Introduction to Kivy

Colin Sauze
<cos@aber.ac.uk>

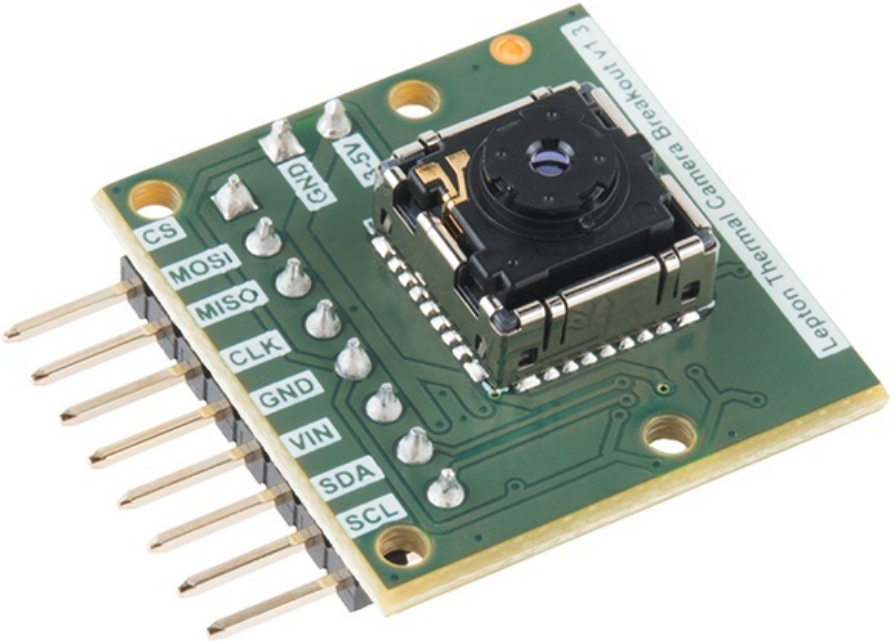
Contents

What is Kivy?

- Cross platform python GUI platform
  - Works in Windows, Mac, Linux, Android and iOS
  - Touchscreen optimised
    - Multitouch support
  - Linux version runs in a framebuffer or X Windows
  - OpenGL accelerated

How I came to use Kivy

- Interface FLIR Lepton Thermal imager
  - Low cost (~£150)
  - 80x60 pixels
  - Demo for showing kids about water stress in plants
  - Raspberry Pi + Touchscreen display



My requirements

- Simple UI needed
  - Only a shutdown button and image
  - Fast startup time
- X based examples for the Lepton
  - Buttons too small for touchscreen
  - Adds to startup time
- Options considered:
  - QT5
  - Kivy
  - Write direct to framebuffer in C
  - Something X based, GTK perhaps
  - Kivy appeared to offer path of least resistance

Installing Kivy

- sudo apt-get install kivy-python
  - Old version
  - Missing some useful features
- pip install kivy
  - Newer version
  - When testing I had to install the system packages:
    - mesa-common-dev
    - libgl-mesa-dev
  - Pip packages:
    - pygame
    - PyHamcrest

Example programs

- https://github.com/colinsauze/kivy-tutorial

Kivy Skeleton App

```
from kivy.app import App
from kivy.uix.widget import Widget


class TestWidget(Widget):
    pass


class TestApp(App):
    def build(self):
        return TestWidget()


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

Empty widget causing blank screen

App class instantiates widget
Returning it, causes it to be displayed

Main runs App class

- A widget is the building block of Kivy
- Lets us draw things on the screen, receive input events etc.

Hello World

```
from kivy.app import App
from kivy.uix.label import Label

class TestApp(App):
    def build(self):
        l = Label(text="Hello world")
        return l

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

Create a label

Returning it causes it to be displayed

Run's the TestApp class

- We actually got away without creating a widget here.
- Will need one for more complex things.

Buttons

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

class TestApp(App):

    def build(self):
        button = Button(text='Hello world')
        button.bind(on_press=self.callback)
        return button

    def callback(self,event):
        print("Pressed Button")


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

Create a button
With text “hello world”

Choose what function
Will be called when button is pressed

Called when button is pressed
Will print “Pressed Button” in terminal

- This will make the button take over whole screen
- Will talk about layouts later

Images

```
from kivy.app import App
from kivy.uix.image import Image

class TestApp(App):

    def build(self):
        image = Image(source='scw-logo.png')
        return image

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

Create an image from the file “scw-logo.png”

Layouts

- Need to be able to manage how/where things appear on screen.
- Add multiple “widgets” such as buttons or images to the layout
- Each will appear in a different location

```
from kivy.app import App
from kivy.uix.gridlayout import GridLayout
from kivy.uix.button import Button
from kivy.uix.image import Image

class TestApp(App):

    def build(self):
       layout = GridLayout(cols=2,rows=2)
       layout.add_widget(Button(text="1"))
       layout.add_widget(Button(text="2"))
       layout.add_widget(Button(text="3"))
       layout.add_widget(Image(source='scw-logo.png'))
       return layout

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

Arrange layout as 2x2 grid
Other layout options are available

Create a button with a “1” in the first cell

Create a button with a “2” in the second cell

Create an image in the fourth cell.

Anchor Layout

- Aligns items to a border:
  - top, bottom, left, right or center
  - layout = AnchorLayout(anchor_x='right', anchor_y='bottom')

Box Layout

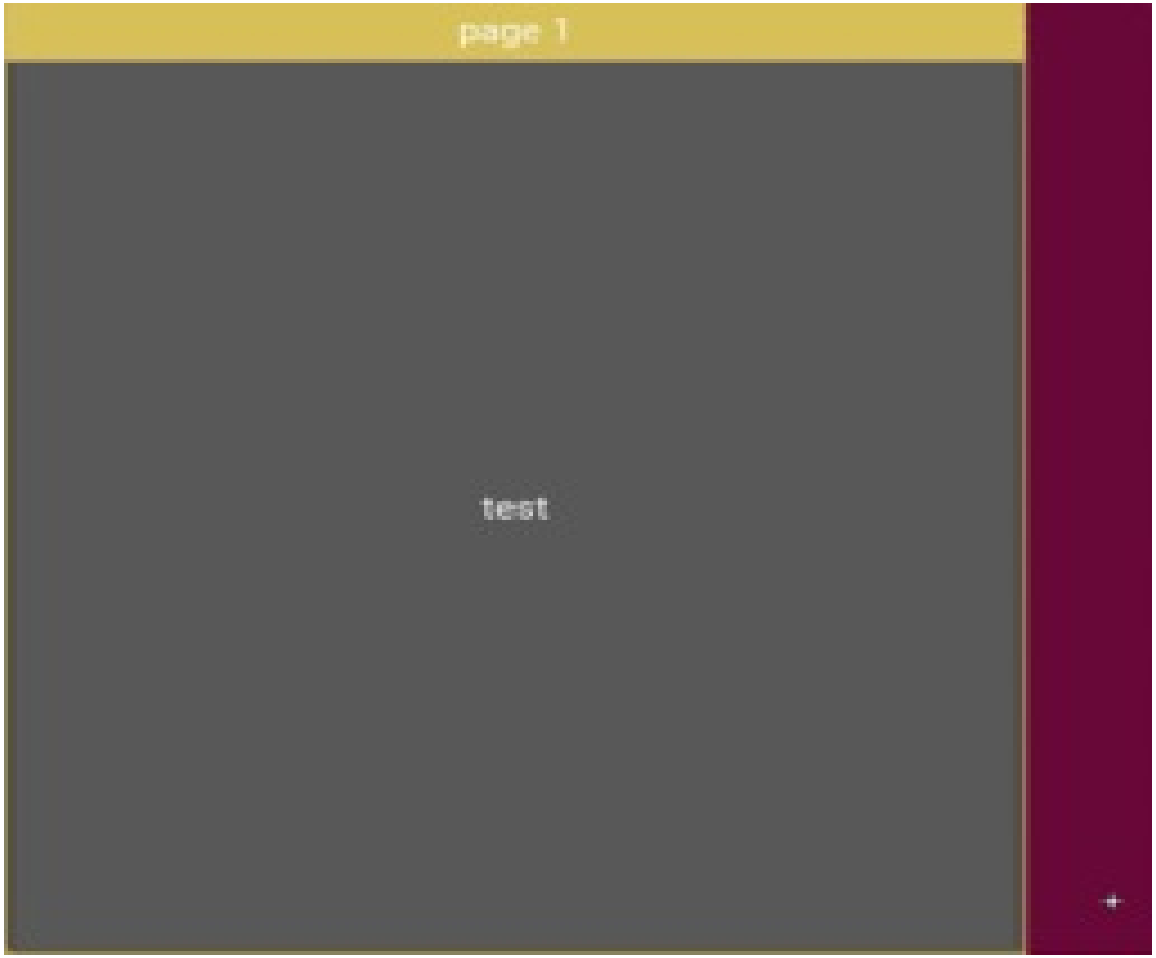
- Horizontal or Vertical Boxes
- layout = BoxLayout(orientation='vertical')

Float, Relative and Scatter Layout

- Float Layout
  - No restrictions
  - Need to give explicit locations
  - Pos argument to widgets
    - e.g. button = Button(text='Hello',pos=(2, 2))
- Relative Layout
  - Like float layout
  - But locations are relative
  - Not absolute
- Scatter Layout
  - Like relative layout
  - Widgets can be rotated or scaled

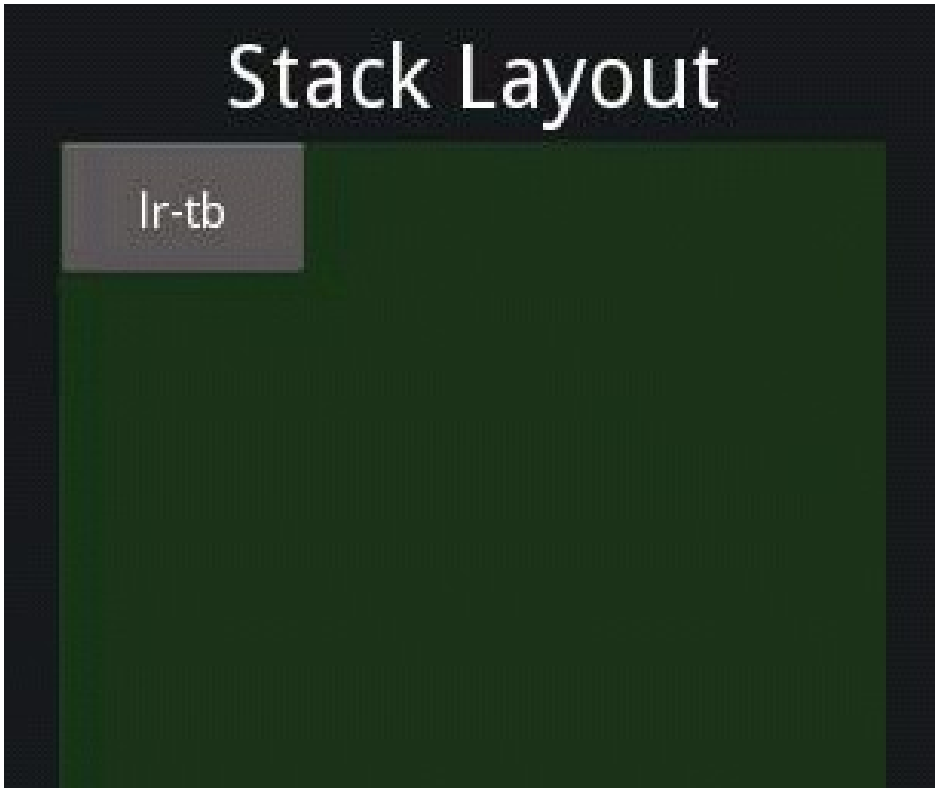
Page Layout

- Sliding/Flipping Pages
- Think mobile app



Stack Layout

- Widgets added in horizontal or vertical stack
- Don't all have to be same size



KV Language

- As interfaces get complicated we need a lot of code
- Mixing of code between layout and program logic
- KV language alternate language to design interface
- Leave Python code to do the logic
- KV is to Kivy what CSS is to HTML/Javascript

KV Example

6.py:

```
from kivy.app import App
from kivy.uix.widget import Widget

class TestWidget(Widget):
    pass

class TestApp(App):
   def build(self):
      return TestWidget()

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

test.kv

```
<TestWidget>:

 GridLayout:
     width: 800
     height: 600
     cols: 2
     rows: 2
     Button:
         text: "1"
     Button:
         text: "2"
     Button:
         text: "3"
     Image:
         source: "scw-logo.png"
```

Set GridLayout Force width Without this, it displays wrong

Make Buttons

Creates a widget called TestWidget

Display image

KV and button events

```
from kivy.app import App
from kivy.uix.widget import Widget

class TestWidget(Widget):
    def button_handler(self):
       print("button pressed")

class TestApp(App):
    def build(self):
       return TestWidget()

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

Handler for button presses

Name of function to do call backs when pressing button.

```
#:kivy 1.2.0
<TestWidget>:
    Button:
        text: "1"
        on_press: root.button_handler()
```

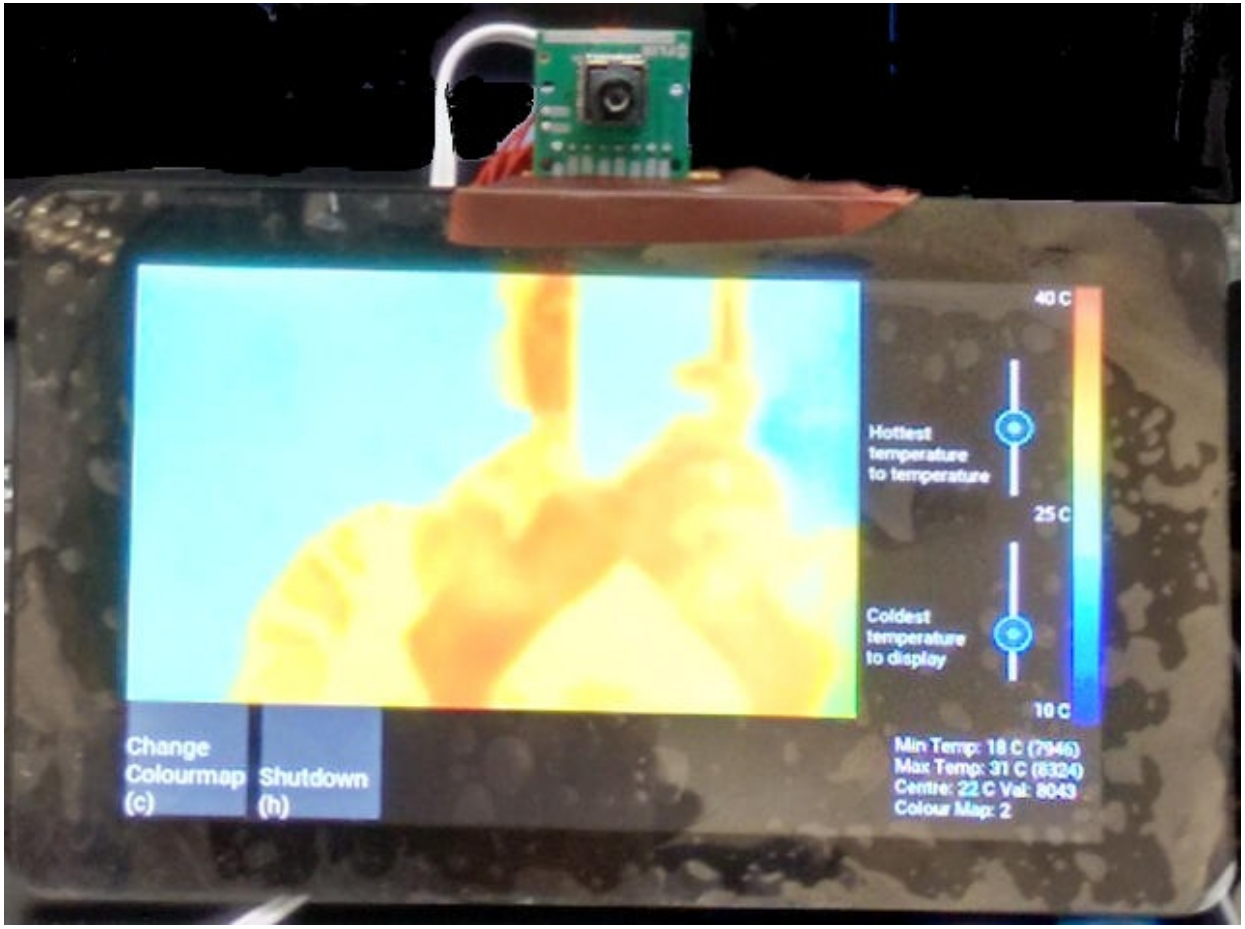
Kivy Quirks and Gotchas

- Multiple ways to do things
- Widgets vs Apps
- Naming of KV files and class names
  - .kv file should be all lower case
  - Named after the class which runs first in the .py file
  - If class name ends “App” (e.g. TestApp”) then just include first bit. Kv file for class caled TestApp is test.kv

Controls Gallery

- Run showcase example
- https://github.com/kivy/kivy/tree/master/examples/demo/showcase
- `cd examples/demo/showcase`
- `python3 main.py`

Thermal Imager



- Live updating image built from an array read from the camera.
- Sliders for min/max temperature
- Temperature Scale image
- Change colourmap button
- Shutdown system button
- Status Labels

Find out More

- Kivy Tutorials - https://kivy.org/docs/tutorials-index.html
  - Pong game
  - Simple Drawing
- Gallery of Examples - https://kivy.org/docs/examples/index.html
- Kivy Crash Course (with Youtube videos) - http://inclem.net/pages/kivy-crash-course/