Lecture 11

Scope
PIL (Image Processing)
GUI Programming

Prof. Steven Ludtke
N410.07, sludtke@bcm.edu

Lab 4 Review

```
#!/usr/bin/env python

import socket
import time

sock=socket.socket()   # default socket

target=raw_input("target address:")
fsp=raw_input("File: ")

fin=file(fsp,"r")
data=fin.read()               # read the entire file into a string

sock.connect((target,40000))  # connect to someone listen()ing
sfile=sock.makefile()         # make a file-like object
sfile.write("{}\n".format(fsp))        # Write the filename
sfile.write("{}\n".format(len(data)))  # Write the file length
sfile.write(data)                      # Write the file contents
sfile.close()
sock.close()
```

Lab 4 Review

- Ask user for address and filename
- read file to be sent to string (provides data and length)
- open connection to remote machine
- Convert socket into file-like object
- send filename, with \n
- send file length, with \n
- send file (exactly matches length)
- close connection

Lab 4 Review

```
import socket
import time
import os

sock=socket.socket()          # default socket
sock.bind(("",40000))         # Nothing magic about 40000
sock.listen(1)                # Wait for 'connect' requests

sock2=sock.accept()           # accept the connection (new socket)
sfile=sock2[0].makefile()     # make a file-like object

fsp=sfile.readline().strip()  # read the filename
fsp=os.path.split(fsp)[1]     # clean up the filename
flen=int(sfile.readline())    # read the file length
out=file(fsp,"w")             # open the file
out.write(sfile.read(flen))   # read from socket directly to file
sfile.close()
sock.close()
```

Lab 4 Review

- Create socket
- bind to port and listen
- accept connection
- convert connected socket to file-like object
- receive filename (readline)
- receive file length as integer
- open file on disk
- read data (length) from network and write to file
- close connection

Scope

What will this produce ?

```
def f(x):
    y=x*10
    return y

x=5
y=6

print y
print f(x)
print y
```

Scope

How about this ?

```
def f():
    y=x*10
    return y

x=5
y=6

print y
print f()
print y
```

Scope

- Local scope
  - Variables defined within a function, exist only within the function
- Global scope
  - Variables defined at the “top level” in the program or module.
  - Variables declared using the “global” keyword
- Built-in names
  - Built in functions and variable names

Scope

What will this produce ?

```
def f(x):
    global y
    y=x*10
    return y

x=5
y=6

print y
print f(x)
print y
```

PIL

Creating Images

```
from PIL import Image

im=Image.new("RGB", (512, 512), "white")

im=Image.open("file.jpg")                  # from a file
im.save(filename,[format],[options])  # write to a file

a=numpy.fromfunction(lambda x,y:sin(x/10.0)*cos(y/10.0),(128,128))
im=Image.fromarray(a)
```

PIL - File Formats*

Fmt	Bits	Loss	Cmpr	Notes
BMP	8 or less			
GIF	8 total, cmap	X	X	
IM	all modes !			LabEye & IFUNC
JPEG	8	X	X	
PCX	8 or less			
PNG	8		X	
PPM	8			PBM,PGM,PPM
TIFF	8	R	R	
EPS				Needs GS to read most
PDF				Write only

* - Only the most common ones

PIL Attributes

- im.format
- im.mode
- im.size
- im.info

PIL

simple image processing

```
im.show()                          # machine specific display
im2=im.convert(mode)               # Change mode, i.e. RGB->L
im.resize((width,height))          # resize image in-place
im.thumbnail((width,height)) # resize to fit in a box
im.rotate(angle)                   # CCW in degrees
im2=im.crop((left,upper,right,lower))
im.paste(im2,(left,upper))

pix=im.load()                      # for pixel access
pix[x,y]                           # access pixel at x,y
```

PIL

- ImageChops - invert(a), lighter(a,b), darker(a,b), add(a,b), subtract(a,b), difference(a,b), screen(a,b)
- ImageDraw
  - a=Image.new(“RGBA”,(128,128))
  - draw=ImageDraw.Draw(a)
  - draw.line((x0,y0,x1,y1),fill=”red”), point, rectangle, arc, chord, ellipse, text
- ImageFilter - BLUR, CONTOUR, DETAIL, EDGE_ENHANCE, EDGE_ENHANCE_MORE, EMBOSS, FIND_EDGES, SMOOTH, SMOOTH_MORE, SHARPEN
- etc.

Using Numpy

```
from numpy import *

from PIL import Image

a=fromfunction(lambda x,y:sin(x/10.0)*cos(y/10.0),
(128,128))

im=Image.fromarray(a)

im.show()              # Image is black !?!?

a+=1

a*=127

im2=Image.fromarray(a)

im2.show()
```

PIL

```
a=fromfunction(lambda x,y:(127+sin(x/100.)*127., 127+cos(y/
100.)*127.,127+sin(x/500.)*127.),(256,256))

b=dstack(a)

c=b.astype(uint8)

im=Image.fromarray(c)

im.show()
```

PIL

```
im=Image.open("hh.jpg")

a=array(im)

a[0,0]

im2=Image.fromarray(a)
```

Images in MatPlotLib

```
ipython -pylab

im=fromfunction(lambda x,y:sin(x/10.0)*cos(y/10.0),(64,64))

imshow(im)

imshow(im,cmap=cm.gray)

savefig("a.png")
```

GUI Programming

Standard

Best (IMHO)

- Tkinter, PyQt, PyGTK, wxPython, FXPy, PyOpenGL*

- widget - A graphical object, like a button or a slider
- callback - a function which is called when the user interacts with a widget
- geometry or layout manager - controls where widgets are displayed

* - just 3-D graphics, no widgets

Tkinter

- ‘standard’ Python GUI toolkit
- Python interface elegant, but built on top of Tcl/Tk
- A bit clunky and slow, but has been used to build some very large applications (eg - Chimera)
- If you have a choice, for larger projects,
  use PyQt4 (just my suggestion)
- http://www.pythonware.com/library/tkinter/introduction/index.htm
- Tkinter extended by PMW and Tix

Modal Widgets

- Get specific info from the user without writing a full GUI for the program.
- or can be used as part of a full GUI.

- tkFileDialog
- tkMessageBox
- tkColorChooser

tkFileDialog

- import tkFileDialog
  - askdirectory(**options)
  - askopenfile(mode='r', **options)
  - askopenfilename(**options)
  - askopenfilenames(**options)
  - askopenfiles(mode='r', **options)
  - asksaveasfile(mode='w', **options)
  - asksaveasfilename(**options)

tkMessageBox

- import tkMessageBox
  - askokcancel(title=None, message=None, **options)
  - askquestion(title=None, message=None, **options)
  - askretrycancel(title=None, message=None, **options)
  - askyesno(title=None, message=None, **options)
  - showerror(title=None, message=None, **options)
  - showinfo(title=None, message=None, **options)
  - showwarning(title=None, message=None, **options)

tkColorChooser

- import tkColorChooser
  - askcolor(color=None, **options)

Tkinter

- Event driven programming
  - Set up all of your widgets
    - Create widget
    - Set callbacks
    - Place widget in window
  - Call the event loop
  - Cleanup

```
from Tkinter import *
root = Tk()          # Initializes Tkinter

###setup widgets

root.mainloop()     # Runs the GUI until the user triggers an exit
root.destroy()      #  Cleanup
```

Simple Tkinter

```
from Tkinter import *

root = Tk()

w = Label(root, text="Hello, world!")
w.pack()

root.mainloop()
```

Tkinter Widgets

- BitmapImage
- Button
- Canvas
  - Arc, Bitmap, Image, Line, Oval, Polygon, Rectangle, Text
- Checkbutton
- Entry
- Font
- Frame (window)
- Label
- Listbox
- Menu/Menubutton
- Message
- PhotoImage
- Radiobutton
- Scale
- Scrollbar
- Text
- Toplevel Widget

Tkinter Misc

- DoubleVar
- IntVar
- StringVar

- SimpleDialog

- tkFont

- For Callbacks, use:
- command, after, bind

Geometry Managers

- Grid Geometry Manager
  - Arrange widgets like a table
- Pack Geometry Manager
  - Arrange widgets sequentially into the available space
- Place Geometry Manager
  - Explicitly position widgets - tricky

Button Callback Example

```
from Tkinter import *
root = Tk()

def pushed(): print "You pushed me too far!"

w = Button(root, text="Push Me",command=pushed)
w.pack()

root.mainloop()
```

Entry Example

```
import Tkinter as tk
root = tk.Tk()

def enter(): print "You entered: ",str(v.get())

l=tk.Label(root,text="Enter something:")
l.pack()

v=tk.StringVar()
w = tk.Entry(root, width=40,textvariable=v)
w.pack()
v.set("Start")

w2 = tk.Button(root, text="Push Me",command=enter)
w2.pack()

root.mainloop()
```

Timer Callback Example

```
from Tkinter import *
root = Tk()

def timeout(): print "It's Time !"

w = Button(root, text="Push Me")
w.pack()
w.after(3000,timeout)

root.mainloop()
```

Full Tkinter Example

```
import Tkinter as tk
import tkMessageBox

def say_hi():
   tkMessageBox.showinfo("Hi","Hello There !")

root = tk.Tk()

QUIT = tk.Button(root)
QUIT["text"] = "QUIT"
QUIT["fg"]   = "red"
QUIT["command"] = root.quit
QUIT.pack({"side": "left"})

hi_there = tk.Button(root)
hi_there["text"] = "Hello",
hi_there["command"] = say_hi
hi_there.pack({"side": "left"})

root.mainloop()
root.destroy()
```

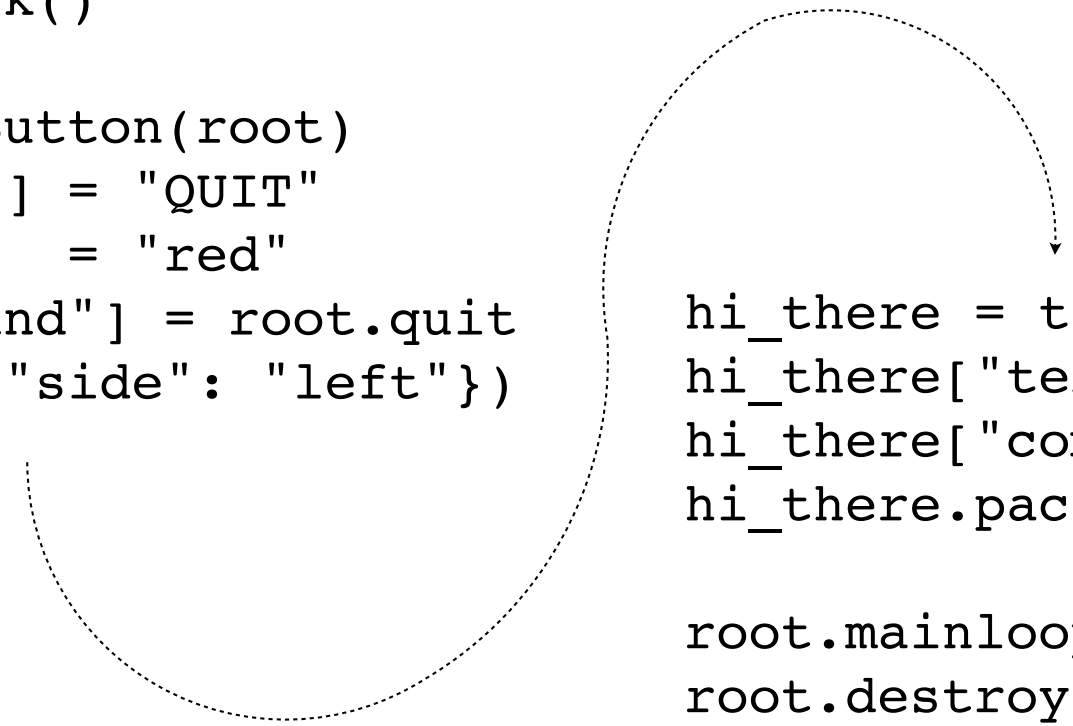

Photo Viewer

```
import Tkinter as tk
from PIL import Image, ImageTk
import os

files=os.listdir(".")
files=[fsp for fsp in files if fsp[-4:].lower()==".jpg"]

root=tk.Tk()
curimg,curphoto,curn=None,None,0

def nextbut():
        global curn,curimg,files,lbl,curphoto

        curn+=1
        curimg=Image.open(files[curn])
        curimg.thumbnail((1024,768))
        curphoto = ImageTk.PhotoImage(curimg)
        lbl["image"]=curphoto

lbl=tk.Label(root)
lbl.pack()

nextbut = tk.Button(root, text="Next", command=nextbut)
nextbut.pack()

quitbut = tk.Button(root,text="Quit",command=root.quit)
quitbut.pack()

root.mainloop()
root.destroy()
```

tkinter References

- http://www.pythonware.com/library/tkinter/introduction/index.htm
- http://infohost.nmt.edu/tcc/help/pubs/tkinter.pdf
- A few kindle books are available as well. “Python and Tkinter Programming” out of print

~~Homework 6~~

- No more homework this term. Class projects are due two weeks from tomorrow. Presentations two weeks from Monday!
- A practice is available on the website if you want to try your hand at GUI programming, but there is nothing to turn in.