Lecture 12

Graphical User Interfaces (GUIs)

Prof. Steven Ludtke
N420, sludtke@bcm.edu

Debugging

- print statements ?
- traceback module: print_exc()

Debuggers

- Jupyter:
  - import pdb; pdb.set_trace()
- Spyder in Anaconda, built in editor/debugger!

- https://wiki.python.org/moin/PythonDebuggingTools

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)
- http://www.pythonware.com/library/tkinter/introduction/index.htm
- Tkinter extended by PMW and Tix
- If you have a choice, for larger projects,
  use PyQt4 (just my suggestion)

Modal Widgets

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

- tkFileDialog
- tkMessageBox
- tkColorChooser

tkFileDialog

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

tkMessageBox

- from tkinter import messagebox
  - 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

- from tkinter import colorchooser
  - 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
from tkinter import messagebox

def say_hi():
    messagebox.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=sorted(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()
```

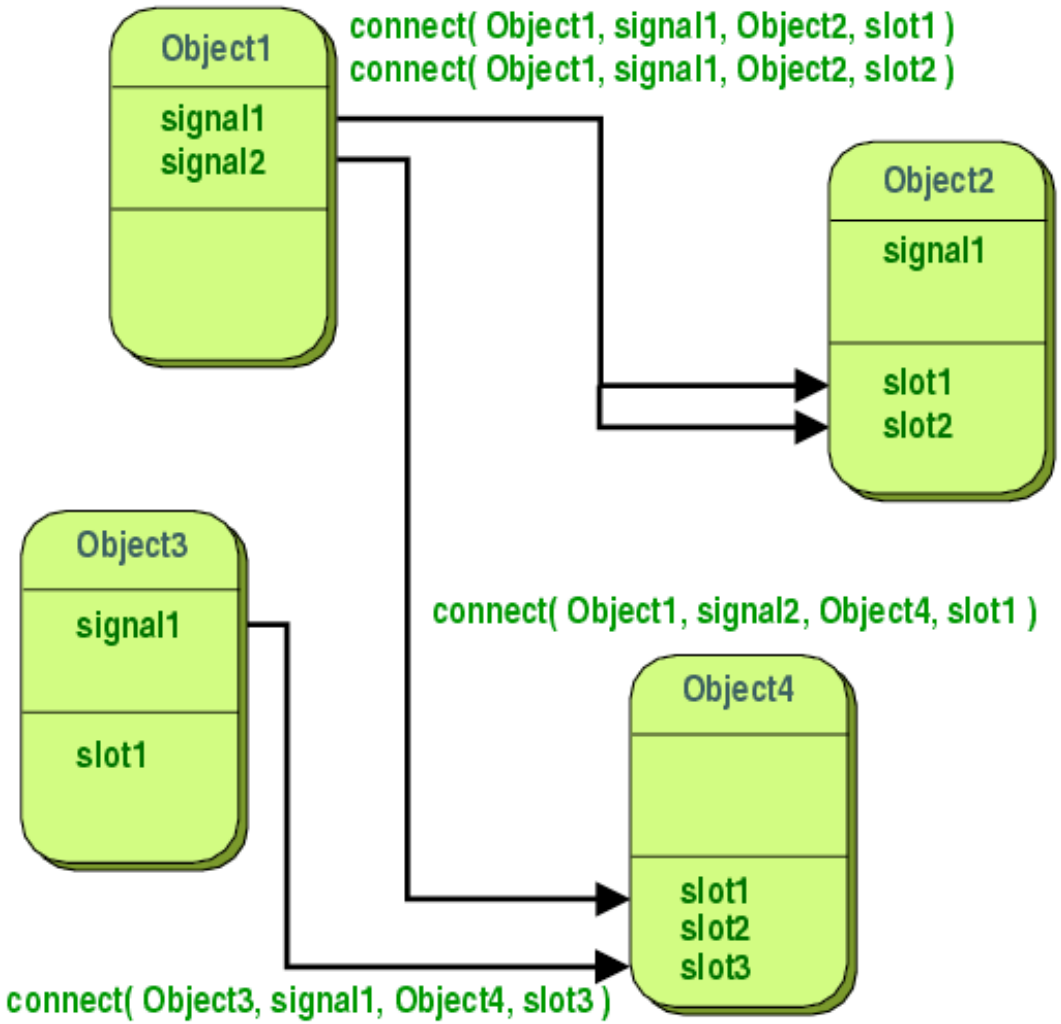
tkinter References

- https://pythonprogramming.net/python-3-tkinter-basics-tutorial/
- https://www.tutorialspoint.com/python3/python_gui_programming.htm
- Google will give you many more...

Qt 5.x

- Qt:
  - http://www.qt.io/
  - Docs: http://doc.qt.io/qt-5/reference-overview.html
  - Ref: http://doc.qt.io/qt-5/classes.html
- PyQt:
  - http://www.riverbankcomputing.co.uk/software/pyqt/intro

Signals and Slots



Simple Qt5 Application

```
from PyQt5.QtWidgets import QApplication, QWidget

# This is a class representing the main window for the application
class MyGuiWindow(QWidget):
    def __init__(self,parent=None):
        QWidget.__init__(self,parent)
        # setup widgets

    def respond(self,value):
        pass
        # do something

# This is the actual program.
# Create an Application object, set up widgets, and exec()
app = QApplication([])
window = MyGuiWindow()
window.show()
app.exec()
```

Button

- Public Slots
    - void animateClick (int msec = 100)
    - void click ()
    - void setChecked (bool)
    - void setIconSize (const QSize & size)
    - void toggle ()

- Signals
    - void clicked (bool checked = false)
    - void pressed ()
    - void released ()
    - void toggled (bool checked)

Simple Qt4 Application

```
from PyQt5.QtWidgets import *

# This is a class representing the main window for the application
class MyGuiWindow(QWidget):
    def __init__(self,parent=None):
        QWidget.__init__(self,parent)
         # organizes the widgets into a grid
        self.gbl = QGridLayout(self)

        # create a PushButton and add it to the window
        self.but = QPushButton("Push Me")
        self.gbl.addWidget(self.but,0,0)

        # connect the 'clicked' signal to the respond() method
        self.but.clicked.connect(self.respond)

    def respond(self,value):
        QMessageBox.information(None,"Ouch","That hurt!  Why did you do that?")

# This is the actual program.
# Create an Application object, set up widgets, and exec()
app = QApplication([])
window = MyGuiWindow()
window.show()
app.exec()
```

Graphical Layout Design

- Qt Designer - GUI design (look in anaconda/bin)
- uic - Build C++ code from designs
- pyuic5 - Build python code from designs

- Gallery: http://doc.qt.io/qt-5/gallery.html

Simple Python Webserver

```
# This will serve files from the current directory
# we use port 8080 because port 80 is restricted

from http.server import *

httpd = HTTPServer(("",8080),SimpleHTTPRequestHandler)
httpd.serve_forever()
```

Scripting, Server vs. Client

- Serverside scripting depends on the webserver you use
  - Many choices
  - May put load on server
- Clientside
  - Java - often available, but many issues
  - Flash - Almost ubiquitous, but rapidly fading
  - HTML5 - provides many dynamic capabilities
  - Javascript built in to most browsers
    - AJAX - Asynchronous Javascript and XML
    - AJAJ - Asynchronous Javascript and JSON

Within Jupyter

Embed simple HTML, but limited:

```
from IPython.core.display import display, HTML

display(HTML('<h1>Hello, world!</h1>'))
```

Javascript - Button

```
<HTML><HEAD><TITLE>Hi there</TITLE></HEAD>
<BODY>
<h3>Here is a title</h3>
And some text
<p>
<input type="button" value="Push Me" onclick="alert('You pushed me too far')">
</p>
</body>
```

Javascript - mouseover

```
<HTML><HEAD><TITLE>Hi there</TITLE></HEAD>
<BODY>
<h3>Here is a title</h3>
And some text
<p>
<a href="index3.html" onmouseover="window.document.bgColor='red'">Red</a>
<a href="index3.html" onmouseover="window.document.bgColor='green'">Green</
a>
<a href="index3.html" onmouseover="window.document.bgColor='blue'">Blue</a>
<a href="index3.html" onmouseover="window.document.bgColor='white'">White</
a>
</p>
</body>
```

Javascript Calculator

```
<HTML><HEAD><TITLE>Hi there</TITLE></HEAD>
<BODY>
<h3>Calculator</h3>
<input type=text name='data' onkeypress='compute(event)' />
<br><input type=text name='result' readonly=true />
<script>
function compute(event) {
if (event.keyCode!=13) { return; }
data=document.getElementsByName('data')[0];
result=document.getElementsByName('result')[0];
result.value=eval(data.value);
}
</script>
</body>
```

Javascript - Calculator #2

```
<HTML><HEAD><TITLE>Hi there</TITLE></HEAD>
<BODY>
<h3>Calculator</h3>
<form name=calc onsubmit=compute()>
<input type=text name=data value="0"></input>
<table><tr>
<td><input type="button" value="7" onclick="num('7')"></td>
<td><input type="button" value="8" onclick="num('8')"></td>
<td><input type="button" value="9" onclick="num('9')"></td>
<td><input type="button" value="X" onclick="fn('*')"></td></tr><tr>
<td><input type="button" value="4" onclick="num('4')"></td>
<td><input type="button" value="5" onclick="num('5')"></td>
<td><input type="button" value="6" onclick="num('6')"></td>
<td><input type="button" value="-" onclick="fn('-')"></td></tr><tr>
<td><input type="button" value="1" onclick="num('1')"></td>
<td><input type="button" value="2" onclick="num('2')"></td>
<td><input type="button" value="3" onclick="num('3')"></td>
<td><input type="button" value="+" onclick="fn('+')"></td></tr><tr>
<td colspan=3><input type="button" value="0" onclick="num('0')"></td>
<td><input type="button" value="=" onclick="eql()"></td>
</tr> </table> </form>
```

Javascript - Calculator #2

```
<script>
xpr=""
rst=1
function num(val) {
	xpr+=val
	if (rst) {
		rst=0
		document.calc.data.value=""
	}
	document.calc.data.value+=val
}

function fn(val) {
	xpr+=val
	rst=1
}

function eql() {
	document.calc.data.value=eval(xpr)
	xpr=""
	rst=1
}
</script>
</body>
```

Javascript - Statements

```
var name[=value],name[=value]
function f(x,y) statement
if (expression) statement; else statement;
do statement while (expression)
while (expression) statement
for ( var in array ) statement
for (init; update; test) statement
switch (expr) {
case const:
        statements
        break
default:
        statements
}
```

Javascript - Events

- onclick
- onfocus, onblur
- onmousedown, up, move, over,out
- onkeydown, up, press
- onreset
- onsubmit
- onload, unload

References

- http://www.w3.org/TR/html4/
- http://www.w3.org/TR/html4/index/elements.html
- http://htmlhelp.com/reference/html40/olist.html

- http://www.javascriptkit.com/jsref
- http://www.w3schools.com/jsref/default.asp

CLASS PROJECTS

- Must do something useful in some specific context
- Not be trivial, or readily available with the same functionality (without approval)
- If you have past programming experience I will expect more

- **Please follow these instructions exactly:**
- Your class project MUST be submitted by 11:59 PM on Sat, Feb 24. No revisions will be accepted after this time. You can use Sunday to prepare your oral presentation
- Your submission should consist of:
- one or more .py files (should have sufficient comments to figure out how they work)
- any necessary additional files to demonstrate that the program works
- A 1-2 page PDF file with a brief description of your program, what inputs the program takes, what outputs the program produces, and what it is supposed to do.
- The final item in the PDF should be a command-line to use in running the program, and any necessary instructions to demonstrate that it works.
- Combine all files into a .zip or .tgz file named: Familyname_Givenname_project_2018.zip
- Email **sludtke42@gmail.com** with the subject "Class project submission", and attach the .zip file. If the file is too large for email, feel free to transfer it via Box, DropBox, etc.

CLASS PROJECT PRESENTATIONS

- Monday, Feb 26
- **8 AM** - 10:30
- **First slide:** Program Title, Your Name, Department/Graduate Program
- You will have 6 minutes total:
  - Set up your presentation (1 minute) - TEST LAPTOP BEFORE FEB 26!!!
  - Give your talk (4 minutes. practice!)
    - What does your software do, and why did you write it
    - Inputs and outputs
    - Demonstration (mock demonstration if it takes too long)
  - Questions (1 minute)
  - Class projects are 1/2 of your final grade in the class.
- The program MUST WORK to get a good grade. Better to turn in something that doesn't do everything you wanted, but works, than something broken

CLASS PROJECT RUBRIC

- Project (2/3 project grade):
  - Program works (2 pt)
  - Does something useful (1 pt)
  - Programming style (1/2 pt)
  - Followed all instructions (1/2 pt)
- Presentation (1/3 project grade):
  - Presentation Clarity (2 pt)
  - Program Demonstrated Effectively (1 pt)
  - Presentation Organization (1/2 pt)
  - Followed all instructions, including projector issues (1/2 pt)