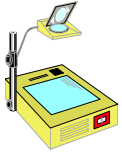




## Python: Tkinter menus



Learning Python, Lutz & Ascher

1

## Advanced widgets

- Menu, Menubutton and OptionMenu
- Scrollbar
- Listbox
- Canvas
- Grid

2

```
from Tkinter import *
from tkMessageBox import *

def notDone():
    showerror('Not Implemented', 'Not yet available')

def makeMenu(win):
    top = Menu(win)
    win.config(menu=top)

    file = Menu(top)
    file.add_command(label='New...', command=notDone, underline=0)
    file.add_command(label='Open...', command=notDone, underline=0)
    file.add_command(label='Quit...', command=win.quit, underline=0)
    top.add_cascade(label='File...', menu=file, underline=0)

    edit = Menu(top, tearoff=0)
    edit.add_command(label='Cut', command=notDone, underline=0)
    edit.add_command(label='paste', command=notDone, underline=0)
    edit.add_separator()
    top.add_cascade(label='Edit', menu=edit, underline=0)

    submenu = Menu(edit, tearoff=0)
    submenu.add_command(label='done', command=win.quit, underline=0)
    submenu.add_command(label='frodo', command=notDone, underline=0)
    edit.add_cascade(label='Stuff', menu=submenu, underline=0)
```

Make a top menu

Make a file menu

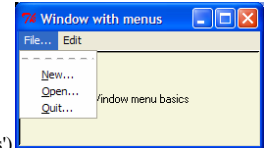
Make an edit menu

Make a submenu hanging off of the edit menu

3

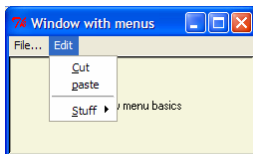
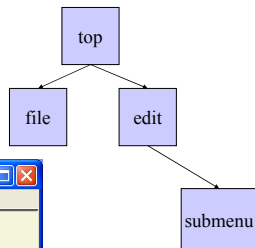
## The main routine

```
if __name__ == '__main__':
    root = Tk()
    root.title("Window with menus")
    makeMenu(root)
    msg = Label(root, text='Window menu basics')
    msg.pack(expand=YES, fill=BOTH)
    msg.config(relief=SUNKEN, width=40, height=7, bg='beige')
    root.mainloop()
```



4

## Cascade of Menu trees



5

## add\_cascade

- Associate a horizontal menu bar with the top-level window object
- Add other pull-down menu objects as cascades of the top-level Menu
- Menus are cross-linked with the next higher level by using parent widget arguments and the add\_cascade method

6

## algorithm

- Create a topmost Menu
- For each pull-down, make a new Menu as the child of the topmost Menu and add the child using **add\_cascade**
- Add menu selections to each pull-down Menu using **add\_command** to register callback handlers

7

## Tear offs

- Dashed lines that appear by default at the top of Tkinter menus
- They create a new window when clicked

8

## Keyboard shortcuts

- Underline option makes a unique letter in a menu entry as a keyboard shortcut
- For example, Quit option can be selected with the mouse or alt-f-q

9

## Windows with both menus and toolbars

Menus & buttons

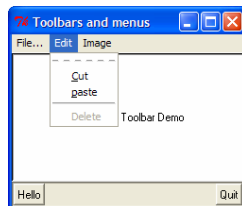
10

```
from Tkinter import *
from tkMessageBox import *
```

```
class NewMenuDemo(Frame):
    def __init__(self, parent=None):
        Frame.__init__(self, parent)
        self.pack(expand=YES, fill=BOTH)
        self.createWidgets()
        self.master.title("Toolbars and menus")
        self.master.iconname("Tkinter")
```

```
def createWidgets(self):
    self.makeMenuBar()
    self.makeToolBar()
    L = Label(self, text='Menu and Toolbar Demo')
    L.config(relief=SUNKEN, width=40, height=10, bg='white')
    L.pack(expand=YES, fill=BOTH)
```

```
def makeToolBar(self):
    toolbar = Frame(self, cursor='hand2', relief=SUNKEN, bd=2)
    toolbar.pack(side=BOTTOM, fill=X)
    Button(toolbar, text='Quit', command=self.quit).pack(side=RIGHT)
    Button(toolbar, text='Hello', command=self.greeting).pack(side=LEFT)
```



11

```
def makeMenuBar(self):
    self.menubar = Menu(self.master)
    self.master.config(menu=self.menubar)
    self.fileMenu()
    self.editMenu()
    self.imageMenu()
```

```
def fileMenu(self):
    pulldown = Menu(self.menubar)
    pulldown.add_command(label='New...', command=self.notDone)
    pulldown.add_command(label='Open...', command=self.notDone)
    pulldown.add_command(label='Quit...', command=self.quit)
    self.menubar.add_cascade(label='File...', menu=pulldown, underline=0)
```

```
def editMenu(self):
    pulldown = Menu(self.menubar)
    pulldown.add_command(label='Cut', command=self.notDone, underline=0)
    pulldown.add_command(label='paste', command=self.notDone, underline=0)
    pulldown.add_separator()
    pulldown.add_command(label='Delete', command=self.greeting)
    pulldown.entryconfig(4, state=DISABLED)
    self.menubar.add_cascade(label='Edit', menu=pulldown, underline=0)
```

12

```

def imageMenu(self):
    photoFiles = ('frodo.gif', 'Image011.gif', 'Image007.gif')
    pulldown = Menu(self.menubar)
    self.photoObjs = []
    for file in photoFiles:
        img = PhotoImage(file='./gifs/'+file)
        pulldown.add_command(image=img, command=self.notDone)
        self.photoObjs.append(img) #keep a reference
    self.menubar.add_cascade(label='Image', underline=0, menu=pulldown)

def notDone(self):
    showerror('Not Implemented', 'Not yet available')

def greeting(self):
    showinfo('greeting', 'Greetings...')

def quit(self):
    if askyesno('Verify quit', 'Are you sure you want to quit?'):
        Frame.quit(self)

if __name__ == '__main__':
    NewMenuDemo().mainloop()

```

13

## Listboxes and scrollbars

14

```

class ScrolledList(Frame):
    def __init__(self, options, parent=None):
        Frame.__init__(self, parent)
        self.pack(expand=YES, fill=BOTH)
        self.makeWidgets(options)
    def handleList(self, event):
        index = self.listbox.curselection()
        label = self.listbox.get(index)
        self.runCommand(label)
    def makeWidgets(self, options):
        sbar = Scrollbar(self)
        list = Listbox(self, relief=SUNKEN)
        sbar.config(command=list.yview)
        list.config(yscrollcommand=sbar.set)
        sbar.pack(side=RIGHT, fill=Y)
        list.pack(side=LEFT, expand=YES, fill=BOTH)
        pos=0
        for label in options:
            list.insert(pos, label)
            pos += 1
        #list.config(selectmode=SINGLE, setgrid=1)
        list.bind('<Double-1>', self.handleList)
        self.listbox = list
    def runCommand(self, selection):
        print 'You selected:', selection

```



You selected: Lord of the Rings 5

```

if __name__ == '__main__':
    options = map(lambda x: 'Lord of the Rings '+
                  str(x), range(1,21))
    ScrolledList(options).mainloop()

```

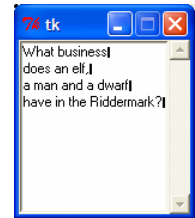
15

```

if __name__ == '__main__':
    file = open('data', 'r')
    lines = file.readlines()
    ScrolledList(lines).mainloop()

```

What business  
does an elf,  
a man and a dwarf  
have in the Riddermark?



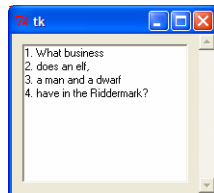
16

## Get rid of newline!

```

if __name__ == '__main__':
    file = open('data', 'r')
    linelist = []
    x = 1
    while 1:
        line = file.readline()
        if not line: break
        linelist = linelist+ [str(x)+' '+line[:-1]]
        x += 1
    ScrolledList(linelist).mainloop()

```



17

## Grids

- Most commonly used alternative to pack
- Tkinter layout managers arrange child widgets within a container (parents are Frames or top-level window)
- Grid arranges widgets in rows & columns
- Grid & pack are mutually exclusive for widgets that have the same parent
- Grid is handy for laying out form-like data

18

```

from Tkinter import *
colors = ['red','green','yellow','orange','blue','navy']

def gridbox(parent):
    r = 0
    for c in colors:
        l = Label(parent, text=c, relief=RIDGE, width=25)
        e = Entry(parent, bg=c, relief=SUNKEN, width=50)
        l.grid(row=r, column=0)
        e.grid(row=r, column=1)
        r += 1

def packbox(parent):
    for c in colors:
        f = Frame(parent)
        l = Label(f, text=c, relief=RIDGE, width=25)
        e = Entry(f, bg=c, relief=SUNKEN, width=50)
        f.pack(side=TOP)
        l.pack(side=LEFT)
        e.pack(side=RIGHT)

```

19

## main

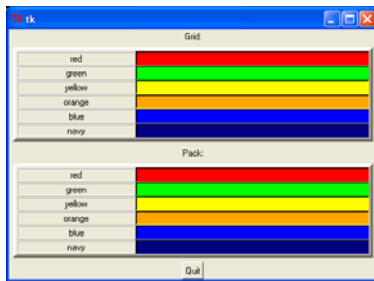
```

root = Tk()
Label(root, text='Grid:').pack()
frm = Frame(root, bd=5, relief=RAISED); frm.pack(padx=5, pady=5)
gridbox(frm)
Label(root, text='Pack:').pack()
frm = Frame(root, bd=5, relief=RAISED); frm.pack(padx=5, pady=5)
packbox(frm)
Button(root, text='Quit', command=root.quit).pack()
mainloop()

```

20

## Compare grid and pack



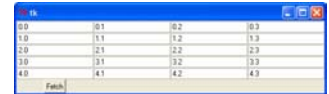
21

```
from Tkinter import *
```

```

rows=[]
for i in range(5):
    cols=[]
    for j in range(4):
        e = Entry(relief=RIDGE)
        e.grid(row=i, column=j, sticky=NSEW)
        e.insert(END,"%d.%d" % (i,j))
        cols.append(e)
    rows.append(cols)

```



```

def onPress():
    for row in rows:
        for col in row:
            print col.get()
    print

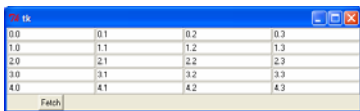
```

```

Button(text='Fetch', command=onPress).grid()
mainloop()

```

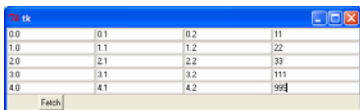
22



```

0.0 0.1 0.2 0.3
1.0 1.1 1.2 1.3
2.0 2.1 2.2 2.3
3.0 3.1 3.2 3.3
4.0 4.1 4.2 4.3

```



```

0.0 0.1 0.2 1.1
1.0 1.1 1.2 2.2
2.0 2.1 2.2 3.3
3.0 3.1 3.2 1.1
4.0 4.1 4.2 9.99

```

23