

How to use **Kivy** in your Python projects





Installation

Open your command line and enter the following*:

***Note: macOS users follow a different procedure from Windows users**

INSTALL ON MACOS

Download the kivy.app and copy it to Applications.

Run the makesymLinks in the window that opens when you open the dmg.

INSTALL ON WINDOWS

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

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

```
python -m pip install kivy.deps.gstreamer
```

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

For a more in-depth tutorial on how to install Kivy, visit <https://kivy.org/doc/stable/installation/installation.html>

More modules and dependencies are available to install through pip on the website.



Hello!

*I'm an **NUI Framework***

I can help you create a simple, cross-platform
Natural User Interface for your program.



What does Kivy do?

Simplify

Kivy simplifies the process of creating a UI for your program. It has a default look and large array of interactive elements to get you started.

Organize

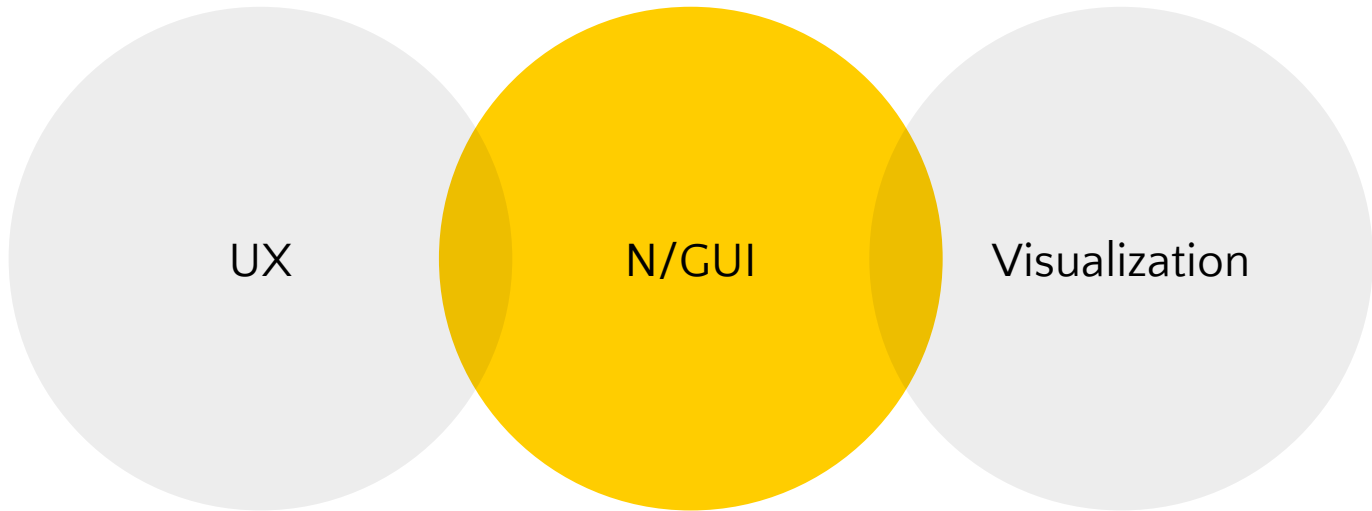
Kivy organizes your user interface in an easily legible and editable way. You can change the interaction between elements easily with one line of code.

Visualize

Kivy has many components that will help users manipulate and visualize 3D, video, and audio files.



What does Kivy do?





Writing in Kv

An easy way to describe user interfaces



Why we use Kv over Python

Kv

Clean, focused on layout and interactions.

Responsive to display size and platform.

Easy to read and edit.

Python

Complicated, focused on logic and function.

Rigid in terms of display size and compatibility not guaranteed.

Messy to edit or debug.



The Kv syntax

Call Kv code



Root Widget



```
from kivy.uix.textinput import TextInput
from kivy.uix.boxlayout import BoxLayout

import random

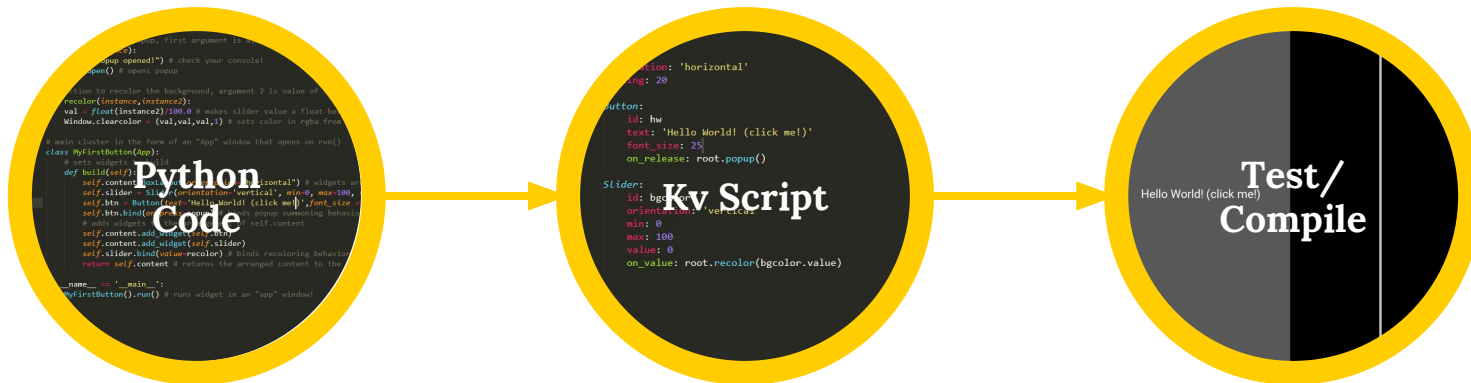
class ScatterTextWidget(BoxLayout):
    def change_label_colour(self, *args):
        colour = [random.random() for i in x
]

class TutorialApp(App):
    def build(self):
        return ScatterTextWidget()

<ScatterTextWidget>:
    orientation: 'vertical'
    TextInput:
        id: my_textinput
        font_size: 150
        size_hint_y: None
        height: 200
        text: 'default'
        on_text:
            FloatLayout:
                Scatter:
                    Label:
                        id: my_label
                        text: my_textinput.text
                        font_size: 150
```



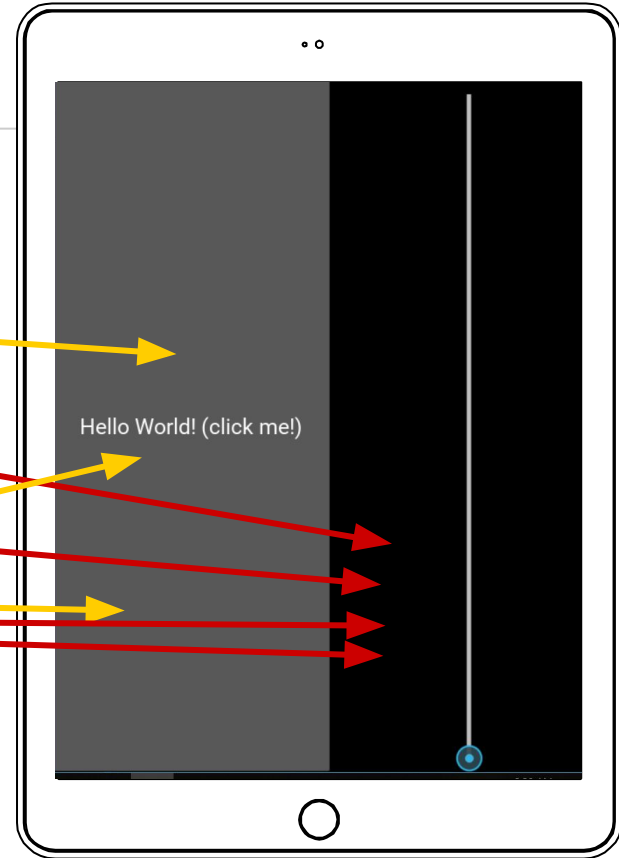

The Kv Workflow





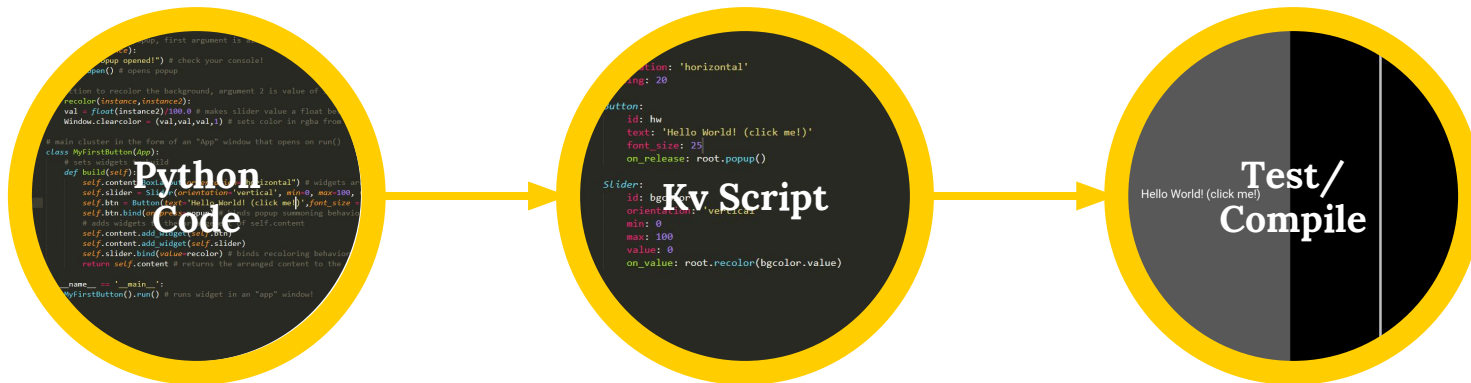
Python syntax

```
100
107 # function to open popup, first argument is always input name (btn in this case)
108 def popup(instance):
109     print("Popup opened!") # check your console
110     code.open() # opens popup
111
112 # function to recolor the background, argument 2 is value of slider
113 def recolor(instance,instance2):
114     val = float(instance2)/100.0 # makes slider value a float between 0 and 1
115     Window.clearcolor = (val,val,val,1) # sets color in rgb from 0-1
116
117 # main cluster in the form of an "App" window that opens on run()
118 class MyFirstButton(App):
119     # sets widgets to build
120     def build(self):
121         self.content=BoxLayout(orientation="horizontal") # widgets arranged horizontally
122         self.slider = Slider(orientation='vertical', min=0, max=100, value = 0) # vertical slider
123         self.btn = Button(text='Hello World! (click me!)',font_size = 25) # button with text
124         self.btn.bind(on_press=popup) # binds popup summoning behavior to button
125         # adds widgets to the arrangement of self.content
126         self.content.add_widget(self.btn)
127         self.content.add_widget(self.slider)
128         self.slider.bind(value=recolor) # binds recoloring behavior to slider
129         return self.content # returns the arranged content to the App
130
131 if __name__ == '__main__':
132     MyFirstButton().run() # runs widget in an "app" window!
133
```





The Kv Workflow





Kv syntax

```
1 # kivy 1.10.1
2
3 <Layout>
4     orientation: 'horizontal'
5     padding: 20
6
7     Button:
8         id: hw
9         text: 'Hello World! (click me!)'
10        font_size: 25
11        on_release: root.popup()
12
13     Slider:
14         id: bgcolor
15         orientation: 'vertical'
16         min: 0
17         max: 100
18         value: 0
19         on_value: root.recolor(bgcolor.value)
20
```

Hello World! (click me!)

```

8   TabbedPanel:
9       id: tp
10      do_default_tab: False
11
12      TabbedPanelItem:
13          id: tab_fl
14          text: 'FloatLayout'
15          on_release: app.showcase_floatlayout(fl)
16          FloatLayout:
17              CFloatLayout:
18                  id: fl
19
20      TabbedPanelItem:
21          text: 'BoxLayout'
22          on_release: app.showcase_boxlayout(box)
23          FloatLayout
24              CBoxLayout:
25                  id: box
26
27      TabbedPanelItem:
28          text: 'GridLayout'
29          on_release: app.showcase_gridlayout(grid)
30          FloatLayout
31              CGridLayout:
32                  id: grid
33                  rows: 3
34
35      TabbedPanelItem:
36          on_release: app.showcase_stacklayout(stack)
37          text: 'StackLayout'
38          FloatLayout
39              CStackLayout:
40                  id: stack
41
42      TabbedPanelItem:
43          text: 'AnchorLayout'
44          on_release: app.showcase_anchorlayout(anchor)
45          FloatLayout
46              CAnchorLayout:
47                  id: anchor
48                  BoxLayout:
49                      orientation: 'vertical'
50                      size_hint: .4, .5
51                      Button
52                      Button
53                      text: 'anchor_x: {}'.format(anchor.anchor_x)

```

oatLayout

BoxLayout

GridLayout

StackLayout

AnchorLayout

rows: None
cols: 3

rows: None
cols: 3

rows: None
cols: 3

rows: None
cols: 3

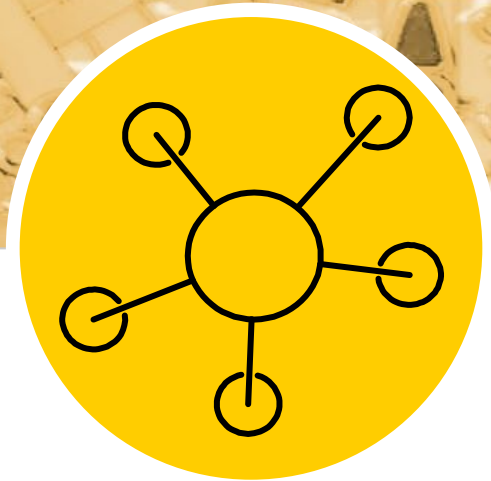
rows: None
cols: 3

rows: None
cols: 3

rows: None
cols: 3

rows: None
cols: 3

rows: None
cols: 3



Cross-platform

Kivy's biggest advantage is its versatility on mainstream platforms.



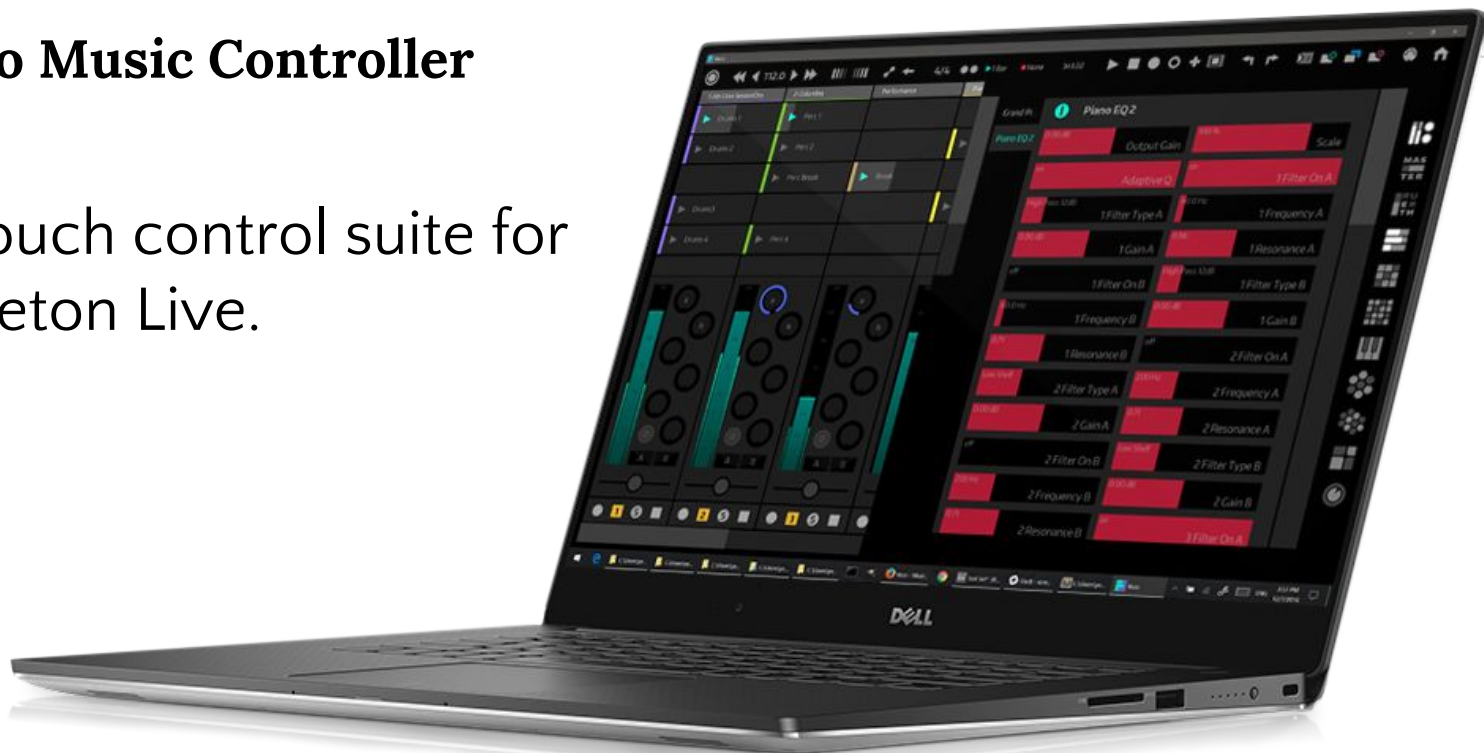
Applications

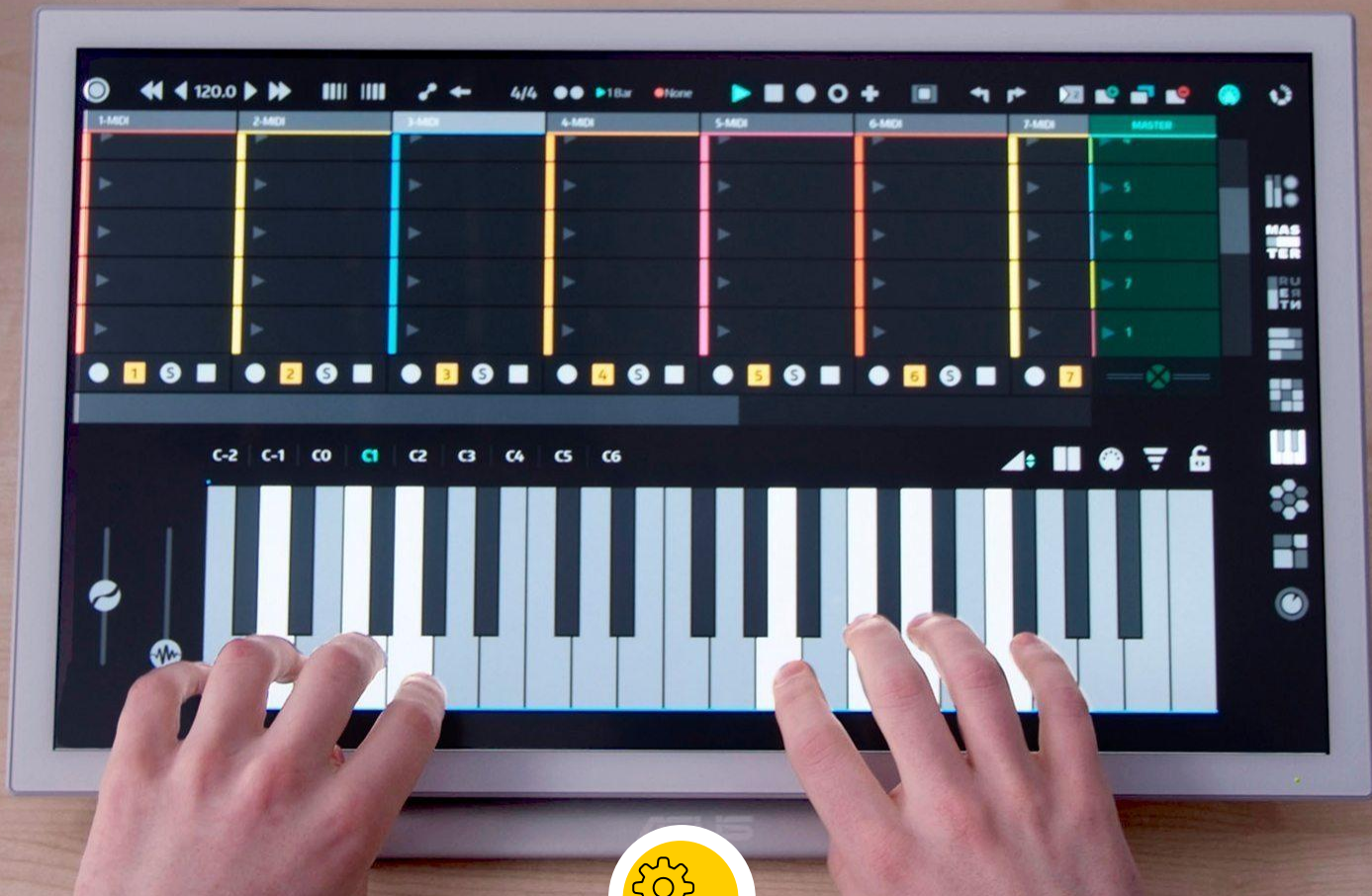
- The Mission Pinball Framework
- Gravity Ride on Google Play (Android)
- Yeco music controller (macOS and Windows)
- The noBOOTH Photo Station
- Various RPi Projects



Yeco Music Controller

A touch control suite for Ableton Live.







Thanks!

Any **questions** ?

Other materials:

- @suddenlykevin on GitHub
- <https://www.kivy.org/>



Credits

<https://kivy.org/#gallery>

<http://www.yeco.io/>

<https://kivy.org/doc/stable/>