Scientific Computing: Lecture 25

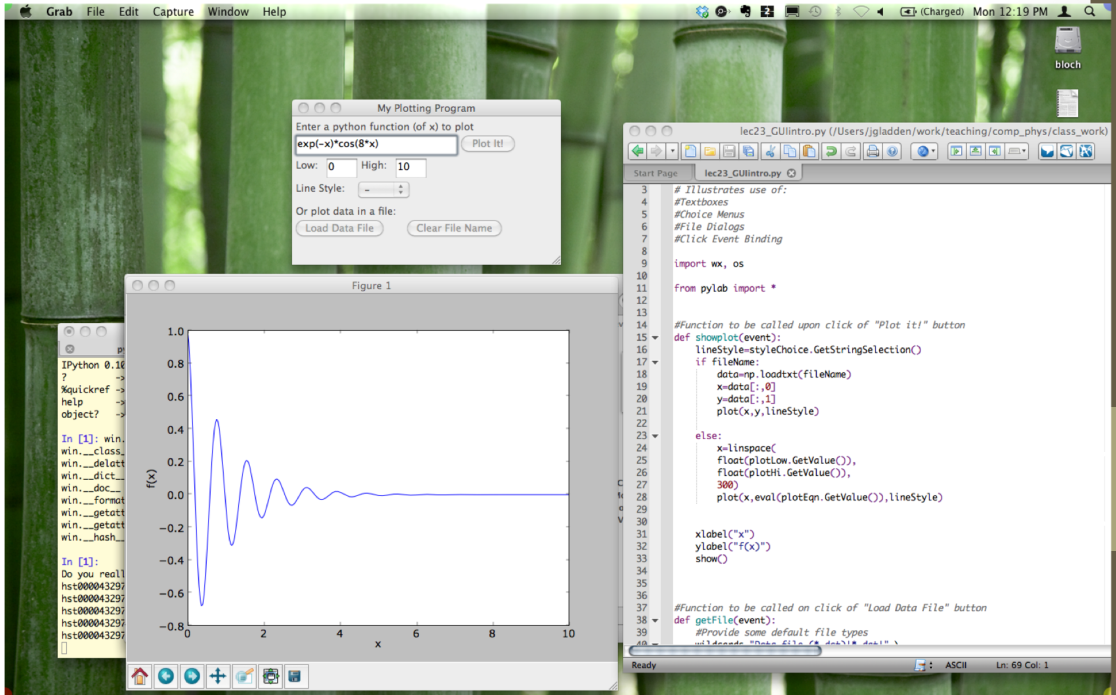
- General Introduction to GUI programming
- GUI Modules for Python
- wxPython widgets and events
- Example: Plot a function or data
- Box Sizers

CLASS NOTES

- WORK ON PROJECTS!
- Ask questions!
- Due date is Thursday evening of exam week (NO LATER!)
- Turn in to Box folder

GUI Programming

- GUI: Graphical User Interface
  - Windows, menus, buttons, listboxes,…
  - The "normal" mode of programming
- Most scientific programs are focused on correctly and efficiently "crunching the numbers" and do not use/need and GUI interface.
- Sometimes it does enhance the usability of your program.



My Plotting Program

Enter a python function (of x) to plot: exp(-x)*cos(8*x) [Plot It!]

Low: 0 High: 10

Line Style: -

Or plot data in a file:

[Load Data File] [Clear File Name]

```
IPython 0.10
?          ->
%quickref ->
help       ->
object?    ->

In [1]: win.
win.__class_
win.__delatt
win.__dict__
win.__doc__
win.__format
win.__getatt
win.__getatt
win.__hash__

In [1]:
Do you reall
hst000043297
hst000043297
hst000043297
hst000043297
hst000043297
```

lec23_GUIintro.py (/Users/jgladden/work/teaching/comp_phys/class_work)

```python
# Illustrates use of:
#Textboxes
#Choice Menus
#File Dialogs
#Click Event Binding

import wx, os

from pylab import *


#Function to be called upon click of "Plot it!" button
def showplot(event):
    lineStyle=styleChoice.GetStringSelection()
    if fileName:
        data=np.loadtxt(fileName)
        x=data[:,0]
        y=data[:,1]
        plot(x,y,lineStyle)

    else:
        x=linspace(
        float(plotLow.GetValue()),
        float(plotHi.GetValue()),
        300)
        plot(x,eval(plotEqn.GetValue()),lineStyle)


    xlabel("x")
    ylabel("f(x)")
    show()



#Function to be called on click of "Load Data File" button
def getFile(event):
    #Provide some default file types
```

Ready ASCII Ln: 69 Col: 1

GUI Modules for Python

- Python has several well supported GUI modules.
- Tkinter
  - Comes with “plain vanilla” python – all python users will definitely have it installed.
  - Oldest, but a but outdated and “clunky”.
- wxPython
  - Needs to be installed.
  - Versions of all major platforms (MS Windows, Mac, Linux)
  - Rich set of widgets (sliders, gauges, graphics)
  - Currently popular, native feel (OSX vs Linux, vs MS Windows)
  - Same code works on all platforms!

GUI Modules for Python

- Python Win
  - MS Windows only
  - Uses Windows specific GUI features
- PyGTK and Py QT
  - Built on GTK and QT windowing libraries (these need to be installed first).
  - Cross platform
  - Very appealing graphics and layout
  - Popular for Linux developers.
- We'll focus on wxPython since it is most popular, cross platform, and well supported. Comes as part of EPD/Canopy.

wxPython basics

- First thing to do is create an application and a window.
- Then create 'widgets' to get/provide information from/to the user and ways to make the program do something.
  - Text boxes, buttons, menus, checkboxes, slider bars, progress bars, dialog boxes, …
- Next, (some) widgets need to be 'bound' to python functions which perform some task (plot some data, run a simulation, get a file, connect to a server,…)
- Often useful to 'sketch' out what you want the program to look like before starting.
- There are some "drag and drop" IDEs to make layout easier. (BOA Constructor, Eclipse, wxGlade)

First Steps: Create an App and Frame

- An ‘app’ must be created 1st: `app=wx.PySimpleApp()`
  - We’ll use “PySimpleApp()” for these little programs. wx.App() offers more control options.
- An app can have multiple windows (or Frames), but must have at least one.
  `win = wx.Frame(None, title=“My Window”, size=(500,300))`
- Once the Frame is created, we need to display it: `win.Show()`, a blank window would appear.
- Now we can start creating widgets and placing them in the Frame.

Basic Widgets

- Text Labels: use the StaticText class
  - `label1 = wx.StaticText(win,-1,”Text to Display”, pos=(5,15))`
  - The ‘-1’ is an ID number. You can use a unique number here if you will need to refer to this widget later. But you have named this object ‘label1’ and can do most of what you need to do with that.
- Buttons: Click and do something
  - `mybutton=wx.Button(win,label=‘Become Smarter!’, pos=(100,25))`
  - Now “bind” this button to a python function to do something
  - `mybutton.Bind((wx.EVT_BUTTON, makeMeSmarter)`

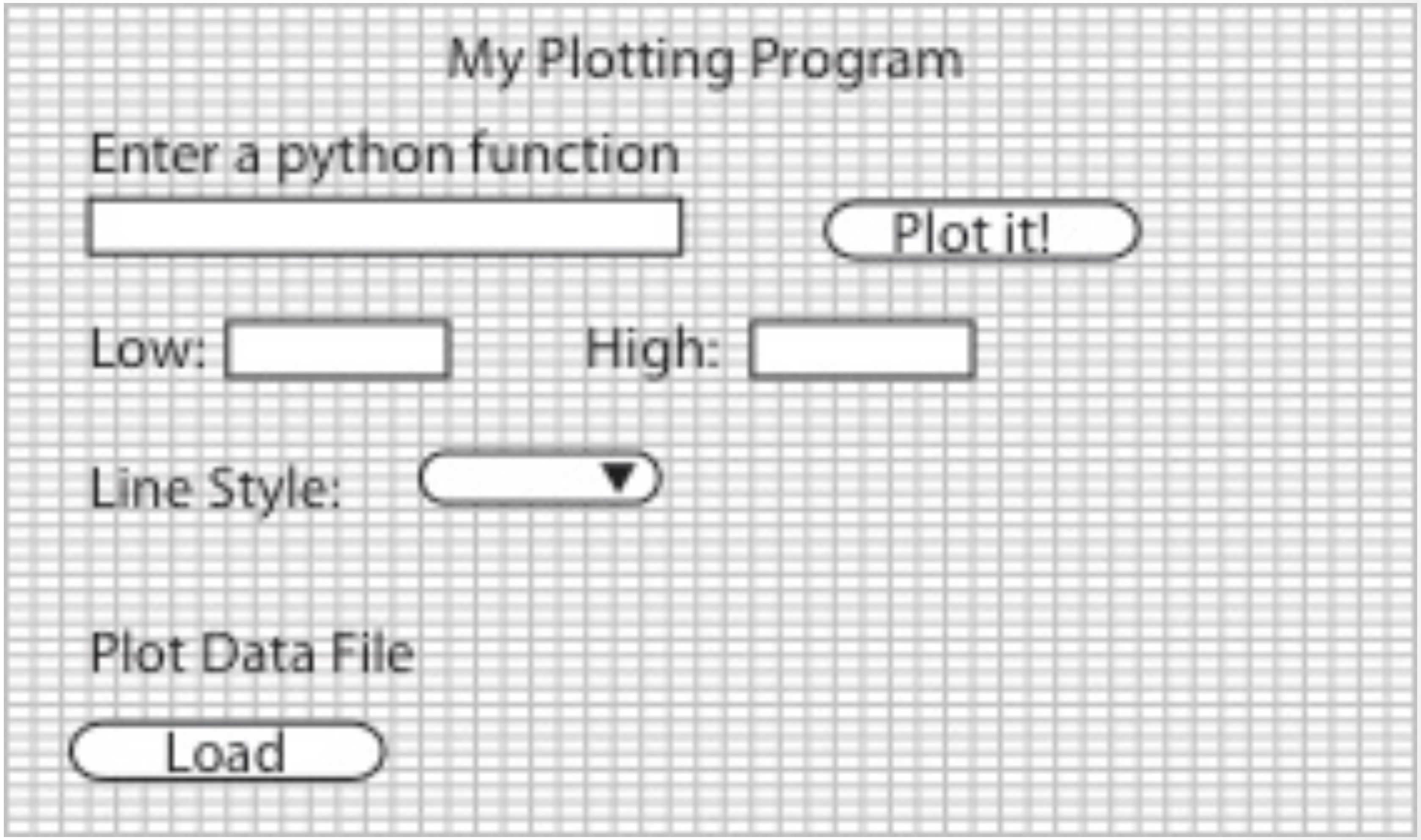
Basic Widgets

- Text Input Box
  - `minXinput = wx.TextCtrl(win, pos=(50,25), size=(40,25))`
  - Get the contents of that box by:
    - `x=minXinput.GetValue(),` This returns a string!
  - Set content for the box by:
    - `minXinput.SetValue(“-10”)`
- List Choices: Require a choice from specific choices
  - `colorList=[“blue”,”red”,”green”,”yellow”]`
  - `colorChoice = wx.Choice(win,-1,(50,70),choices=colorList)`
  - `currentColor = colorChoice.GetStringSelection()`

Example: plot function

- As a simple example, let’s make a small GUI app to generate a plot with the following features:
  - Read a python function to plot
  - Read in high and low values for independent variable (x)
  - Select a line style for the plot
  - Provide and option to plot data in a file rather than a function.
  - Leverage existing features of pylab to do plotting.
  - Pylab plots can be embedded using matplotlib in the windows (not done here).
- Make a layout sketch, then on to the code…

Example: Layout sketch



GUI Layout: Box Sizers

- The way we positioned widgets last time is called "absolute positioning".
- Most GUI frameworks offer some sort of "relative" positioning by placing collections of widget in a container.
- wxPython uses several such containers: box sizers, static sizers, grid sizers (several types).
- Box sizers are most common. They can either be vertically or horizontally oriented.
- You can nest a box sizer inside another box sizer.
- Once the sizers are in place, you add widgets to them.

Box Sizers: Basic procedure

- Create a panel for the sizers: `panel1=wx.Panel()`
- Create vertical box sizer with:

```
vbox  = wx.BoxSizer(wx.VERTICAL)
```

- Create a horizontal box sizer with

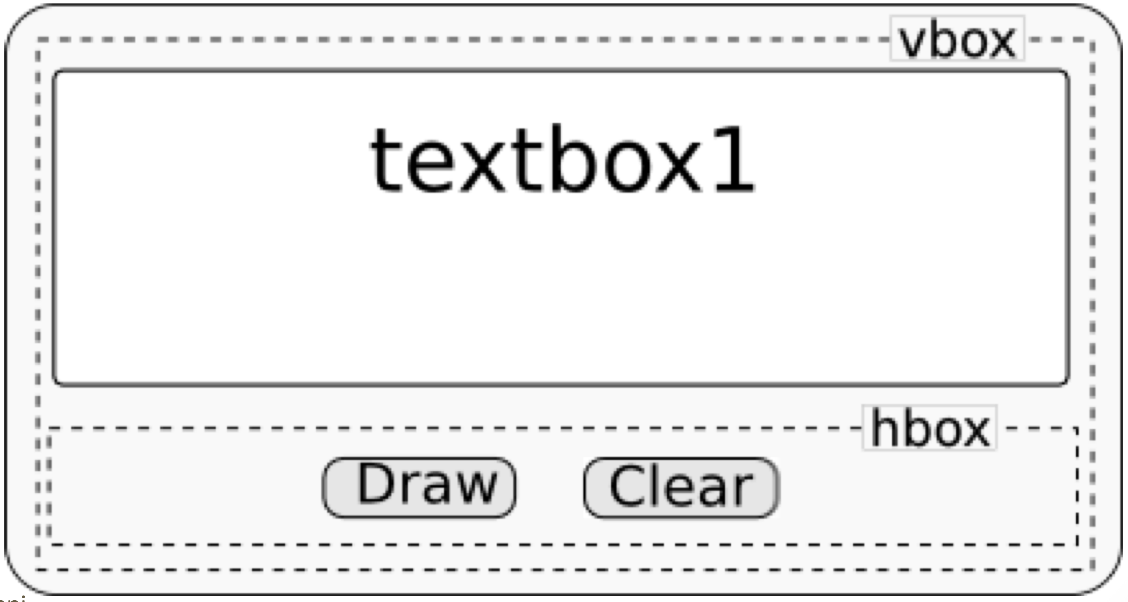
```
hbox = wx.BoxSizer(wx.HORIZONTAL)
```

- Add some widgets (widgets must already be created):

```
vbox.Add(textbox1, 1 , border = 3, wx.LEFT |
    wx.TOP | wx.GROW)
hbox.Add(drawbutton,0,wx.ALIGN_CENTER | wx.ALL
    | wx.ALIGN_CENTER_VERTICAL)
hbox.Add(clearbutton,0,wx.ALIGN_CENTER |
    wx.ALL | wx.ALIGN_CENTER_VERTICAL)
```

Stitch it all together

- Add hbox to vbox
  `vbox.Add(hbox,0,wx.ALIGN_LEFT | wx.TOP)`
- Add vbox to panel: `panel.SetSizer(vbox)`
- Make everything fit OK: `vbox.Fit()`


vbox
textbox1
hbox
Draw
Clear