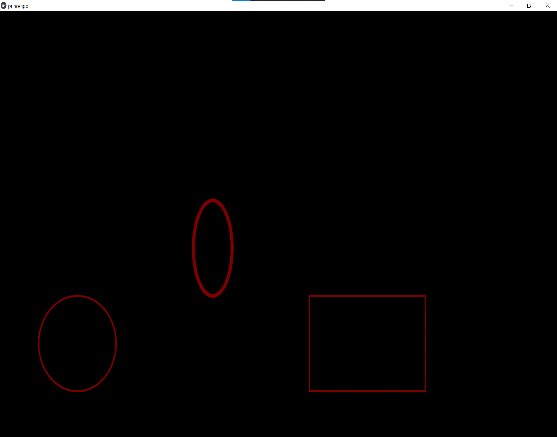
rgba:0,0,.5,1

Line:

rectangle:(800,100,300,200)# x,y,m,n

width:2

Output:



Color property in shape

class CanvasExample3(Widget):

    pass

.Kv file:

CanvasExample3:

<CanvasExample3>

canvas:

Color:

rgba:1,0,0,.5

Line:

circle:(200,200,100)# center\_x,center\_y,radius

width:2

Line:

ellipse:(500,300,100,200)# center\_x,center\_y,radius\_x,radius\_y

width:4

Color:

rgba:0,0,1,1

Line:

rectangle:(800,100,300,200)# x,y,m,n

width:2

Color:

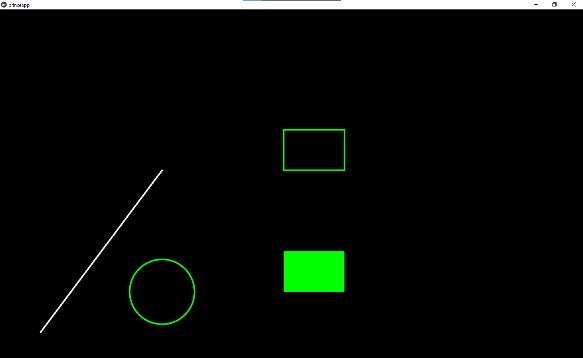
rgba:0,1,0,1

Rectangle:

pos:self.center\_x-s/2,self.center\_y-s/2

size:s,s

output:



Use of animation inside code:

class CanvasExample4(Widget):

    def \_\_init\_\_(self, \*\*kwargs):

        super().\_\_init\_\_(\*\*kwargs)

        with self.canvas:

            Line(points=(100,100,400,500),width=2)

            Color(0,1,0)

            Line(circle=(400,200,80),width=2)

            Line(rectangle=(700,500,150,100),width=2)

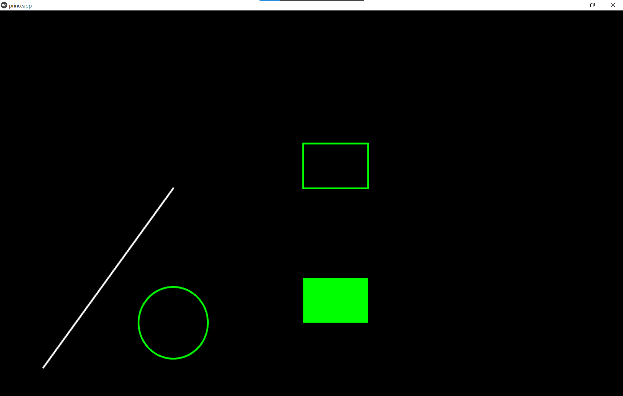
            self.r=Rectangle(pos=(700,200),size=(150,100))

.Kv file:

CanvasExample4:

<CanvasExample4>

Output:



Use of postion property:

class CanvasExample4(Widget):

    def \_\_init\_\_(self, \*\*kwargs):

        super().\_\_init\_\_(\*\*kwargs)

        with self.canvas:

            Line(points=(100,100,400,500),width=2)

            Color(0,1,0)

            Line(circle=(400,200,80),width=2)

            Line(rectangle=(700,500,150,100),width=2)

            self.r=Rectangle(pos=(700,200),size=(150,100))

    def on\_button\_a\_click(self):

        self.r.pos=(100,100)

.Kv file:

CanvasExample4:

<CanvasExample4>

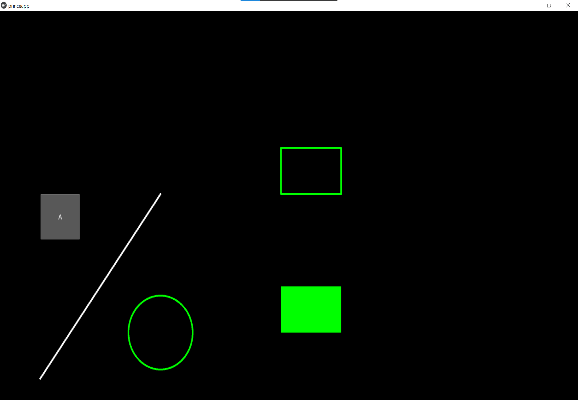
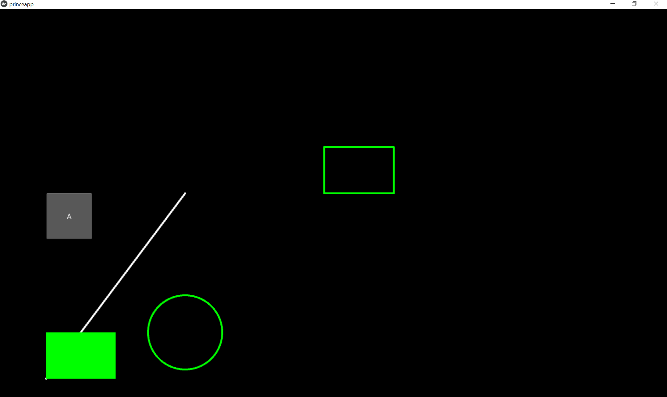
Button:

pos:100,400

text:"A"

on\_press:root.on\_button\_a\_click()

Output:-

Initial: final:

File that we have imported in this file:

from kivy.app import App

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.anchorlayout import AnchorLayout

from kivy.properties import StringProperty, BooleanProperty

from kivy.uix.gridlayout  import GridLayout

from kivy.uix.stacklayout import StackLayout

from kivy.uix.scrollview import ScrollView

from kivy.uix.pagelayout import PageLayout

from kivy.metrics import dp

from kivy.graphics.context\_instructions import Color

from kivy.graphics.vertex\_instructions import Line

from kivy.graphics.vertex\_instructions import Rectangle

from kivy.uix.button import Button

from kivy.uix.widget import Widget

Increment of values:

class CanvasExample4(Widget):

    def \_\_init\_\_(self, \*\*kwargs):

        super().\_\_init\_\_(\*\*kwargs)

        with self.canvas:

            Line(points=(100,100,400,500),width=2)

            Color(0,1,0)

            Line(circle=(400,200,80),width=2)

            Line(rectangle=(700,500,150,100),width=2)

            self.r=Rectangle(pos=(700,200),size=(150,100))

    def on\_button\_a\_click(self):

        x,y=self.r.pos

        x+=dp(10)

        self.r.pos=(x,y)

.Kv file:

CanvasExample4:

<CanvasExample4>

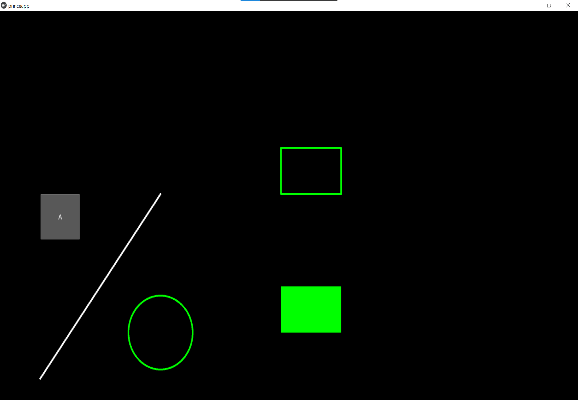
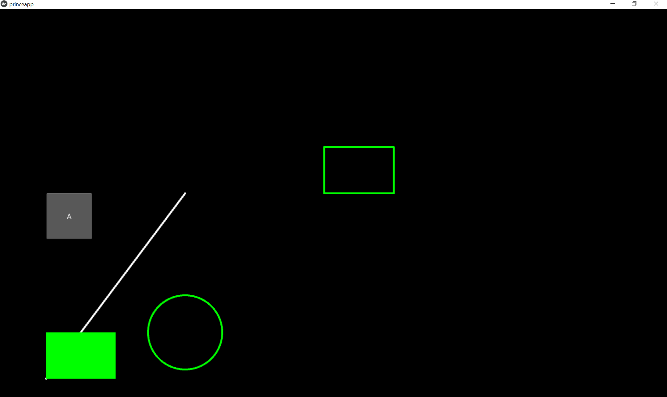
Button:

pos:100,400

text:"A"

on\_press:root.on\_button\_a\_click()

output:

Initial: final:

Using clock inside Moving Shape:

class CanvasExample5(Widget):

    def \_\_init\_\_(self, \*\*kwargs):

        super().\_\_init\_\_(\*\*kwargs)

        self.ball\_size=dp(50)

        self.vx=dp(3)

        self.vy=dp(3)

        with self.canvas:

            self.ball=Ellipse(pos=self.center,size=(self.ball\_size,self.ball\_size))

        Clock.schedule\_interval(self.update,0.2)

    def on\_size(self,\*args):

        self.ball.pos=(self.center\_x-self.ball\_size/2,self.center\_y-self.ball\_size/2)

    def update(self,dt):

        #print("update")

        x,y=self.ball.pos

        x+=self.vx

        y+=self.vy

        self.ball.pos=(x,y)

        if y+self.ball\_size>self.height:

            y=self.height-self.ball\_size

            self.vy=-self.vy

        if x+self.ball\_size>self.width:

            x=self.width-self.ball\_size

            self.vx=-self.vx

        if y<0:

            y=0

            self.vy=-self.vy

        if x<0:

            x=0

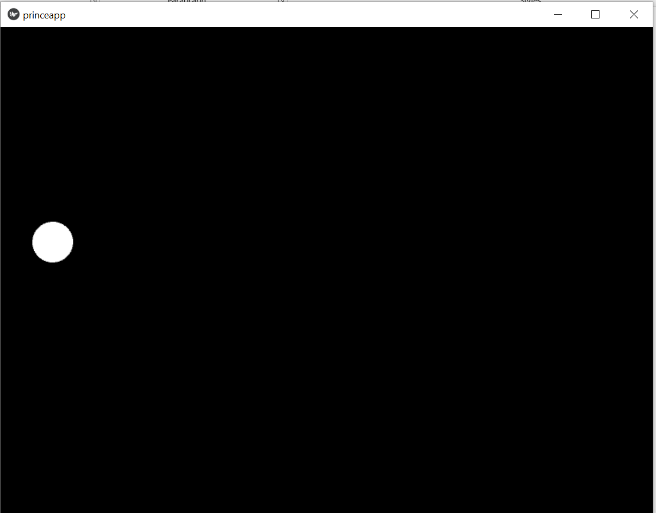
            self.vx=-self.vx

.Kv file:

CanvasExample5:

<CanvasExample5>

Output:



Use of before and after key:

class CanvasExample6(Widget):

    pass

.Kv file:

CanvasExample6:

<CanvasExample6>

Button:

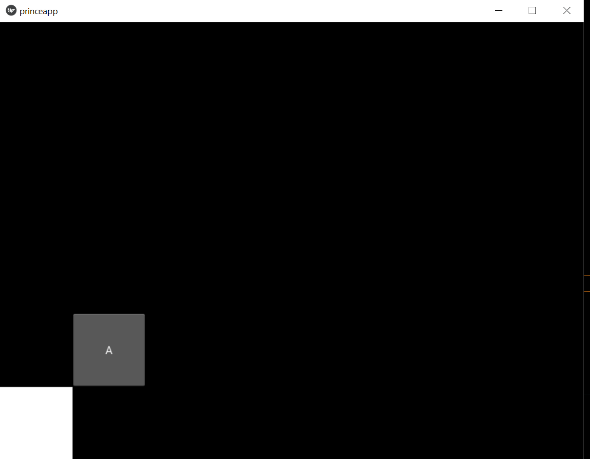
canvas.before:

Rectangle:

text:"A"

pos:100,100

Output:



Nested,Boxlayout,canavs:

class CanvasExample6(Widget):

    pass

.kv file:

CanvasExample6:

<CanvasExample6>

BoxLayout

size:root.size

Widget:

canvas:

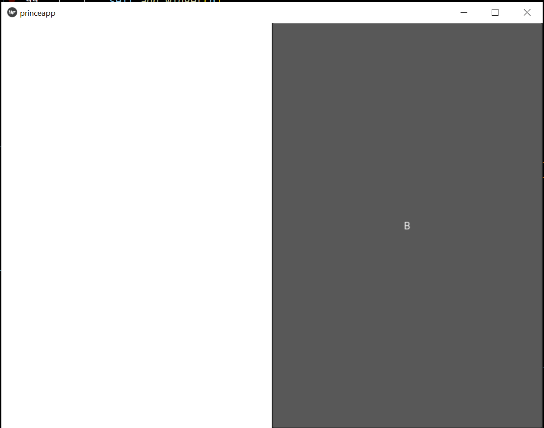
Rectangle:

size:self.size

Button:

text:"B"

Output:



Use of positon in kivi:

class CanvasExample6(Widget):

    pass

.Kv file:

CanvasExample6:

<CanvasExample6>

BoxLayout

size:root.size

Widget:

canvas:

Color:

rgb:0,1,0

Rectangle:

size:self.size

Widget:

canvas:

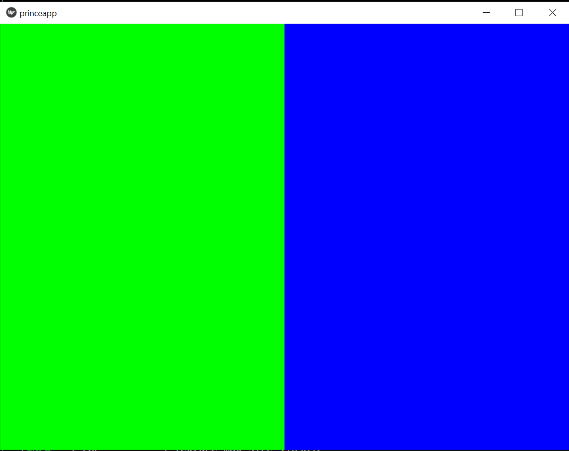
Color:

rgb:0,0,1

Rectangle:

size:self.size

pos:self.pos

output:-

relavtiveLayout:

.Kv file:

CanvasExample6:

<CanvasExample6>

BoxLayout

size:root.size

Widget:

canvas:

Color:

rgb:0,1,0

Rectangle:

size:self.size

RelativeLayout:

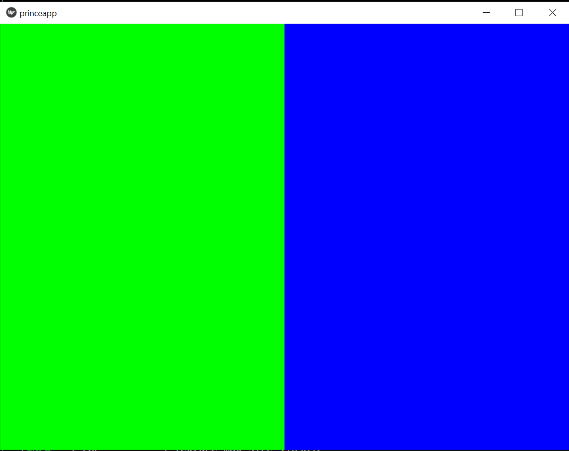
canvas:

Color:

rgb:0,0,1

Rectangle:

size:self.size

output:- :-

Project 1:(create a France flage:)

class CanvasExample6(Widget):

    pass

.Kv file:

CanvasExample6:

<CanvasExample6>

BoxLayout

size:root.size

Widget:

canvas:

Color:

rgb:1,0,0

Rectangle:

size:self.size

RelativeLayout:

canvas:

Color:

rgb:1,1,1

Rectangle:

size:self.size

RelativeLayout:

canvas:

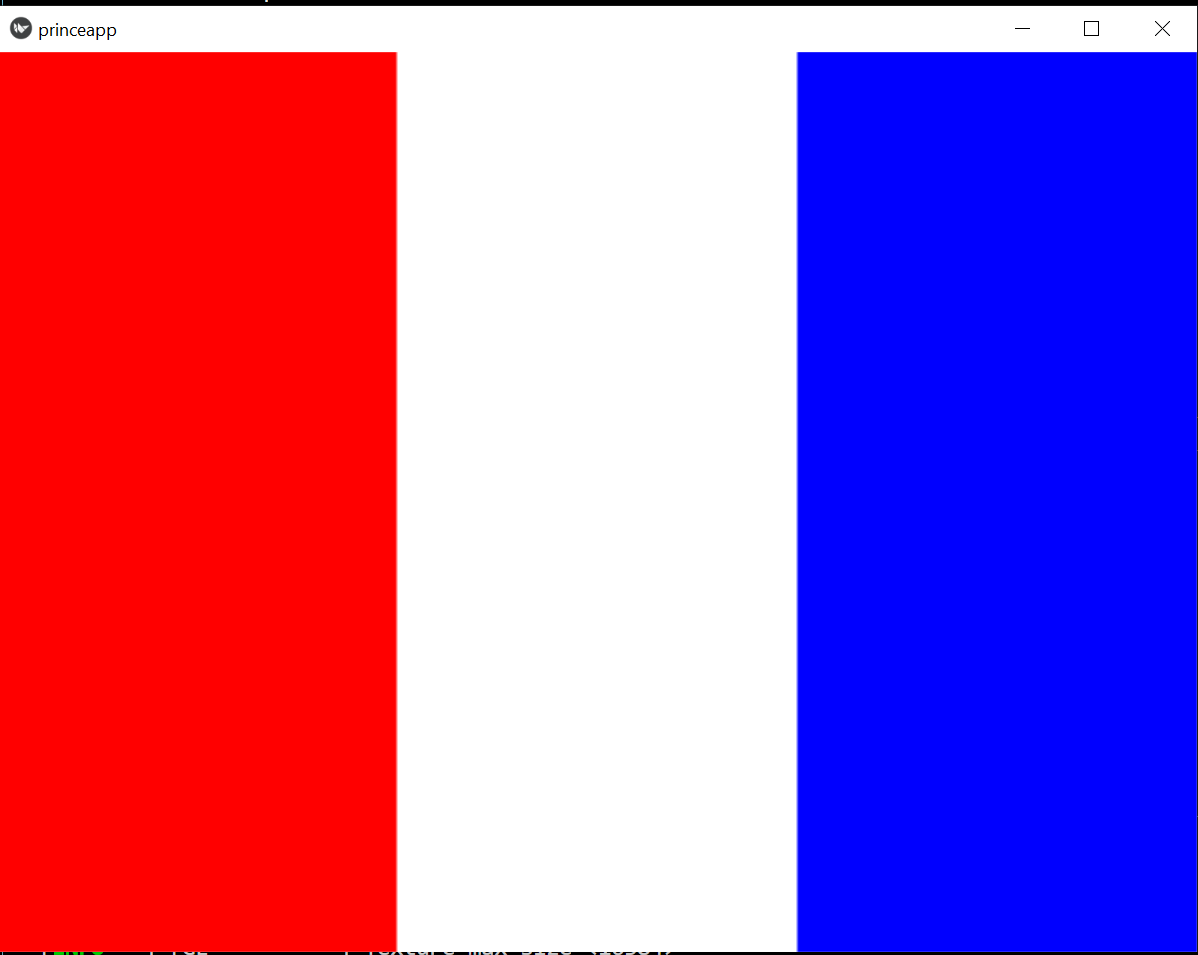
Color:

rgb:0,0,1

Rectangle:

size:self.size

Output:



Co-oridinates in :

from kivy.app import App

from kivy.uix.widget import Widget

from kivy.properties import NumericProperty

class Mainwidget(Widget):

    x=NumericProperty(0)

    y=NumericProperty(0)

    def \_\_init\_\_(self, \*\*kwargs):

        super(Mainwidget,self).\_\_init\_\_(\*\*kwargs)

        print("INIT W:"+ str(self.width) + "W:" + str(self.height))

    def on\_parent(self,widget,parent):

        print("ON PARENT W:" + str(self.width) + "W:" + str(self.height))

    def on\_size(self,\*args):

        print("ON SIZE W:"+ str(self.width)+"H:"+ str(self.height))

        self.x=self.width/2

        self.y=self.height\*0.75

    def on\_x(self,widget,value):

        print("PX:"+ str(value))

    def on\_y(self,widget,value):

        print("PY:"+ str(value))

class GalaxyApp(App):

    pass

GalaxyApp().run()

.KV file:

Mainwidget

OUTPUT:

ON SIZE W:697H:600

PX:348.5

ON SIZE W:698H:600

PX:349.0

[INFO ] [Base ] Leaving application in progress...

ON PARENT W:698W:600

using kivy file use of co-oridenet:

from kivy.app import App

from kivy.uix.widget import Widget

from kivy.properties import NumericProperty

class Mainwidget(Widget):

    x=NumericProperty(0)

    y=NumericProperty(0)

    def \_\_init\_\_(self, \*\*kwargs):

        super(Mainwidget,self).\_\_init\_\_(\*\*kwargs)

        print("INIT W:"+ str(self.width) + "W:" + str(self.height))

    def on\_parent(self,widget,parent):

        print("ON PARENT W:" + str(self.width) + "W:" + str(self.height))

    def on\_size(self,\*args):

        print("ON SIZE W:"+ str(self.width)+"H:"+ str(self.height))

        self.x=self.width/2

        self.y=self.height\*0.75

    def on\_x(self,widget,value):

        print("PX:"+ str(value))

    def on\_y(self,widget,value):

        print("PY:"+ str(value))

class GalaxyApp(App):

    pass

GalaxyApp().run()

.KV file:

Mainwidget

OUTPUT:

ON SIZE W:697H:600

PX:348.5

ON SIZE W:698H:600

PX:349.0

[INFO ] [Base ] Leaving application in progress...

ON PARENT W:698W:600

Constructing vertical points:

from kivy.app import App

from kivy.uix.widget import Widget

from kivy.properties import NumericProperty

from kivy.graphics.context\_instructions import Color

from kivy.graphics.vertex\_instructions import Line

class Mainwidget(Widget):

    x=NumericProperty(0)

    y=NumericProperty(0)

    V\_NB\_LINES=7

    V\_LINES\_SPACING=0.1

    vertical\_lines=[]

    def \_\_init\_\_(self, \*\*kwargs):

        super(Mainwidget,self).\_\_init\_\_(\*\*kwargs)

        #print("INIT W:"+ str(self.width) + "W:" + str(self.height))

        self.init\_vertical\_lines()

    def on\_parent(self,widget,parent):

       # print("ON PARENT W:" + str(self.width) + "W:" + str(self.height))

       pass

    def on\_size(self,\*args):

       # print("ON SIZE W:"+ str(self.width)+"H:"+ str(self.height))

       # self.x=self.width/2

       # self.y=self.height\*0.75

       self.update\_vertical\_lines()

       pass

    def on\_x(self,widget,value):

        # print("PX:"+ str(value))

        pass

    def on\_y(self,widget,value):

        # print("PY:"+ str(value))

        pass

    def init\_vertical\_lines(self):

        with self.canvas:

            Color(1,1,1)

            #self.line=Line(points=[100,0,100,0])

            for i in range(0,self.V\_NB\_LINES):

                self.vertical\_lines.append(Line())

    def update\_vertical\_lines(self):

        central\_line\_x=int(self.width/2)

        spacing=self.V\_LINES\_SPACING\*self.width

        offset=-int(self.V\_NB\_LINES/2)

        for i in range(0,self.V\_NB\_LINES):

            line\_x=int(central\_line\_x+offset\*spacing)

            self.vertical\_lines[i].points=[line\_x,0,line\_x,self.height]

            offset+=1

        #self.line.points=[center\_x,0,center\_x,100]

class GalaxyApp(App):

    pass

GalaxyApp().run()

Output:

