GUIs

Toolkits

- Toolkits provide the set of widgets that make up a GUI
- Typical program design:

```
class MyGUI():
   def __init__(self, parent):

      # setup the widgets here, storing the widget objects
      # as members of self

   def callback(self):
      # a call back is invoked, for instance, when a button
      # is pressed


root = Toolkit_Object()

my_gui = MyGUI(root)
root.event_loop()
```

- All GUIs have an event loop, provided by the GUI—it's job is to respond to key presses, button pressed, etc., and take the action you defined in constructing the GUI (usually calling one of your callback functions)

Toolkits

- Different toolkits have different goals
  - Some will look the same on every platform, some will try to look native of different platforms
  - Core design issues are still the same
- tkinter
  - Default toolkit for python—it comes with python
- GTK+
  - GIMP toolkit—it is the foundation for the GNOME desktop environment
  - written in C, but bindings for C/C++, Python, Javascript, Perl, …
- Qt
  - Basis for the KDE desktop environment
  - Bindings for Python, C++, C#, Javascript, Ruby, …

Interface Designer

- Some toolkits have GUIs to design your GUI
  - Typically they output a text description of the elements that make up your GUI and their placement (e.g. XML)
  - You load this description when and invoke a builder that draws the GUI in your GUI class `__init__`
- GTK+ has `glade`
- Qt has `Qt Designer`
- tkinter...
  - doesn't seem to be standard GUI builder
  - see, e.g. `pygubu`

tkinter basics

- A widget doesn't appear until you use a geometry manager to describe how it should be placed
- Geometry managers
  - `grid`: uses a grid to layout widgets
  - `pack`: puts widgets next to each other, window will shrink to fit widgets
  - `place`: specify the widget in terms of coordinates (absolute or relative)
- Only one geometry manager per container
  - Use multiple frames in a single widget to mix
  - A `frame` also needs to be placed via a geometry manager

http://effbot.org/zone/tkinter-geometry.htm
http://userpages.umbc.edu/~dhood2/courses/cmsc433/spring2010/?section=Notes&topic=Python&notes=94

Grid

- Within a Grid manager, you can do

  `my_widget.grid(row=0, column=1)`

- Options
  - `sticky` to set the centering in a cell
    - `W` means left
    - `W+E` means stretch it to fill the cell horizontally
    - `W+E+N+S` means stretch in all directions
  - `columnspan` or `rowspan` to have it span cells
  - `padx` and `pady` (and `ipadx` and `ipady`) to add padding

http://effbot.org/zone/tkinter-geometry.htm
http://userpages.umbc.edu/~dhood2/courses/cmsc433/spring2010/?section=Notes&topic=Python&notes=94

Pack

- With the pack manager, you can do

  `my_widget.pack()`
- This is good for putting everything into a single row or column (think of a toolbar, for example)
  - `side` can set the placement
    - `Tk.LEFT`, `Tk.RIGHT`, `Tk.TOP`, `Tk.BOTTOM`
  - `fill` can cause the widget to stretch to fill the container

http://effbot.org/tkinterbook/pack.htm