CS 683 Emerging Technologies
Fall Semester, 2005
Doc 15 Python for Series 60 p3

Contents

References

Python for Series 60 Platform API Reference, version 1.1.5

Programming with Python Series 60 Platform, version 1.1.5

Things you need to do

Download Documentation

Python for Series 60 Platform API Reference

Programming with Python for Series 60 Platform

Both are pdf files

Available at
http://www.forum.nokia.com/main/0,,034-821,00.html

Go to bottom of page & click on Python for Series 60 Documentation bundle

Installation Phone Emulator

Step 1. Download the following

http://www.csd.uwo.ca/courses/CS457a/tutorial/Symbian_python_tutorial.zip

Step 2 Follow the instructions in the Environment section

Reading

Run and understand the examples

- ball.py
- filebrowser.py
- default.py

Chapters in Programming with Python

Chapter 2 - Hello World Application

Chapter 4 - GUI Programming

Chapter 6 - Access to File System

Chapter 7 - Logging

Chapter 11 - Handling Key Bindings: RSS Reader

Chapter 12 - Real-Time Graphics Support and Key Event Handling: ball.py

Files

Path separator / or \\

```
f = open("c:/temp/test.txt",'r')
f = open("c:\\temp\\test.txt",'r')
```

Drives

- 'C' is the built-in phone memory
- 'D' is a RAM disk
- 'E' is an extra memory card
- 'Z' is read-only ROM memory

Drives on simulator

```
C:\Symbian\6.1\Series60\Epoc32\Wins\c
C:\Symbian\6.1\Series60\Epoc32\Wins\d
```

Make sure that the above directories are writable

They installed as read-only

File Example

```
import appuifw

fileTest = open(u"c:\\test.txt",'w')
for font in appuifw.available_fonts():
    fileTest.write(font + '\r\n')
fileTest.close()
```

Contents of test.txt

Alpi12	alp17	Aco21	LatinBold19
Albi12	Alb17b	Acalc21	LatinPlain12
Alp13	albi17b	LatinBold12	Acb14
Alpi13	alpi17	LatinBold13	Acb30
Albi13	Aco13	LatinBold17	Acp5

File Browser Example

Example that is installed with simulator

In appendix of Programming with Python for Series 60

Shows how to deal with directories

stdin, stdout & Logging

- stdin
  standard input

  Not clear if it is useable
- stdout
  standard output

  print uses stdout
- stderr - standard error output

Defined in sys module

Not clear what these do on actual phone

Can be redirected

```
import sys
debugfile = open("c:/system/debuginfo.txt","w")
sys.stdout = debugfile
# This will now go to the file
print "Execution reached this far"
```

Logger

```
class Logger:
    def __init__(self, log_name):
        self.logfile = log_name

    def write(self, obj):
        log_file = open(self.logfile, 'a')
        log_file.write(obj + '\r\n')
        log_file.close()

    def writelines(self, obj):
        self.write(''.join(list))

    def flush(self):
        pass
```

placed in
C:\Symbian\6.1\Series60\Epoc32\Release\wins\udeb\z\system\libs

Using the Logger

```
import sys
import logger

my_log = logger.Logger("c:/log.txt")
sys.stderr = sys.stdout = my_log
print "Testing logging"
```

Contents of c:/log.txt

Testing logging

Handling Key Events

Key events are given names

```
EKeyLeftArrow = 0xf807
EKeyRightArrow = 0xf808
EKeyUpArrow = 0xf809
```

For all 288 names see

C:\Symbian\6.1\Series60\Epoc32\Release\wins\udeb\z\system\libs\key_codes.py

Listbox and Key Events

```
def run(self):
    self.setup()
    from key_codes import EKeyLeftArrow
    entries = [u'a', u'b', u'c']
    self.list_box = appuifw.Listbox(entries, self.lbox_observe)
    self.list_box.bind(EKeyLeftArrow, lambda: self.lbox_observe(0))
    self.refresh()
    self.script_lock.wait()
    self.tear_down()
```

Graphics and Key Events

Canvas([redraw_callback=None, event_callback=None])

redraw_callback

method called when canvas needs to be redrawn

event_callback

method called when a key event occurs

Argument to the method is a dictionary

- 'type'
- 'keycode'
- 'scancode'
- 'modifiers'

type

- EEventKeyDown
- EEventKey
- EEventKeyUp

scancode

Physical key itself

Each key has 1 or more scancode

keycode

Each key has zero or more keycode

Result of OS processing of the key

The A key has scancode of 65

The A key has keycode 65 or 91 depending on Shift & Caps lock keys

Helper Code to handle Key events

```
from key_codes import *

class Keyboard(object):
    def __init__(self,onevent=lambda:None):
        self._keyboard_state={}
        self._downs={}
        self._onevent=onevent

    def handle_event(self,event):
        if event['type'] == appuifw.EEventKeyDown:
            code=event['scancode']
            if not self.is_down(code):
                self._downs[code]=self._downs.get(code,0)+1
            self._keyboard_state[code]=1
        elif event['type'] == appuifw.EEventKeyUp:
            self._keyboard_state[event['scancode']]=0
        self._onevent()

    def is_down(self,scancode):
        return self._keyboard_state.get(scancode,0)

    def pressed(self,scancode):
        if self._downs.get(scancode,0):
            self._downs[scancode]-=1
            return True
        return False
keyboard=Keyboard()
```

Sample Use of Keyboard

```
import appuifw
import e32
from graphics import *
class KeyEventExample:
    def __init__(self):
        self.script_lock = e32.Ao_lock()

    def setup(self):
        self.old_title = appuifw.app.title
        appuifw.app.exit_key_handler = self.exit_key_handler
        appuifw.app.screen='full'
        self.image=Image.new((176,208))
        self.keyboard=Keyboard()
        self.canvas=appuifw.Canvas(
                            event_callback=self.keyboard.handle_event,
                            redraw_callback=self.handle_redraw)
        appuifw.app.body=self.canvas

    def run(self):
        self.setup()
        self.move_ball()
        self.script_lock.wait()
        self.tear_down()

    def tear_down(self):
        appuifw.app.title = self.old_title
        appuifw.app.body = None

    def exit_key_handler(self):
        appuifw.app.exit_key_handler = None
        self.script_lock.signal()
```

Sample Use of Keyboard Continued

```
    def move_ball(self):
        location=[75.,50.]
        ball_size=16
        direction = 1
        while True:
            self.image.clear(0)
            if self.keyboard.is_down(EScancodeLeftArrow):
                location[0] -= 1
            if self.keyboard.is_down(EScancodeRightArrow):
                location[0] += 1
            self.image.point((location[0]+ball_size/2,location[1]+ball_size/2),
                  0x00ff00,width=ball_size)
            self.handle_redraw(())
            e32.ao_yield()
            location[1] += direction  * 0.5
            if location[1] > 208 - ball_size/2 or location[1] < ball_size/2:
              direction = -1 * direction

      def handle_redraw(self, rect):
            self.canvas.blit(self.image)

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

Important Graphics Issues

Need to redraw entire image

Don't draw image directly on screen to avoid flicker

Yield to allow UI thread to handle events

```
def move_ball(self):
    location=[75.,50.]
    ball_size=16
    direction = 1
    while True:
        self.image.clear(0)
        if self.keyboard.is_down(EScancodeLeftArrow):
            location[0] -= 1
        if self.keyboard.is_down(EScancodeRightArrow):
            location[0] += 1
        self.image.point((location[0]+ball_size/2,location[1]+ball_size/2),
              0x00ff00,width=ball_size)
        self.handle_redraw(())
        e32.ao_yield()
        location[1] += direction  * 0.5
        if location[1] > 208 - ball_size/2 or location[1] < ball_size/2:
          direction = -1 * direction

  def handle_redraw(self, rect):
        self.canvas.blit(self.image)
```