Kivy

High-level Python-based tool for developing cross-platform GUI applications

Kivy

High-level Python-based tool for developing cross-platform GUI applications:

- Windows
- Linux
- Android
- IOX
- Raspberry Pi

Kivy

High-level Python-based tool for developing cross-platform GUI applications:

- Windows
- Linux
- Android
- IOX
- Raspberry Pi

Interact with:

- Touch-sensitive screens
- Smart phone hardware

AutoHotKey

- I decided to take advantage of Windows 10 Multiple Desktops for this presentation
- Frustrated because I could not find any way to control the desktops from a command prompt
- Enter Google
  Which led to https://autohotkey.com/
- More about this unexpedted gem later

How did this start?

- Wife on low-carb diet
- Using Excel to determine carbs in recipe serving
- I tell her this should be a database application
- We start discussing Microsoft Access
- She says she'd like to see recipes on her mobile devices
- I say I'd like to develop an App to do that
- I research App development tools
  - what I find is Java- based or graphical meaning limited
- I say to self: But I can do more with less effort using Python
  - so I Google android app using python

I discovered Buildozer

- Buildozer is a tool that aim to package mobiles application easily. It automates the entire build process, download the prerequisites like python-for-android, Android SDK, NDK, etc.
- Currently, Buildozer supports packaging for:
- Android: via Python for Android. You must have a Linux or OSX computer to be able to compile for Android.
- I played around with this for a while, then found it was based on Kivy. What's that, I say?
- Well you already heard my "high-level" intro:

Kivy

High-level Python-based tool for developing cross-platform GUI applications:

- Windows
- Linux
- Android
- IOX
- Raspberry Pi

Interact with:

- Touch-sensitive screens
- Smart phone hardware

- So let’s jump into Kivy
- We’ll come back to Buildozer later

Demo 1 - Pong

Our First App

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

def press(self):
    print('pressed!')
    self.text = "Ouch"

class Tutorial3(App):
     def build(self):
        return Button(
            text='Hello',
            background_color=(0, 0, 1, 1),
            font_size=150,
            on_press = press)

if__name__ == "__main__":
    Tutorial3().run()
```

Kivy Language

- As your application grow more complex, it's common that the construction of widget trees and explicit declaration of bindings, becomes verbose and hard to maintain. The *KV* Language is a attempt to overcome these short-comings.
- The *KV* language (sometimes called kvlang, or kivy language), allows you to create your widget tree in a declarative way and to bind widget properties to each other or to callbacks in a natural manner. It allows for very fast prototyping and agile changes to your UI. It also facilitates a good separation between the logic of your application and its User Interface.

Two Ways to Incorporate Livy Language

- Embed Kivy Language in Python program
  - Edit both in one edit window
  - Must provide code to use Kivy Language
- Put Kivy Language in separate file
  - 2 edit windows
  - No extra code
  - Filename derived from Application name

Three Ways to Make an App

- Python
- Python with Kivy Language embedded
- Python with Kivy Language in separate file

First App with
Using Kivy Language – .py + .kv

```
# tutorial.3.py
from kivy.app import App

class Tutorial3(App):
  pass

if__name__ == "__main__":
    Tutorial3().run()
```

```
# tutorial3.kv
Button:
    text:'Hello'
    background_color: (0, 0, 1, 1)
    font_size=150
     on_press:
        print('pressed!')
        self.text = "Ouch"
```

Kivy
  extracts “tutorial3” from the root class name
  looks for a file named tutorial4.kv

Button: replaces
 from kivy.uix.button import Button
 Button()

on_press: replaces
 def press()
    on_press = press

First App
Embedded Kivy Language

```
# tutorial.3.py
from kivy.app import App
from kivy.lang import Builder

class Tutorial3(App):
    def  build(self):
        return Builder.load_string("""
Button:
    text:'Hello'
    background_color: (0, 0, 1, 1)
    font_size=150
     on_press:
        print('pressed!')
        self.text = "Ouch"
""")

if__name__ == "__main__":
    Tutorial3().run()
```

Benefits of Kivy

- Leverage your knowledge of Python
- Kivy apps run on Windows, Mac, Linux, Android, Ios.
- Open Source.
- Extensive library of widgets.
- Touch interaction.
- Smart phone hardware interface.
- Kivy language for separation between application logic and User Interface.

Kivy Negatives

- Documentation
- Learning curve
- Support

Components

UX: Classical user interface widgets, ready to be assembled to create more complex widgets.
Label, Button, CheckBox, Image, Slider, Progress Bar, Text Input, Toggle button, Switch, Video

Layouts: do no rendering, just acts as a trigger that arranges its children in a specific way.
Anchor Layout, Box Layout, Float Layout, Grid Layout, PageLayout, Relative Layout, Scatter Layout, Stack Layout

Complex UX: Non-atomic widgets (the result of combining multiple classic widgets). We call them complex because their assembly and usage are not as generic as the classical widgets.
Bubble, Drop-Down List, FileChooser, Popup, Spinner, List View, TabbedPanel, Video player, VKeyboard,

Behaviors: do no rendering but act on the graphics instructions or interaction (touch) behavior of their children.
Scatter, Stencil View

Screen manager: Manages screens & transitions when switching from one to another.

Events:

Your boss asks you to...

Kivy comes with a rich set of examples. I want a GUI app that will let me navigate through a directory tree of these examples to a particular python program and then run that program.

How much code do you think this will take?

What Python libraries do you have to choose from?

What can Kivy do?

Browse Examples App 1

tutorial.0.py

```
from kivy.app import App

class Tutorial0App(App):
    pass

if __name__ == "__main__":
    Tutorial0App().run()
```

Gotcha!
kv filename is lower case!

tutorial0.kv

```
#:import call subprocess.call
FileChooserListView:
    path: "C:\python35\share\kivy-examples"
    filters: ['*.py']
    on_submit: call(r'c:\python35\python %s' % self.selection[0])
```

Browse Examples App 2

```
from kivy.app import App
from kivy.lang import Builder

class MyApp(App):
    def build(self): return Builder.load_string("""
#:import call subprocess.call
<TabbedPanelStrip>
    canvas:
        Color:
            rgba: (0, 1, 0, 1) # green
        Rectangle:
            size: self.size
            pos: self.pos
<ToggleButton>:
    font_name: "arial"
    bold: True
<FileChooserListView, FileChooserIconView>
    path: "C:\python35\share\kivy-examples"
    filters: ['*.py']
    on_submit: call('c:\python35\python ' + self.selection[0])
    TabbedPanel:
    do_default_tab: False
    tab_height: 35

    TabbedPanelItem:
        text: "List"

        FileChooserListView:

    TabbedPanelItem:

        text: "Icon"

        FileChooserIconView:

    """)

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

Layouts

https://kivy.org/docs/guide/widgets.html - organize-with-layouts

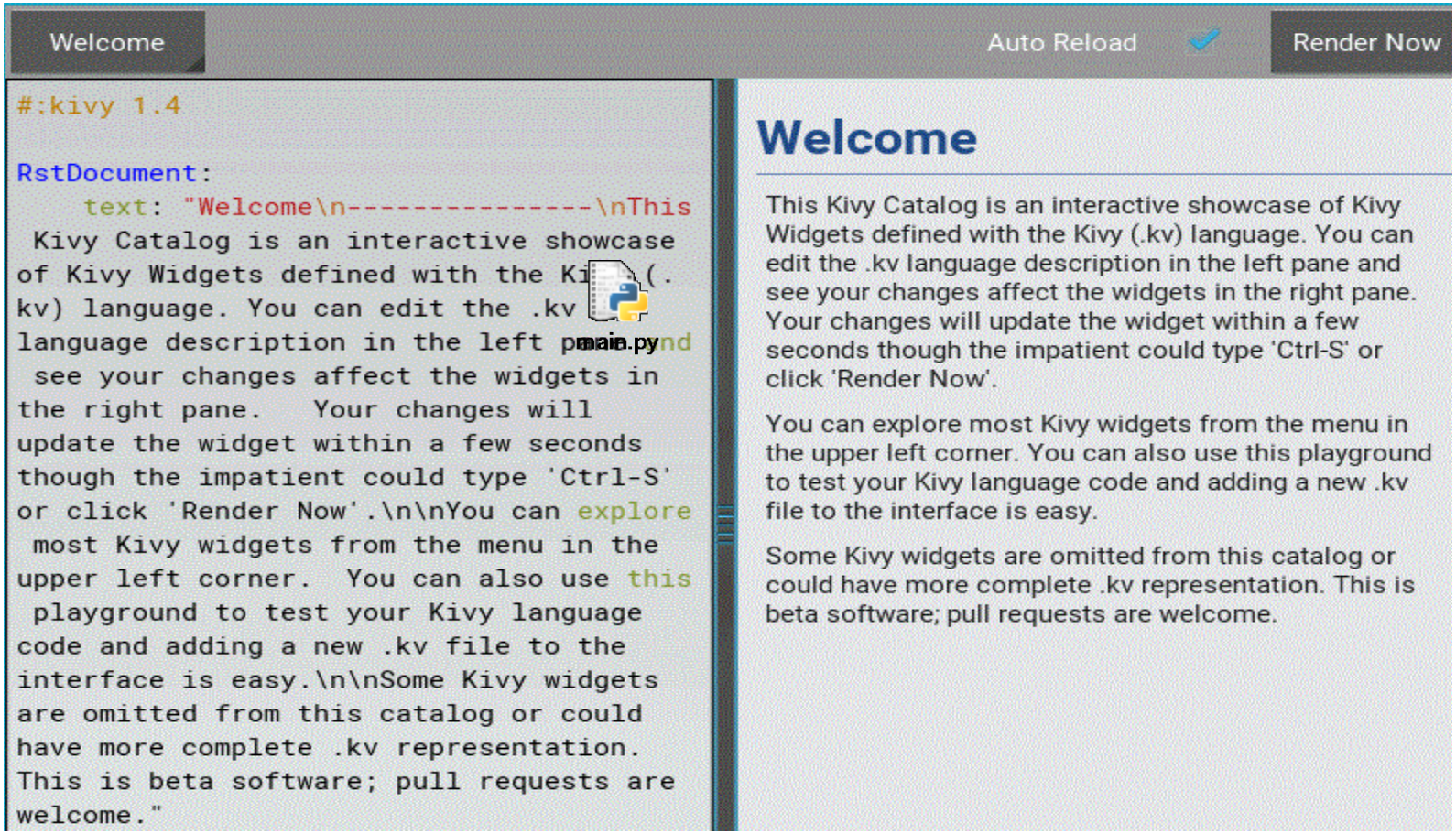
Kivy Examples and Layouts

Installing Kivy Examples and where did they go?

C:\Python27>python share\kivy-examples\demo\kivycatalog\main.py

An interactive tool for demonstrating and exploring layouts

Click Welcome to see the menu



Let’s see it in action!

Behaviors

- https://kivy.org/docs/api-kivy.uix.behaviors.html

```
from kivy.uix.behaviors import ButtonBehavior
from kivy.uix.image import Image

class IconButton(ButtonBehavior, Image):
    def on_press(self):
        print("on_press")
```

```
Button
ToggleButton
Drag
Focus
CompoundSelection
CodeNavigation
Cover
```

Android Emulator

- Droid4X – Windows
- Caution – “hijacks” Oracle VirtualBox
- Demo

Android - Kivy Launcher

android app that runs code
Get from Google Play Store or
http://kivy.org/#download
Files in /sdcard/kivy/project

androiod.txt

```
title=Pong
author=Kivy team
orientation=landscape
```

a

main.py

```
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.properties import
NumericProperty, ReferenceListProperty,
ObjectProperty
from kivy.vector import Vector
from kivy.clock import Clock

class PongPaddle(Widget):
    score = NumericProperty(0)

    def bounce_ball(self, ball):
        if self.collide_widget(ball):
            vx, vy = ball.velocity
            offset = (ball.center_y - self.center_y)
/ (self.height / 2)
            bounced = Vector(-1 * vx, vy)
            vel = bounced * 1.1
            ball.velocity = vel.x, vel.y + offset
```

Buildozer

- Power tool to create android / ios apps
- Available only on linux
- https://buildozer.readthedocs.io/en/latest/

Following is ubuntu 16.04 64 bit

```
$> pip install --upgrade buildozer
$> sudo pip install --upgrade cython==0.21
$> sudo dpkg --add-architecture i386
$> sudo apt-get update
$> sudo apt-get install build-essential ccache git libncurses5:i386 libstdc++6:i386
  libgtk2.0-0:i386 libpangox-1.0-0:i386 libpangoxft-1.0-0:i386 libidn11:i386 python2.7
  python2.7-dev openjdk-8-jdk unzip zlib1g-dev zlib1g:i386
$> mkdir project1 # they forgot to tell you this
$>cd project1
$> # put kivy code in main.py; add .kv file as needed
$>buildozer init (edit buildozer.spec)
$> buildozer -v android debug (deploy run logcat)
```

First time only downloads and installs various dependencies – allow LOTS of time.

.apk file is placed in bin directory

Buildozer

- So why doesn't it work – what they did not tell you.
- Android device must be in debug mode

http://www.techotopia.com/index.php/Testing_Android_Applications_on_a_Physical_Android_Device_with_ADB

- Be prepared to capture 1000's of lines of log file thru which you search

Installating Kivy

Python 3.5 - Windows

http://www.newthinktank.com/2016/10/kivy-tutorial/
https://github.com/kivy/kivy/wiki/Batch-installer-for-windows(KivyInstaller)
https://github.com/kivy
https://github.com/KeyWeeUsr/KivyInstaller

Other

- Forum
- IRC
- Garden
- Git
- Google

Button — Kivy 1.9.2.dev0 documentation
https://kivy.org/docs/api-kivy.uix.button.html
The *Button* is a Label with associated actions that are triggered when the *button* is pressed (or released after a click/touch). To configure the *button*, the same ...

TabbedPanel — Kivy 1.9.2.dev0 documentation
https://kivy.org/docs/api-kivy.uix.tabbedpanel.html
The TabbedPanel widget manages different widgets in tabs, with a header area for the actual tab *buttons* and a content area for showing the current tab content.You visited this page.

Button Behavior — Kivy 1.9.2.dev0 documentation
https://kivy.org/docs/api-kivy.uix.behaviors.button.html
The ButtonBehavior mixin class provides *Button* behavior. You can combine this class with other widgets, such as an Image , to provide alternative *buttons* that ...

kivy.uix.button — Kivy 1.9.2.dev0 documentation
https://kivy.org/docs/_modules/kivy/uix/button.html
Label` with associated actions that are triggered when the *button* is pressed (or released after a click/touch). To configure the *button*, the same properties ...

Toggle button — Kivy 1.9.2.dev0 documentation
https://kivy.org/docs/api-kivy.uix.togglebutton.html
When you touch/click it, the state toggles between 'normal' and 'down' (as opposed to a *Button* that is only 'down' as long as it is pressed). Toggle *buttons* can ...

python - Bind function to Kivy button - Stack Overflow
stackoverflow.com/questions/23127203/bind-function-to-kivy-button
Apr 17, 2014 - I'm trying to bind the following function to a *Button* in *Kivy* .... I'm guessing at your class structure, but it would be good if you could fill out the ...

How to set command for Button wiget in Kivy language - Software ...

https://www.daniweb.com › ... › Programming Forum › Software Development Forum

Feb 10, 2015 - Hello! Can you give me an example of *Button* widget in *Kivy* language? I can creat the *button* but don't ...

A button that does something in Kivy - YouTube

https://www.youtube.com/watch?v=RJigyKq9D4M

Jan 29, 2015 - Uploaded by Mark Winfield

Here we build an App that has a *button* that when we press it a message is displayed in the console.

Searches related to kivy button

kivy button **size**
kivy button **tutorial**
kivy button **color**
kivy button **kv**
kivy **toggle** button **example**
kivy **on_touch_down**
kivy button **behavior**
kivy **bind property**

Links

https://github.com/MYounas/books/blob/master/Kivy%20Interactive%20Applications%20in%20Python.pdf
https://kivy.org/docs/api-kivy.html
https://kivy.org/docs/guide/widgets.html - organize-with-layouts
https://github.com/bgailer/TriZPUG-Presentation

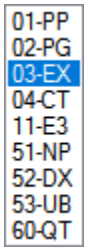
AutoHotKey

From https://autohotkey.com/

- **Key Binds**
- Define hotkeys for the mouse and keyboard, remap keys or buttons and autocorrect-like replacements. Creating simple hotkeys has never been easier; you can do it in just a few lines or less!
- **What is AutoHotkey**
- AutoHotkey is a free, open-source scripting language for Windows that allows users to easily create small to complex scripts for all kinds of tasks such as: form fillers, auto-clicking, macros, etc.
- **Is it good for me?**
- Autohotkey has easy to learn built-in commands for beginners. Experienced developers will love this full-fledged scripting language for fast prototyping and small projects.
- **Why AutoHotkey**
- Autohotkey gives you the freedom to automate any desktop task. It's small, fast and runs out-of-the-box. Best of all, it's free, open-source (GNU GPLv2), and beginner-friendly. Why not give it a try?

What I did with AutoHotKey

- Created a little listbox that appears upper left.
- A click starts an application in new desktop or switches to it if application window already exists.
- In some cases also runs a python program.
- PP = PowerPoint
- PG = Pong
- EX = Examples Explorer/Launcher
- NP = NotePad
- DX = Android Emulator
- UB = VNC connection to the Ubuntu Laptop
- QT = Quit



```
apps := {}
apps["01-PP"] := ["Kivy.pptx", "N:\Kivy\TriZPUG\Kivy.pptx"]
apps["02-PG"] := ["cmd", "pong"]
apps["03-EX"] := ["cmd", "ex"]
apps["11-E3"] := ["cmd", "tutorial.3"]
apps["51-NP"] := ["Notepad", "Notepad"]
apps["53-UB"] := ["Tiger", "n:\kivy\trizpug\vncviewer"]
apps["52-DX"] := ["Droid4X", "C:\Program Files (x86)\Droid4X\Droid4X.exe"]
apps["60-QT"] := ""
l := "" ; string of keys separated by | for ListBox
r := 0  ; # of keys
for k in apps {
        l := l k "|"
        r += 1
}
l := SubStr(l, 1, strlen(l)-1)
SetTitleMatchMode 2
Gui, New,  AlwaysOnTop -caption
Gui, Add, ListBox, vApp gLsub Sort R%r% W37 x0 y0, %l%
gui, margin, 0, 0
Gui, Show, x0 y0 NoActivate
return
```

```
Lsub:
GuiControlGet, app
if (App = "60-QT")
        ExitApp
w := apps[App]
if (w.length() = 1)
  runwait % w[1]
else {
  IfWinExist % w[1]
    WinActivate % w[1]
  else {
        send ^#d   ; add a desktop to the task views
        If substr(w[1], 1, 3) = "cmd"
                run % "n:\kivy\trizpug\cmd_as_admin.lnk"
        else
                run % w[2]
        WinWait % w[1]
  }
}
If substr(w[1], 1, 3) = "cmd"
        send % "c:\python35\python n:\kivy\trizpug\" w[2] ".py{Enter}“
```

SciTE4AutoHotkey editor/debugger: //fincs.ahk4.net/scite4ahk/

Final Words

- In the immortal words of Raymond Hettinger: “Did anyone learn something new?”
- The last chapter in my first assembler language textbook was titiled “Well?”
  - In essence asking “now that you know the language what are you going to do with it?”