Additional UI Techniques

olsen software

Contents

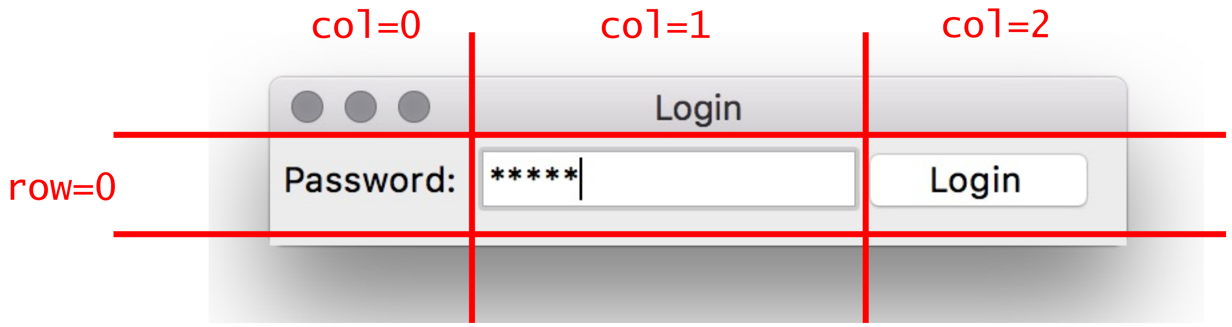
Geometry Managers

- Creating a widget doesn't automatically make it appear
  - It has to be positioned by a geometry manager.
- Can be a complex problem
  - Area wanted by controls is larger than the frame?
  - If there is lots of whitespace left, what to do?
  - If frame resized, how to move controls?
- Tk comes with several
  - 'pack' to arrange automatically
  - 'grid' to arrange on a grid
  - 'place' to put at a position

Grid Basics

- Grid is the most widely used Geometry Manager
  - Rows and columns numbered from 0
  - Sticky option controls positioning

```
pwd_entry = ttk.Entry(mainframe, width=15, textvariable=pwd, show='*')
pwd_entry.grid(column=1, row=0, sticky=(W, E))
ttk.Label(mainframe, text="Password: ").grid(column=0, row=0, sticky=W)

login_btn = ttk.Button(mainframe, text="Login", command=check_login)
login_btn.grid(column=2, row=0, sticky=W)
```



Other Grid Options

- Widgets can occupy more than cell
  - rowspan and columnspan options

```
namelbl.grid(column=3, row=0, columnspan=2)
```

- You can set resize behaviour per row/column
  - rowconfigure and columnconfigure

```
content = ttk.Frame(root, padding=(3,3,12,12))

content.grid(column=0, row=0, sticky=(N, S, E, W))

content.columnconfigure(0, weight=3)
content.columnconfigure(1, weight=3)
content.columnconfigure(2, weight=1)
content.rowconfigure(1, weight=1)
```

Events

- Many widgets can specify a command callback

```
ttk.Button(mainframe, text="Calculate", command=calculate)
```

- You can also bind to events on widgets
  - Key presses
  - Mouse button clicks and movement
- Use the 'bind' command
  - Takes an event name and a lambda
  - There are many events
  - You can have more than one binding for an event
  - Widgets can fire 'virtual events'

Events

- The lambda takes a single parameter
  - This may contain info about the event

```
l.bind('<B3-Motion>', lambda e: l.configure(
          text='right button drag to %d,%d' % (e.x, e.y)))
```

- It is common to make the lambda call a function
  - So the code doesn't get unreadable

```
lbl = ttk.Label(root, text="Starting...")
lbl.grid()

lbl.bind('<Enter>', lambda e: l.configure(text='Moved mouse inside'))
lbl.bind('<Leave>', lambda e: l.configure(text='Moved mouse outside'))
lbl.bind('<1>', lambda e: l.configure(text='Clicked left mouse button'))
```

Menus

- Menus are complex
  - Platforms handle them very differently
  - E.g. OS X has one menubar, Windows is per-window
  - We also have popup and system menus

- Before you start
  - On Windows and Linux
  - Prevents dashed line as first item...

```
root.option_add('*tearOff', FALSE)
```

Menus

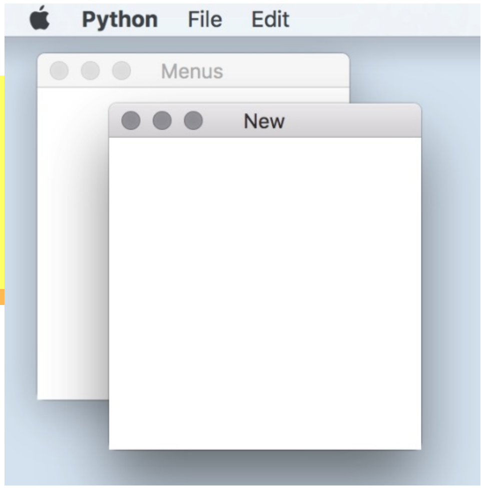
- Menus are represented by Menu widgets

- Use the 'menu' option to attach to a window

```
win = Toplevel(root)             # Create a toplevel window
win.title('New')                 # Set the title

menubar = Menu(win)              # Create a Menu widget
win['menu'] = menubar            # Attach the menu to the window
```

- Add top-level menus

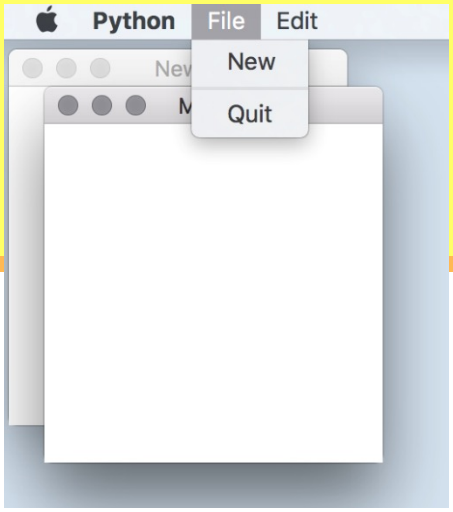
```
menu_file = Menu(menubar)    # Create menus
menu_edit = Menu(menubar)

menubar.add_cascade(menu=menu_file, label='File')  # Add items
menubar.add_cascade(menu=menu_edit, label='Edit')
```



Menu Commands

- Use 'add_command' to add menu items
  - And provide a command

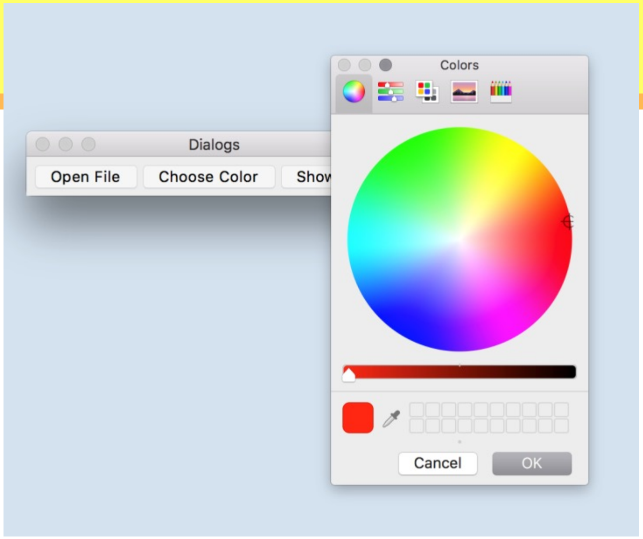
```
import os, sys

def newFile():
    print('Selected new file')

def exitProgram():
    sys.exit(0)

menu_file = Menu(menubar)    # Create menus
menu_edit = Menu(menubar)

menubar.add_cascade(menu=menu_file, label='File')  # Add items
menubar.add_cascade(menu=menu_edit, label='Edit')

menu_file.add_command(label='New', command=newFile)
menu_file.add_separator()
menu_file.add_command(label='Quit', command=exitProgram)
```



Standard Dialogs

- Tk supports almost all the standard system dialogs
  - File open and save
  - Directory picker
  - Colour picker
  - Alerts and messageboxes

```
from tkinter import filedialog

filename = filedialog.askopenfilename()
```



Any Questions?