A proposal for your Final Project is due along with HW 10
Details are up on the course website
1. Graphical User Interface
   - Tkinter
   - Other Graphics Modules

2. Text User Interface
   - curses
   - Other Text Interface Modules
Tkinter

- The module tkinter is a wrapper of the graphics library Tcl/Tk
- Why choose tkinter over other graphics modules
  - It’s bundled with Python so you don’t need to install anything
  - It’s fast
  - Guido van Rossum helped write the Python interface
- The docs for tkinter aren’t that good
  - The docs for Tk/Tcl are much better
  - Tk/Tcl functions translate well to tkinter
  - It’s helpful to learn the basic syntax of Tk/Tcl
- Tk/Tcl syntax → Python:
  - `class .var_name -key1 val1 -key2 val2` → `var_name = class(key1=val1, key2=val2)`
  - `.var_name method -key val` → `var_name.method(key=val)`
import tkinter as tk

class SomeApp(tk.Frame):
    def __init__(self, master=None):
        tk.Frame.__init__(self, master)

def main():
    root = tk.Tk()
    app = SomeApp(master=root)
    app.mainloop()

if __name__ == '__main__':
    main()
tkinter has a bunch of widgets
- Button, Label, Listbox, Radiobutton

Create a widget with `b = tk.Button(parent)`
- `parent` is the containing widget

Options can be accessed and set dictionary style
- `b['text'] = 'Press Me'`
- equivalently: `b = tk.Button(parent, text='Press Me')`
Displaying Widgets

- Just creating a widget will not display it.
- The widget must be told where to go in the parent widget.
  - `some_widget.grid()` puts the widget in the parent.
  - `grid(row=r, column=c)` gives more control over placement.
- The height of a row and width of a column are determined by the largest widget in that row/col.
- Any row/col with no widgets has size 0 and is skipped.
By default, tkinter only supports bitmap, gif, and ppm/pgm images.

More images are supported with Pillow.

Pillow is a fork of Python Imaging Library. You can install it with:

```
pip install pillow
```

```python
from PIL import Image, ImageTk
```

Create a PIL image with:

```python
img = Image.open('path_to_img')
```

Make a Tk image with:

```python
tk_img = ImageTk.PhotoImage(im)
```

Set it as an attribute in tkinter with:

```python
b[‘image’] = tk_img
```
A Widget can have a registered **Event Handler**

The handler is a function that gets called when the widget is used.

Register a handler for a widget:

```python
b[‘command’] = some_function
```

Handlers do not automatically say what widget was used.

- Use a lambda to partially apply arguments:
  ```python
  b[‘command’] = lambda w: button_handler(w)
  ```
- Or use `from functools import partial`:
  ```python
  b[‘command’] = partial(button_handler, w)
  ```
Graphical User Interface
- Tkinter
- Other Graphics Modules

Text User Interface
- curses
- Other Text Interface Modules
WxPython

- Similar to `tkinter` in that it wraps an existing library
- Wraps the C++ `wxWidgets`
- A little bit more user-friendly
- Not yet supporting Python3
  - `wxPython Phoenix` is the project porting to Python 3
  - It’s partially finished (only `wx core` has been ported)
1. Graphical User Interface
   - Tkinter
   - Other Graphics Modules

2. Text User Interface
   - curses
   - Other Text Interface Modules
curses is a wrapper around the ncurses library

ncurses is the standard for terminal graphics
- Is terminal independent (XTerm, Command Prompt, ...)
- Treats screen as a grid of characters
- Pretty low-level

An **curses** program runs in your current terminal
- Not a new window
- Debugging with **print** statements can cause weird behavior
- Changes made in the program can persist after termination
- **curses.wrapper** ensures that clean-up happens on termination
import curses

class SomeApp(object):
    def __init__(self, stdscr):
        self.stdscr = stdscr

    def run(self):
        while True:
            key = self.stdscr.getch()

def main(stdscr):
    app = SomeApp(stdscr)
    app.run()

if __name__ == '__main__':
curses.wrapper(main)
Wrapper Explained

- wrapper(main) executes main(stdscr) in a try/except

- stdscr is an initialized curses WindowObject

- The initialization includes:
  - cbreak: Buffering is turned off (But Ctrl-C still works)
  - no echo: Typed characters are not displayed on screen
  - colors: If the terminal supports colors they are initialized

- before exiting the settings are reset
Writting Strings

- A WindowObject is a uniform grid of characters
- Given a WindowObject $w$
  
  $w.addstr(row, column, some_string)$ will write some_string to the window starting at $(row, column)$
- Overwriting a section of a window will only replace those characters
  - Use $w.clear()$ to clear the entire window
- For the effects of a write or clear to take effect
  - $w.refresh()$ repaints the window
  - $w.noutrefresh()$ marks the window for update
  - $w.doupdate()$ actually repaints the screen
  - $w.refresh()$ marks the current window and repaints all marked windows
A window is basically a name for a rectangle of the screen

curses.newwin(height, width, r, c)
creates a window starting at row=r and column=c

Windows allow parts of the screen to be refreshed separately

Windows give a new coordinate system with (0, 0) in the top-left

A panel is a window with depth

You can overlap panels without overwriting other panels data
Setting Attributes

- When writing a string you can specify Attributes
- Background/Foreground color pairs:
  - red = curses.COLOR_RED
  - black = curses.COLOR_BLACK
  - curses.init_pair(1, red, black)
  - w.addstr(0, 0, 'some_text', curses.color_pair(1))
- Bolding and Highlighting:
  - w.addstr(0, 0, 'other_text', curses.A_BOLD)
  - w.addstr(0, 0, 'other_text', curses.A_BLINK)
- Attributes are not guaranteed to mix well but multiple can be specified
- Attributes can be applied to entire windows
  - w.bkgd(0, curses.A_STANDOUT)
key = w.getch() waits for a key to be pressed
The return value is an integer representing the character
Compare against constants to detect special keys
  if key == curses.KEY_RIGHT:
  w.getkey() will return a string instead of an integer
Control Logic

- `curses` is very low-level
- Minimal abstraction (Rectangles of characters)
- No notion of event handlers
  - All key-presses and mouse clicks must be explicitly directed
- You are in charge of all state
- a small fast music player

<table>
<thead>
<tr>
<th>Album / Artist</th>
<th>Track 1</th>
<th>Track 2</th>
<th>Track 3</th>
<th>Track 4</th>
<th>Track 5</th>
<th>Track 6</th>
<th>Track 7</th>
<th>Track 8</th>
<th>Track 9</th>
<th>Track 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmus</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Eric Kutschera (University of Pennsylvania)  
CIS 192  
March 20, 2015  
21 / 23
Outline

1. Graphical User Interface
   - Tkinter
   - Other Graphics Modules

2. Text User Interface
   - curses
   - Other Text Interface Modules
Urwid is a text widget library
Has more abstraction (Widgets instead of blocks of text)
If your UI is that complex just use a GUI