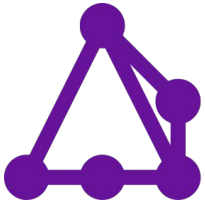
Python on Your Phone

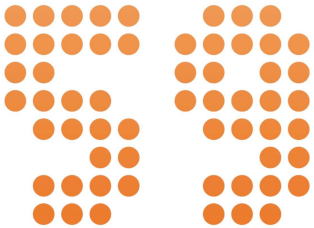
A brief introduction to building mobile apps in Python with Kivy

Hi, I'm Derek.

ordrslip™



root access
hackerspace



DAYS of CODE

fresno.py

bit.ly/kivy-djangocon

github.com/dmpayton/snakes-on-a-droid/tree/djangocon-2018

How I got here...


3:37 PM
kUSC.org
CLASSICAL | fm 91.5
Classical Music with Kimberlea Daggy
Antonin Dvorak: Symphony #9 "New World" in e Op 95

meh.


I ❤ Python
conf.py
router.py
screens.py
store.py
utils.py
kv
main.py
bin
data
libs
garden
garden.router
garden.simpletable
utils
.gitignore
README.rst
activated-chords.json
buildozer.spec
requirements.txt
clearcolor=(1,1,1,1),
duration=.1
)
return screens.MenuScreen()
@route('/chords')
def chord_list(self):
return screens.ChordListScreen()
@route('/chords/<path:chord>')
def chord_detail(self, chord):
return screens.ChordDetailScreen(chord=chord.strip('/'))
@route('/practice')
def practice(self):
return screens.PracticeScreen()
@route('/scales')

I **want** to build software that works on my phone…

…but I also prefer to build software in **Python**

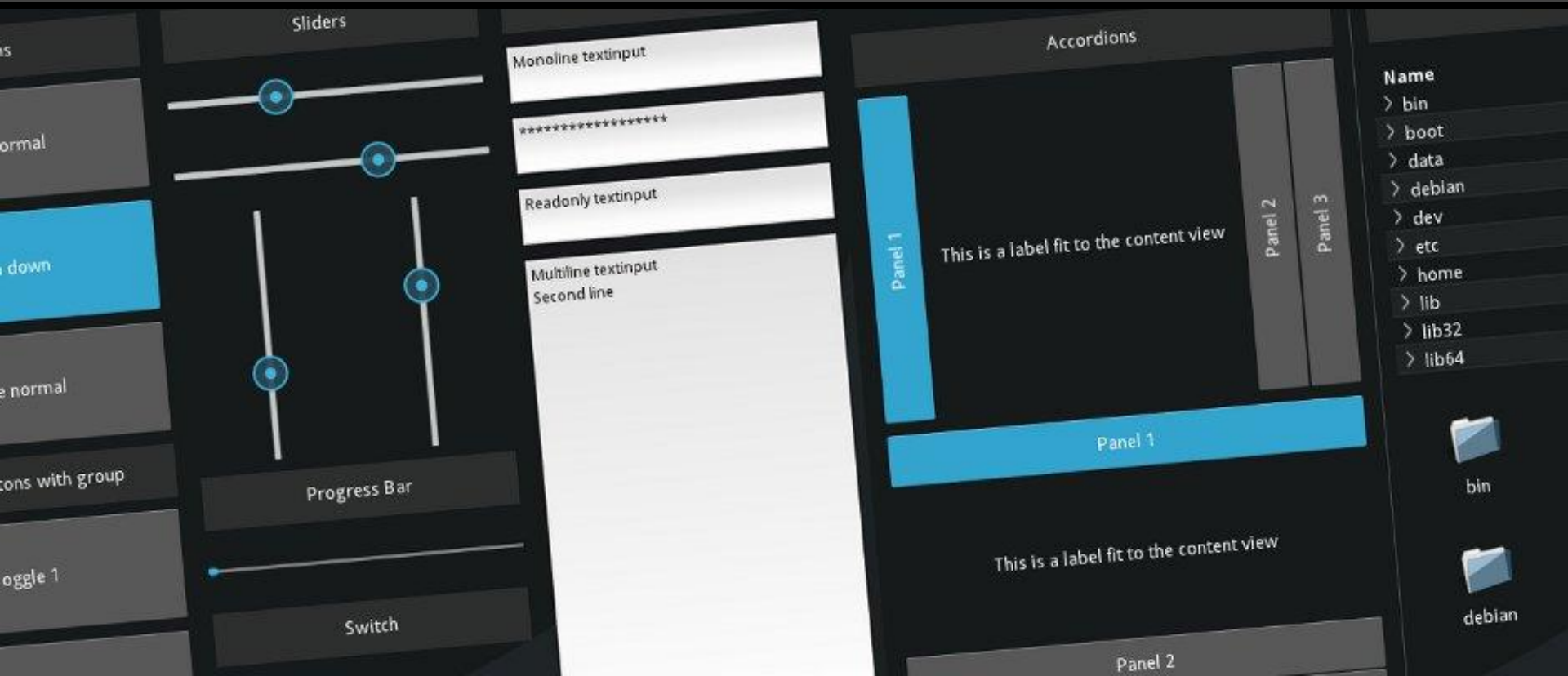
BeeWare

Build native apps with Python.

pybee.org

kivy

kivy.org



Kivy is...

A Python library for creating multi-touch applications

Kivy is...

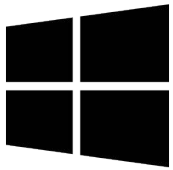
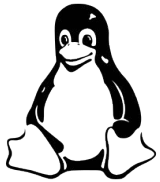
Open Source

MIT License

Kivy is...

Cross-platform



Kivy is...

Fast.

cython

The missing link between the **simplicity of Python** and the **speed of C**

cython.org


DEMO


HELLO
WORLD

Installing Kivy

```
$ pip install Cython==0.25.2
$ pip install kivy==1.10.0

# optional but useful
$ pip install Pillow pygame
```

01. hello, world

main.py

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

class DemoApp(App):
    def build(self):
        return Label(text='hello, world', font_size=60)

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

01. hello, world

main.py

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

class DemoApp(App):
    def build(self):
        return Label(text='hello, world', font_size=60)

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

01. hello, world

main.py

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

class DemoApp(App):
    def build(self):
        return Label(text='hello, world', font_size=60)

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

01. hello, world

main.py

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

class DemoApp(App):
    def build(self):
        return Label(text='hello, world', font_size=60)

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

02. multi-touch

main.py

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

class DemoApp(App):
    def build(self):
        scatter = ScatterLayout()
        label = Label(text='hello, world', font_size=60)
        scatter.add_widget(label)
        return scatter

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

02. multi-touch

main.py

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

class DemoApp(App):
    def build(self):
        scatter = ScatterLayout()
        label = Label(text='hello, world', font_size=60)
        scatter.add_widget(label)
        return scatter

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

02. multi-touch

main.py

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

class DemoApp(App):
    def build(self):
        scatter = ScatterLayout()
        label = Label(text='hello, world', font_size=60)
        scatter.add_widget(label)
        return scatter

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

03. events

main.py

```
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.floatlayout import FloatLayout
from kivy.uix.popup import Popup


...
```

03. events

main.py

```
...

class DemoApp(App):
    def build(self):
        layout = FloatLayout()
        open_button = Button(
            text='click me!',
            size_hint=(.5, .5),
            pos_hint={'center_x': .5, 'center_y': .5}
        )
        layout.add_widget(open_button)

        ...
```

03. events

main.py

```
...

popup = Popup(
    title='hello, world',
    auto_dismiss=False,
    size_hint=(.3, .3)
)
close_button = Button(text='close me!')
popup.add_widget(close_button)

...
```

03. events

main.py

```
        ...

        open_button.bind(on_release=popup.open)
        close_button.bind(on_release=popup.dismiss)

        return layout


        ...
```

Kv Design Language

04. Kv lang

main.py demo.kv

```
from kivy.app import App

class DemoApp(App):
    pass

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

04. Kv lang

main.py | demo.kv

```
FloatLayout:
    id: layout

    Button:
        id: open_button

    Popup:
        id: popup

        Button:
            id: close_button
```

04. Kv lang

main.py | **demo.kv**

```
FloatLayout:
    id: layout

    Button:
        id: open_button
        text: 'click me!'
        size_hint: (.5, .5)
        pos_hint: {'center_x': .5, 'center_y': .5}
        on_release: root.ids['popup'].open()

    Popup:
        id: popup
        ...
```

04. Kv lang

main.py | demo.kv

```
FloatLayout:
    ...

    Popup:
        id: popup
        title: 'hello, world'
        auto_dismiss: True
        size_hint: (.3, .3)
        on_parent: if self.parent == layout: \
            layout.remove_widget(self)
```

04. Kv lang

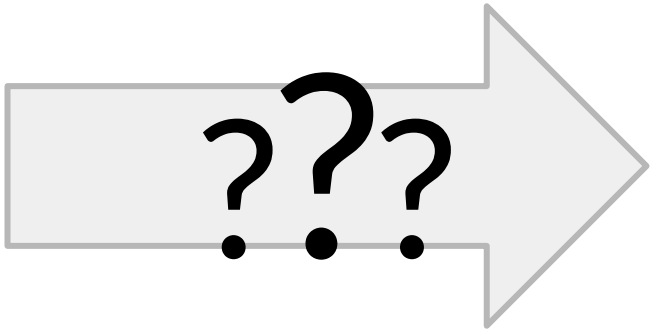
main.py | **demo.kv**

```
FloatLayout:
    ...

    Popup:
        ...

        Button:
            id: close_button
            text: 'close me!'
            on_release: root.ids['popup'].dismiss()
```

We've built a **Kivy** app, and we need to **package** it for mobile devices.



python-for-android

Packages Python apps for Android

github.com/kivy/python-for-android

kivy-ios

A toolchain for compiling Python apps to run on iOS

github.com/kivy/kivy-ios


STOP

buildozer

Tool for creating application packages

github.com/kivy/buildozer

buildozer

Build for Android or iOS using a common spec file.

github.com/kivy/buildozer

buildozer

```
$ pip install buildozer
```

buildozer

```
$ buildozer init
```

buildozer.spec

```
[app]

# (str) Title of your application
title = My Application

# (str) Package name
package.name = myapp

# (str) Package domain (needed for android/ios packaging)
package.domain = org.test.myapp

# (str) Source code where the main.py live
source.dir = .

# (list) Source files to include (let empty to include all the files)
source.include_exts = py,png,jpg,kv,atlas

...
```

buildozer.spec

```
...

#
# Android specific
#

# (bool) Indicate if the application should be fullscreen or not
fullscreen = 1

# (list) Permissions
android.permissions = INTERNET

# (int) Android API to use
android.api = 19

# (int) Minimum API required
android.minapi = 9

...
```

buildozer

```
$ buildozer android debug
```

buildozer

```
$ buildozer android debug /
  deploy run
```

How do you do actual **hardware** things?

Like, maybe, read the accelerometer?

pyjnius

Python module to access Java classes through the JNI

github.com/kivy/pyjnius

pyjnius

```
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;

public class Hardware {
   // You have to write some Java. D:
}
```

pyjnius.readthedocs.io/en/latest/android.html#accelerometer-access

pyjnius

```
from time import sleep
from jnius import autoclass

Hardware = autoclass('org.test.android.Hardware')
Hardware.accelerometerEnable(True)

for x in xrange(20):
    print(Hardware.accelerometerReading())
    sleep(.1)
```

pyjnius.readthedocs.io/en/latest/android.html#accelerometer-access

pyobjus

Python module to access Objective-C classes as Python classes

github.com/kivy/pyobjus

pyobjus

```
// I have no idea what's going on here. :-/
@interface bridge : NSObject {
    NSOperationQueue *queue;
}

@property (strong, nonatomic) CMMotionManager
*motionManager;
@property (nonatomic) double ac_x;
@property (nonatomic) double ac_y;
@property (nonatomic) double ac_z;
@end
```

pyobjus.readthedocs.io/en/latest/pyobjus_ios.html#accessing-the-accelerometer

pyobjus

```
from pyobjus import autoclass

Bridge = autoclass('bridge')
br = Bridge.alloc().init()
br.motionManager.setAccelerometerUpdateInterval_(0.1)
br.startAccelerometer()

for i in range(20):
    print((br.ac_x, br.ac_y, br.ac_z))
```

pyobjus.readthedocs.io/en/latest/pyobjus_ios.html#accessing-the-accelerometer

I just want to write some Python and be done!

plyer

Platform-independent API for common hardware features

github.com/kivy/plyer

plyer

pyjnius on Android
pyobjus on iOS

github.com/kivy/plyer

plyer

```
from time import sleep
from plyer import accelerometer

accelerometer.enable()

for x in xrange(20):
    print(accelerometer.acceleration)
    sleep(.1)
```

I ❤ Guitar


Chordwise


WE CAN BUILD MOBILE
APPS IN PYTHON

Thanks!

Check out Kivy at kivy.org

Hit me up

derek.payton@gmail.com
@dmpayton on Twitter