

 免费电子书

学习

kivy

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#kivy

.....	1
<b>1: kivy</b> .....	<b>2</b>
.....	2
Examples.....	2
.....	2
.....	<b>2</b>
.....	3
.....	3
.....	4
kivy.....	5
Kivy.....	6
RecycleView.....	6
.....	7
.kv.....	9
<b>2:</b> .....	<b>12</b>
.....	12
.....	<b>12</b>
Examples.....	12
.....	12
.....	13
<b>3:</b> .....	<b>15</b>
Examples.....	15
.....	15
.....	<b>18</b>

---

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [kivy](#)

It is an unofficial and free kivy ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official kivy.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

# 1: kivy

KivyPython。 LinuxWindowsOS XAndroidiOSKivy。

OpenGL ES 2。

Kivykvlangpythonpython。 Pythonkvlang。

Kivy。

## Examples

Kivy

python。

```
python -m pip install --upgrade pip wheel setuptools
```

。

```
python -m pip install docutils pygments pypiwin32 kivy.deps.sdl2 kivy.deps.glew
```

KivyGStreamer。

```
python -m pip install kivy.deps.gstreamer --extra-index-url  
https://kivy.org/downloads/packages/simple/
```

<python>python.exe。

1. wheelKivycython。 Python。

Kivypypi。

```
python -m pip install kivy
```

。 python。 。

```
python -m pip install C:\Kivy-1.9.1.dev-cp27-none-win_amd64.whl
```

2. Kivy。

<python>\Lib\distutils\distutils.cfg。

```
[build]
```

```
compiler = mingw32
```

◦ mingwpy ◦ gcc.exe<python>\Scripts ◦

```
python -m pip install -i https://pypi.anaconda.org/carlkl/simple mingwpy
```

## Kivy◦

```
set USE_SDL2=1  
set USE_GSTREAMER=1
```

◦

```
python -m pip install cython kivy.deps.glew_dev kivy.deps.sdl2_dev  
python -m pip install kivy.deps.gstreamer_dev --extra-index-url  
https://kivy.org/downloads/packages/simple/
```

## Paths Kivy◦

```
python -m pip install C:\master.zip  
python -m pip install https://github.com/kivy/kivy/archive/master.zip
```

## Kivy◦ PATH◦

```
set PATH=<python>\Tools;<python>\Scripts;<python>\share\sdl2\bin;%PATH%
```

## Python IDLE IDE<python>\Lib\idlelib; ◦ idleIDLEKivy◦

<python> .bat

```
set PATH=%~dp0;%~dp0Tools;%~dp0Scripts;%~dp0share\sdl2\bin;%~dp0Lib\idlelib;%PATH%  
cmd.exe
```

Kivycmd.exepython <filename>.py

## Ubuntu

kivyubuntukivy

ppa

```
sudo add-apt-repository ppa:kivy-team/kivy
```

kivy

```
sudo apt-get install python-kivy
```

## kivy

```
sudo apt-get install python-kivy-example
```

### 21. ◦

```
from kivy.app import App
from kivy.graphics import Ellipse, Line
from kivy.uix.boxlayout import BoxLayout

class CustomLayout(BoxLayout):

    def __init__(self, **kwargs):
        super(CustomLayout, self).__init__(**kwargs)

        self.canvas_edge = {}
        self.canvas_nodes = {}
        self.nodesize = [25, 25]

        self.grabbed = {}

        #declare a canvas
        with self.canvas.after:
            pass

        self.define_nodes()
        self.canvas.add(self.canvas_nodes[0])
        self.canvas.add(self.canvas_nodes[1])
        self.define_edge()
        self.canvas.add(self.canvas_edge)

    def define_nodes(self):
        """define all the node canvas elements as a list"""

        self.canvas_nodes[0] = Ellipse(
            size = self.nodesize,
            pos = [100,100]
        )

        self.canvas_nodes[1] = Ellipse(
            size = self.nodesize,
            pos = [200,200]
        )

    def define_edge(self):
        """define an edge canvas elements"""

        self.canvas_edge = Line(
            points = [
                self.canvas_nodes[0].pos[0] + self.nodesize[0] / 2,
                self.canvas_nodes[0].pos[1] + self.nodesize[1] / 2,
                self.canvas_nodes[1].pos[0] + self.nodesize[0] / 2,
                self.canvas_nodes[1].pos[1] + self.nodesize[1] / 2
            ],
            joint = 'round',
            cap = 'round',
```

```

        width = 3
    )

    def on_touch_down(self, touch):

        for key, value in self.canvas_nodes.items():
            if (value.pos[0] - self.nodesize[0]) <= touch.pos[0] <= (value.pos[0] +
self.nodesize[0]):
                if (value.pos[1] - self.nodesize[1]) <= touch.pos[1] <= (value.pos[1] +
self.nodesize[1]):
                    touch.grab(self)
                    self.grabbed = self.canvas_nodes[key]
                    return True

    def on_touch_move(self, touch):

        if touch.grab_current is self:
            self.grabbed.pos = [touch.pos[0] - self.nodesize[0] / 2, touch.pos[1] -
self.nodesize[1] / 2]
            self.canvas.clear()
            self.canvas.add(self.canvas_nodes[0])
            self.canvas.add(self.canvas_nodes[1])
            self.define_edge()
            self.canvas.add(self.canvas_edge)
        else:
            # it's a normal touch
            pass

    def on_touch_up(self, touch):
        if touch.grab_current is self:
            # I receive my grabbed touch, I must ungrab it!
            touch.ungrab(self)
        else:
            # it's a normal touch
            pass

class MainApp(App):

    def build(self):
        root = CustomLayout()
        return root

if __name__ == '__main__':
    MainApp().run()

```

**kivy**。

kivy“hello world”。iosandroidmain.pybuildozer。

```

from kivy.app import App
from kivy.uix.label import Label
from kivy.lang import Builder

Builder.load_string('''
<SimpleLabel>:
    text: 'Hello World'
''')

```

```

class SimpleLabel(Label):
    pass

class SampleApp(App):
    def build(self):
        return SimpleLabel()

if __name__ == "__main__":
    SampleApp().run()

```

## Kivy。

## Kivy。

```

from kivy.app import App
from kivy.uix.popup import Popup
from kivy.lang import Builder
from kivy.uix.button import Button

Builder.load_string('''
<SimpleButton>:
    on_press: self.fire_popup()
<SimplePopup>:
    id:pop
    size_hint: .4, .4
    auto_dismiss: False
    title: 'Hello world!!'
    Button:
        text: 'Click here to dismiss'
        on_press: pop.dismiss()
''')

class SimplePopup(Popup):
    pass

class SimpleButton(Button):
    text = "Fire Popup !"
    def fire_popup(self):
        pops=SimplePopup()
        pops.open()

class SampleApp(App):
    def build(self):
        return SimpleButton()

SampleApp().run()

```

## RecycleView

```

from kivy.app import App
from kivy.lang import Builder
from kivy.uix.button import Button

```

```

items = [
    {"color":(1, 1, 1, 1), "font_size": "20sp", "text": "white",      "input_data":
["some","random","data"]},
    {"color":(.5,1, 1, 1), "font_size": "30sp", "text": "lightblue", "input_data": [1,6,3]},
    {"color":(.5,.5,1, 1), "font_size": "40sp", "text": "blue",      "input_data": [64,16,9]},
    {"color":(.5,.5,.5,1), "font_size": "70sp", "text": "gray",      "input_data":
[8766,13,6]},
    {"color":(1,.5,.5, 1), "font_size": "60sp", "text": "orange",    "input_data": [9,4,6]},
    {"color":(1, 1,.5, 1), "font_size": "50sp", "text": "yellow",    "input_data":
[852,958,123]}
]

class MyButton(Button):

    def print_data(self, data):
        print(data)

KV = '''

<MyButton>:
    on_release:
        root.print_data(self.input_data)

RecycleView:
    data: []
    viewclass: 'MyButton'
    RecycleBoxLayout:
        default_size_hint: 1, None
        orientation: 'vertical'

'''

class Test(App):
    def build(self):
        root = Builder.load_string(KV)
        root.data = [item for item in items]
        return root

Test().run()

```

## kivy

```

from kivy.app import App

class TutorialApp(App):
    def build(self):
        return
TutorialApp().run()

```

13.

## 1Hello World

```

from kivy.app import App

```

```

from kivy.uix.button import Button
class TutorialApp(App):
    def build(self):
        return Button(text="Hello World!")
TutorialApp().run()

```

## 2+

```

from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from kivy.uix.button import Button

class TutorialApp(App):
    def build(self):
        mylayout = BoxLayout(orientation="vertical")
        mylabel = Label(text= "My App")
        mybutton =Button(text="Click me!")
        mylayout.add_widget(mylabel)
        mybutton.bind(on_press= lambda a:print(mylabel.text))
        mylayout.add_widget(mybutton)
        return mylayout
TutorialApp().run()

```

## 3+“”

```

from kivy.app import App
from kivy.uix.button import Button

class Mybutton(Button):
    text="Click me!"
    on_press =lambda a : print("My Button")

class TutorialApp(App):
    def build(self):
        return Mybutton()
TutorialApp().run()

```

## 4ex。 2

```

from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from kivy.uix.button import Button

class MyLayout(BoxLayout):
    #You don't need to understand these 2 lines to make it work!
    def __init__(self, **kwargs):
        super(MyLayout, self).__init__(**kwargs)

        self.orientation="vertical"
        mylabel = Label(text= "My App")
        self.add_widget(mylabel)
        mybutton =Button(text="Click me!")
        mybutton.bind(on_press= lambda a:print(mylabel.text))
        self.add_widget(mybutton)

```

```
class TutorialApp(App):
    def build(self):
        return MyLayout()
TutorialApp().run()
```

## .kv

### 5 pythonkv

```
from kivy.app import App
from kivy.ui.boxlayout import BoxLayout
# BoxLayout: it's in the python part, so you need to import it

from kivy.lang import Builder
Builder.load_string("""
<MyLayout>
    orientation:"vertical"
    Label: # it's in the kv part, so no need to import it
        id:mylabel
        text:"My App"
    Button:
        text: "Click me!"
        on_press: print(mylabel.text)
""")
class MyLayout(BoxLayout):
    pass
class TutorialApp(App):
    def build(self):
        return MyLayout()
TutorialApp().run()
```

**\*\*6Tutorial.kvkv\*\***

## .py

```
from kivy.app import App
from kivy.ui.boxlayout import BoxLayout

class MyLayout(BoxLayout):
    pass
class TutorialApp(App):
#the kv file name will be Tutorial (name is before the "App")
    def build(self):
        return MyLayout()
TutorialApp().run()
```

## Tutorial.kv

```
<MyLayout> # no need to import stuff in kv!
    orientation:"vertical"
    Label:
        id:mylabel
        text:"My App"
    Button:
        text: "Click me!"
```

```
on_press: print(mylabel.text)
```

**\*\*7kv+pythonlabel.text \*\***

.py

```
from kivy.app import App
from kivy.ui.boxlayout import BoxLayout

class MyLayout(BoxLayout):
    def printMe(self_xx, yy):
        print(yy)
class TutorialApp(App):
    def build(self):
        self.load_kv('myapp.kv')
        return MyLayout()
TutorialApp().run()
```

myapp.kvorientation“vertical”idmylabel text“” on\_pressroot.printMemylabel.text

**8ids “ID”pythondef**

- 7self\_xxself

.py

```
from kivy.app import App
from kivy.ui.boxlayout import BoxLayout

class MyLayout(BoxLayout):
    def printMe(self):
        print(self.ids.mylabel.text)
class TutorialApp(App):
    def build(self):
        self.load_kv('myapp.kv')
        return MyLayout()
TutorialApp().run()
```

myapp.kv

```
<MyLayout>
orientation:"vertical"
Label:
    id:mylabel
    text:"My App"
Button:
    text: "Click me!"
    on_press: root.printMe()
```

**9StringPropertypythondef**

.py

```
from kivy.app import App
```

```

from kivy.uix.boxlayout import BoxLayout
from kivy.properties import StringProperty
class MyLayout(BoxLayout):
    stringProperty_mylabel= StringProperty("My App")
    def printMe(self):
        print(self.stringProperty_mylabel)

class TutorialApp(App):
    def build(self):
        return MyLayout()
TutorialApp().run()

```

## Tutorial.kv

```

<MyLayout>
orientation:"vertical"
Label:
    id:mylabel
    text:root.stringProperty_mylabel
Button:
    text: "Click me!"
    on_press: root.printMe()

```

## 10ObjectPropertypythondef

.py

```

from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.properties import ObjectProperty
class MyLayout(BoxLayout):
    objectProperty_mylabel= ObjectProperty(None)
    def printMe(self):
        print(self.objectProperty_mylabel.text)

class TutorialApp(App):
    def build(self):
        return MyLayout()
TutorialApp().run()

```

## Tutorial.kv

```

<MyLayout>
orientation:"vertical"
objectProperty_mylabel:mylabel
Label:
    id:mylabel
    text:"My App"
Button:
    text: "Click me!"
    on_press: root.printMe()

```

kivy <https://riptutorial.com/zh-CN/kivy/topic/2101/kivy>

---

# 2:

---

## KivyPython

- Python◦

Python◦ ◦ PythonImportError : cannot import name <classname>

- resources.py ◦

## Examples

```
# A line used mostly as the first one, imports App class
# that is used to get a window and launch the application
from kivy.app import App

# Casual Kivy widgets that reside in kivy.uix
from kivy.uix.label import Label
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.uix.screenmanager import SlideTransition

# Inherit Screen class and make it look like
# a simple page with navigation

class CustomScreen(Screen):

    # It's necessary to initialize a widget the class inherits
    # from to access its methods such as 'add_widget' with 'super()'

    def __init__(self, **kwargs):
        # Py2/Py3 note: although in Py3 'super()' is simplified
        # it's a good practice to use Py2 syntax, so that the
        # code is compatible in both versions
        super(CustomScreen, self).__init__(**kwargs)

        # Put a layout in the Screen which will take
        # Screen's size and pos.

        # The 'orientation' represents a direction
        # in which the widgets are added into the
        # BoxLayout - 'horizontal' is the default
        layout = BoxLayout(orientation='vertical')

        # Add a Label with the name of Screen
        # and set its size to 50px
        layout.add_widget(Label(text=self.name, font_size=50))

        # Add another layout to handle the navigation
        # and set the height of navigation to 20%
        # of the CustomScreen
        navig = BoxLayout(size_hint_y=0.2)
```

```

# Create buttons with a custom text
prev = Button(text='Previous')
next = Button(text='Next')

# Bind to 'on_release' events of Buttons
prev.bind(on_release=self.switch_prev)
next.bind(on_release=self.switch_next)

# Add buttons to navigation
# and the navigation to layout
navig.add_widget(prev)
navig.add_widget(next)
layout.add_widget(navig)

# And add the layout to the Screen
self.add_widget(layout)

# *args is used to catch arguments that are returned
# when 'on_release' event is dispatched

def switch_prev(self, *args):
    # 'self.manager' holds a reference to ScreenManager object
    # and 'ScreenManager.current' is a name of a visible Screen
    # Methods 'ScreenManager.previous()' and 'ScreenManager.next()'
    # return a string of a previous/next Screen's name
    self.manager.transition = SlideTransition(direction="right")
    self.manager.current = self.manager.previous()

def switch_next(self, *args):
    self.manager.transition = SlideTransition(direction="right")
    self.manager.current = self.manager.next()

class ScreenManagerApp(App):

    # 'build' is a method of App used in the framework it's
    # expected that the method returns an object of a Kivy widget

    def build(self):
        # Get an object of some widget that will be the core
        # of the application - in this case ScreenManager
        root = ScreenManager()

        # Add 4 CustomScreens with name 'Screen <order>'
        for x in range(4):
            root.add_widget(CustomScreen(name='Screen %d' % x))

        # Return the object
        return root

# This is only a protection, so that if the file
# is imported it won't try to launch another App

if __name__ == '__main__':
    # And run the App with its method 'run'
    ScreenManagerApp().run()

```

## 2SettingsScreenMenuScreen

```

from kivy.app import App
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen

# Create both screens. Please note the root.manager.current: this is how
# you can control the ScreenManager from kv. Each screen has by default a
# property manager that gives you the instance of the ScreenManager used.
Builder.load_string("""
<MenuScreen>:
    BoxLayout:
        Button:
            text: 'First Button on Menu'
            on_press: root.manager.current = 'settings'
        Button:
            text: 'Second Button on Menu'

<SettingsScreen>:
    BoxLayout:
        Button:
            text: 'First Button on Settings'
            on_press: root.manager.current = 'menu'
        Button:
            text: 'Second Button on Settings'

""")

# Declare both screens
class MenuScreen(Screen):
    pass

class SettingsScreen(Screen):
    pass

# Create the screen manager
sm = ScreenManager()
sm.add_widget(MenuScreen(name='menu'))
sm.add_widget(SettingsScreen(name='settings'))

class TestApp(App):

    def build(self):
        return sm

if __name__ == '__main__':
    TestApp().run()

```

<https://riptutorial.com/zh-CN/kivy/topic/6097/>

# 3:

## Examples

- python
- python◦

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.lang import Builder
from kivy.properties import StringProperty
from kivy.properties import ObjectProperty
from kivy.uix.textinput import TextInput
from kivy.event import EventDispatcher
Builder.load_string("""
<CustLab1@Label>
    size_hint:0.3,1
<CustLab2@Label>
    text: "Result"
    size_hint: 0.5,1
<CustButton@Button>
    text: "+1"
    size_hint: 0.1,1
<CustTextInput@TextInput>:
    multiline: False
    size_hint:0.1,1

<Tuto_Property>:
    orientation: "vertical"
    padding:10,10
    spacing: 10
    Label:
        text: "Press the 3 button (+1) several times and then modify the number in the
        TextInput.The first counter (with StringProperty but no binding) doesn't take into account the
        change that happened in the app, but the second one does.String Property makes it easy to pass
        the update from the python side to the user interface, binding pass the changes that happened
        on the user interface to the python side. "
        text_size: self.size
        padding: 20,20

    Property_no_Binding:
    Property_with_Binding:
    Simple:

<Property_no_Binding>:
    spacing: 10
    label_ObjectProperty: result
    CustLab1:
        text: "With Property but no Binding"
    CustButton:
        on_press: root.counter_textInput_StringProperty()
    CustTextInput:
        id:textinput_id
        text: root.textInput_StringProperty
```

```

    CustLab2:
        id: result

<Property_with_Binding>:
    spacing: 10
    label_ObjectProperty: result
    CustLab1:
        text: "With Property and Binding"
    CustButton:
        on_press: root.counter_textInput_StringProperty()
    CustTextInput:
        id:textinput_id
        text: root.textInput_StringProperty
        on_text: root.textInput_StringProperty = self.text    ## this is the binding
    CustLab2:
        id: result

<Simple>
    spacing: 10
    CustLab1:
        text: "Without Property"
    CustButton:
        on_press: root.simple(textinput_id, result)
    CustTextInput:
        id:textinput_id
        text: "0"
    CustLab2:
        id: result

""")

class Property_no_Binding(BoxLayout):
    textInput_StringProperty= StringProperty("0")
    label_ObjectProperty = ObjectProperty(None)
    def counter_textInput_StringProperty(self):
        self.label_ObjectProperty.text= ("Before the counter was updated:\n\n
textInput_id.text:" + self.ids.textinput_id.text + "\n\n textInput_StringProperty:" +
self.textInput_StringProperty)
        self.textInput_StringProperty =str(int(self.textInput_StringProperty)+1)

class Property_with_Binding(BoxLayout):
    textInput_StringProperty= StringProperty("0")
    label_ObjectProperty = ObjectProperty(None)
    def counter_textInput_StringProperty(self):
        self.label_ObjectProperty.text= ("Before the counter was updated:\n\n
textInput_id.text:" + self.ids.textinput_id.text + "\n\n textInput_StringProperty:" +
self.textInput_StringProperty)
        self.textInput_StringProperty =str(int(self.textInput_StringProperty)+1)
    pass

class Simple(BoxLayout):
    def simple(self,textinput_id, result):
        result.text = ("Before the counter was updated:\n\nIn the TextInput:" +
textInput_id.text)
        textinput_id.text = str(int(textinput_id.text) + 1)
    pass

class Tuto_Property(BoxLayout):

```

```
# def __init__(self, **kwargs):
#     super(All, self).__init__(**kwargs)
#     app=App.get_running_app()
#     self.objproper_number.bind(text=lambda *a: self.change(app))
#     print(self.parent)

# def counter(self,app):
#     print("StringProperty:",app.numbertext)
#     print("ObjectProperty:",self.objproper_number.text)
#     print("text:",self.ids.number.text,"\n")
#     app.numbertext=str(int(app.numbertext)+1)

# def change(self, app):
#     app.numbertext=self.objproper_number.text
pass
class MyApp(App):
    numbertext = StringProperty("0")
    def build(self):
        return Tuto_Property()

MyApp().run()
```

<https://riptutorial.com/zh-CN/kivy/topic/9904/>

---

S. No		Contributors
1	kivy	<a href="#">Community</a> , <a href="#">Daniel Engel</a> , <a href="#">EL3PHANTEN</a> , <a href="#">Enora</a> , <a href="#">Fermi paradox</a> , <a href="#">JinSnow</a> , <a href="#">Kallz</a> , <a href="#">KeyWeeUsr</a> , <a href="#">phunsukwangdu</a> , <a href="#">picibucor</a> , <a href="#">user2314737</a> , <a href="#">Will</a>
2		<a href="#">KeyWeeUsr</a> , <a href="#">M Ganesh</a> , <a href="#">OllieNye</a> , <a href="#">picibucor</a>
3		<a href="#">Enora</a> , <a href="#">YOSHI</a>