ENGR 102 PROGRAMMING PRACTICE

WEEK 5

(GUI) Programming
Grids, Events

Grid Geometry Manager

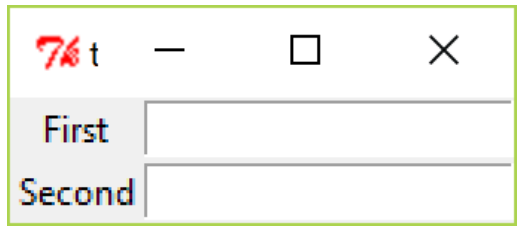
- Puts the widgets in a 2-dimensional table.
- The parent container is split into a number of rows and columns
  - each "cell" in the resulting table can hold a widget.
- First, create the widgets, and
- Then, use the **grid** method to tell the manager in which row and column to place them.
- No need to specify the size of the grid beforehand; the manager automatically figures that out.

grid() – important attributes

- row:
  - value <int>
  - row index of the grid the widget should appear in
- column:
  - value: <int>
  - column index of the grid the widget should appear in

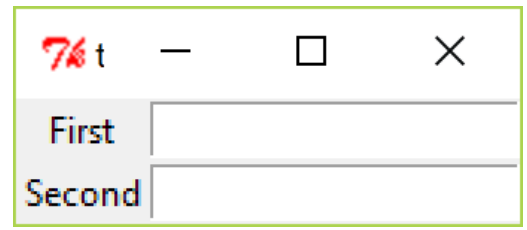
Example

- Draw the following app



Grid Geometry Manager

```
from Tkinter import *

root = Tk()

l1 = Label(root, text="First")
l2 = Label(root, text="Second")

l1.grid(row=0)
l2.grid(row=1)

e1 = Entry(root)
e2 = Entry(root)

e1.grid(row=0, column=1)
e2.grid(row=1, column=1)

root.mainloop()
```



grid() – important attributes

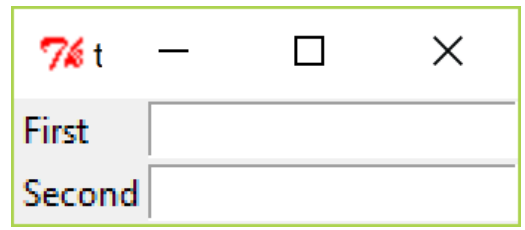
- sticky:
  - value: N,S,E,W,NE,… or any set of these
  - how should the widget aligned and resized inside a grid
    - e.g., sticky="N"

    - e.g., sticky=("N","S")

    - e.g., sticky=("N","S","E","W")

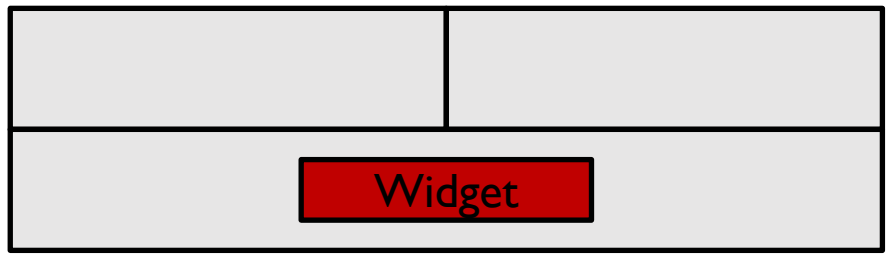
Grid Geometry Manager

```
from Tkinter import *

root = Tk()

l1 = Label(root, text="First")
l2 = Label(root, text="Second")

l1.grid(row=0, sticky = W)
l2.grid(row=1)

e1 = Entry(root)
e2 = Entry(root)

e1.grid(row=0, column=1)
e2.grid(row=1, column=1)

root.mainloop()
```

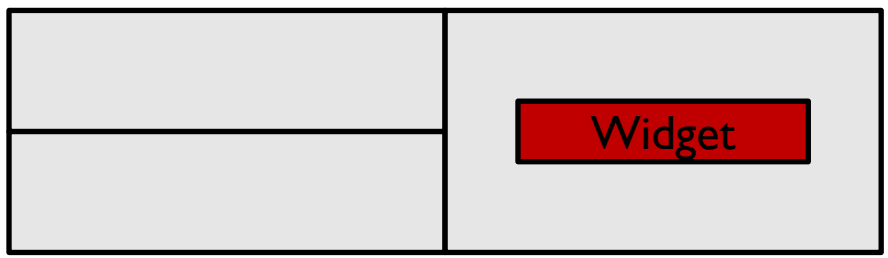


grid() – important attributes

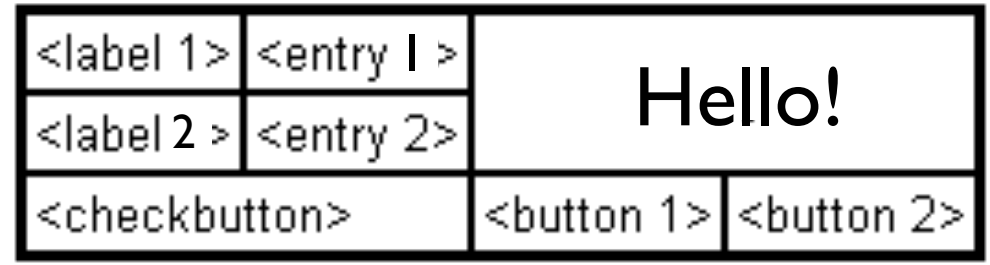
- columnspan:
  - value: <int>
  - specifies if the widget should occupy more than one column (e.g., next columns)



- rowspan:...



Grid Geometry Manager



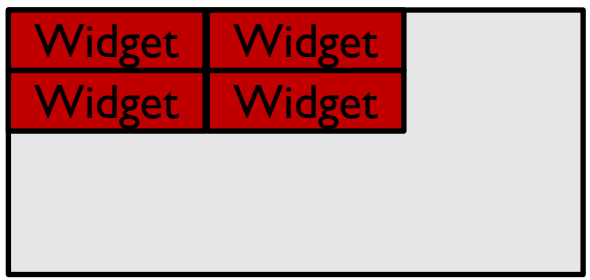

```
master = Tk()
label1 = Label(master, text="First")
label2 = Label(master, text="Second")
entry1 = Entry(master)
entry2 = Entry(master)
checkbutton = Checkbutton(master, text="Show title")
label3 = Label(master, text='Hello!')
ok_button = Button(master, text="OK")
cancel_button = Button(master, text="Cancel")

label1.grid(sticky=E) # by default column 0 of the next empty row
label2.grid(sticky=E) # can you guess row and column number of this guy?
entry1.grid(row=0, column=1)
entry2.grid(row=1, column=1)
checkbutton.grid(columnspan=2, sticky=W)
label3.grid(row=0, column=2, columnspan=2, rowspan=2,
            sticky=W+E+N+S, padx=5, pady=5)

ok_button.grid(row=2, column=2)
cancel_button.grid(row=2, column=3)
```

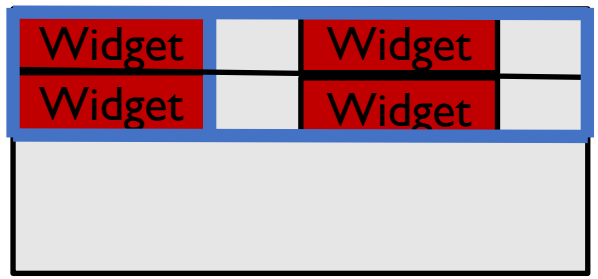
Grid Geometry Manager

```
from Tkinter import *

root = Tk()

colours = ['red','green','orange','white','yellow','blue']

r = 0
for c in colours:
    Label(root, text=c).grid(row=r,column=0, sticky=E)
    Entry(root, bg=c).grid(row=r,column=1, sticky=EW)
    r = r + 1

root.mainloop()
```

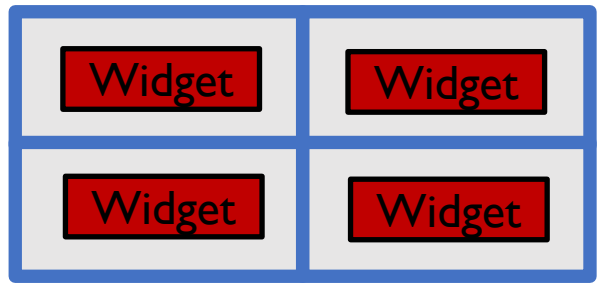
Grid.columnconfigure(container, col_index, weight)

- Used to force columns to expand horizontally.
- Weight can be used to adjust column widths proportional to container width.
- rowconfigure is similar





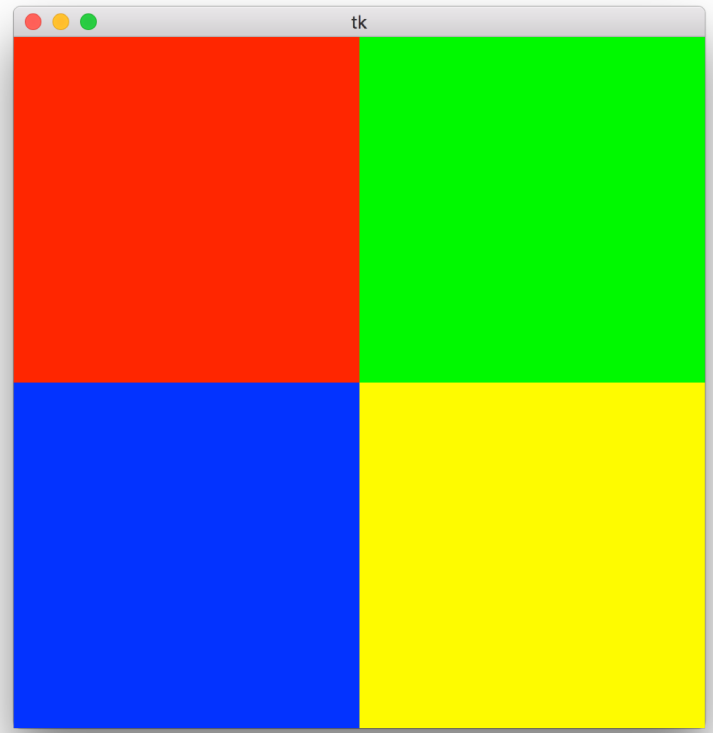
columnconfigure(..., 1, 1)



columnconfigure(..., 0, 1)
columnconfigure(..., 1, 1)
rowconfigure (..., 0, 1)
rowconfigure (..., 1, 1)

Example

- Write an app with pack geo manager with a UI that looks like
  - Hint: use frames
  - See FourColorsGrid.py



Grid Geometry Manager

Expand with Resize

```
from Tkinter import *

root = Tk()

colours = ['red','green','orange','white','yellow','blue']

r = 0
Grid.columnconfigure(root, 0, weight=1)
Grid.columnconfigure(root, 1, weight=1)
for c in colours:
    Label(root, text=c).grid(row=r,column=0, sticky=E)
    Entry(root, bg=c).grid(row=r,column=1, sticky=EW)
    r = r + 1
    Grid.rowconfigure(root, r, weight=1)

root.mainloop()
```

Events

- An ***event*** is some occurrence that your application needs to know about.
- An ***event handler*** is a function in your application that gets called when an event occurs.
- We call it ***binding*** when your application sets up an event handler that gets called when an event happens to a widget.

Events

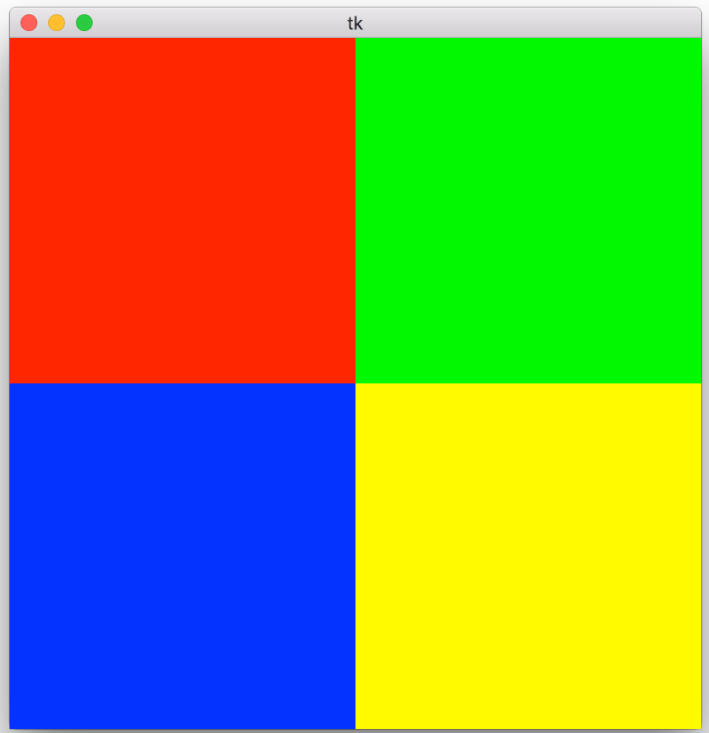
- Event Sources:
  - Mouse Operations by user
  - Key Presses by user
  - Redraw Events by window manager

- Capturing and handling events
- If an event matching the *event* description occurs on the widget, *handler* is called with an event object

```
widget.bind(event, handler)
```

Example

- Write an app that shows four labels on a 2x2 grid.
- When the user double-clicks on any of the labels, the label changes color.
  - red changes to green
  - green changes to blue
  - blue changes to yellow
  - yellow changes to red
- See ChangeFourColors.py



Example

- Write an app that initially draws a balloon on a canvas.
  - When the user clicks on the canvas, the balloon moves to the cursor location.
  - When the user press arrow keys, the balloon moves accordingly.

- See MovingBalloon.py

Events – Capturing Clicks

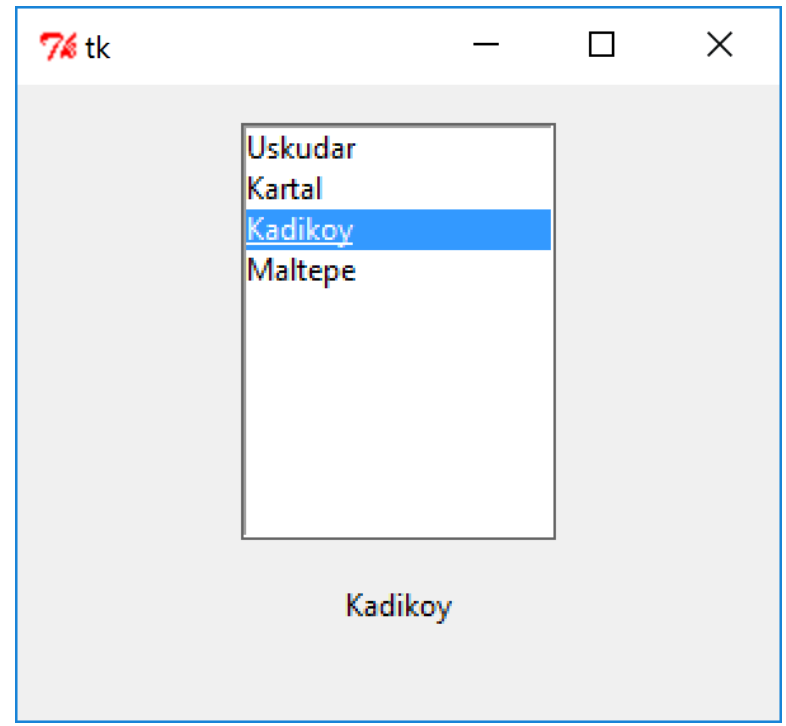
```
from Tkinter import *

class MyApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):
        self.pack(fill=BOTH, expand=True)
        label = Label(self, bg="yellow")
        label.pack(fill=BOTH, expand=True)

        label.bind("<Button-1>", self.onClick)

    def onClick(self, event):
        print "clicked at", event.x, event.y

def main():
    root = Tk()
    root.geometry('300x300+200+200')
    app = MyApp(root)
    root.mainloop()

main()
```

Events – Capturing Clicks

```
from Tkinter import *
class MyApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.colors = ['red', 'yellow', 'green']
        self.color_index = 0
        self.initUI()

    def initUI(self):
        self.pack(fill=BOTH, expand=True)
        label = Label(self, text = 'Click me to change my color!')
        label.pack(fill=BOTH, expand=True)

        label.bind("<Button-3>", self.onClick)

    def onClick(self, event):
        widget = event.widget
        self.color_index = (self.color_index + 1) % len(self.colors)
        widget.config(bg=self.colors[self.color_index])
        print "clicked at", event.x, event.y

def main():
    root = Tk()
    root.geometry('300x300+200+200')
    app = MyApp(root)
    root.mainloop()

main()
```

Events – Capturing ListBox Selections

```
from Tkinter import *
class MyApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()

    def initUI(self):
        self.pack(fill=BOTH, expand=1)
        acts = ['Uskudar', 'Kartal', 'Kadikoy', 'Maltepe']
        lb = Listbox(self)
        for i in acts:
            lb.insert(END, i)

        lb.bind("<<ListboxSelect>>", self.onSelect)
        lb.pack(pady=15)
        self.var = StringVar()
        self.label = Label(self, text=0, textvariable=self.var)
        self.label.pack()

    def onSelect(self, val):
        sender = val.widget
        idx = sender.curselection()
        value = sender.get(idx)
        self.var.set(value)

def main():
    root = Tk()
    ex = MyApp(root)
    root.geometry("300x250+300+300")
    root.mainloop()

main()
```



List of Events

- <Button-1> <Button-2> <Button-3> <Button-4> <Button-5>
- <Motion>
- <ButtonRelease-1> <ButtonRelease-2> <ButtonRelease-3>
- <Double-Button-1> <Double-Button-2> <Double-Button-3>
- <Enter>
- <Leave>
- <FocusIn>
- <FocusOut>
- <Return>
- <Key>
- a
- <Shift-Up>
- <Configure>

Ref: http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm