HOW TO:

Tkinter

Jack And Anan

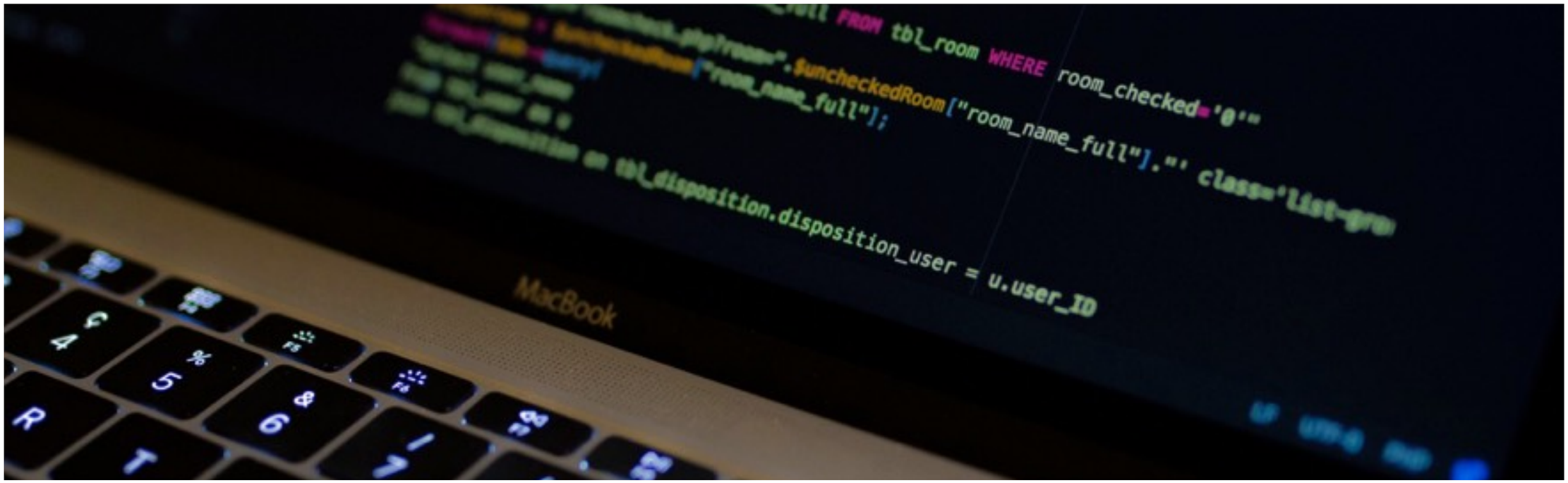
CHAPTER 1

What is Tkinter?

Before we get into the basics of actually using the library, you may have some questions, like what is Tkinter? Or for what can it be used?

CHAPTER 1

What is Tkinter?



Tkinter is a library for Python that allows programers to build a graphical user interface (GUI). A GUI enables users to interact with a window on their computer screen running that program which displays images, backgrounds, shapes and many other visuals that respond to clicks, keyboard strokes, and other inputs, rather than just a text-based terminal. Most programs nowadays use a GUI rather than a terminal because it is more user-friendly to implement a variety of inputs, and it allows for a more eye-appealing, better organized program. For example, Microsoft Word uses a GUI, allowing users to navigate with their mouse and select certain locations in the text to directly edit. This is much more dynamic than a black-and-white terminal that only accepts text. There are many GUI packages available for use with python, but Tkinter is one of the easiest and most functional, plus it comes pre-installed with python.

CHAPTER 2

Getting Started

Making tkinter run on your computer (or laptop)

CHAPTER 2

Getting Started

Since Tkinter comes pre-installed with Python, there is no need to download any files from the internet. In order to use the Tkinter commands within your program though, you have to first import the library. There are several ways to import the library, which depending on you how you do it can make it easier to call on the functions and classes within the library.

A better way to import the library is as follows:

```
import tkinter
```

```
from tkinter import*
```

This is the most basic way to import. Whenever you call on a function or class from the library however, you will have to put tkinter. before it. This can be annoying

Importing as such won't require any additional keywords, other than the function or class from the library. We recommend importing this way

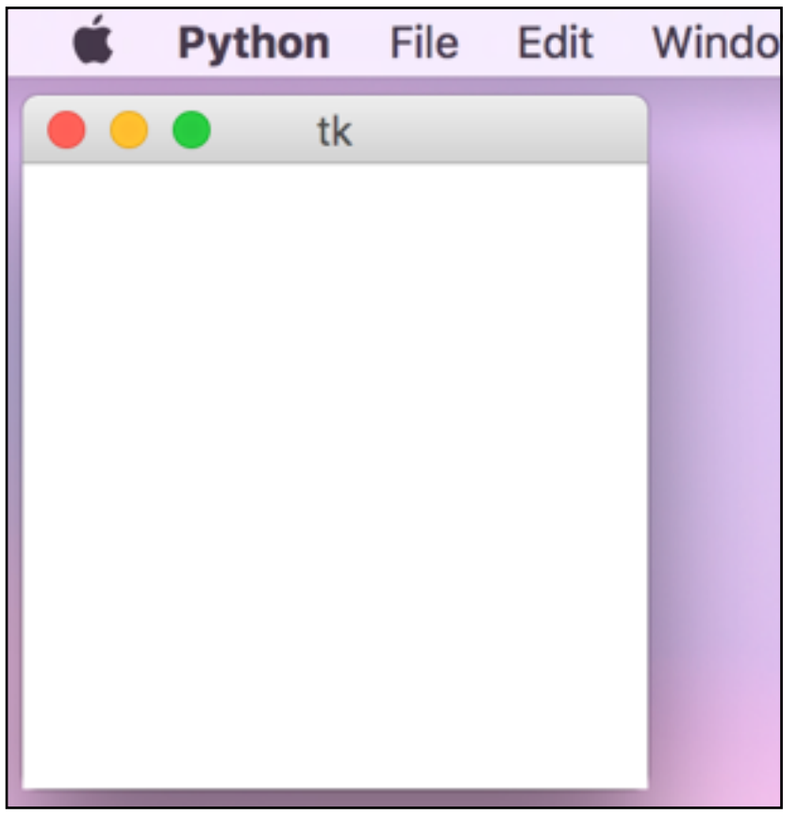
CHAPTER 3

Making a Window

The window is what will pop up when the program runs, in which the user will interact. This is the first step you must take in building your GUI, because it will house other elements of your program and defines the size limits you have to work with.

A Basic Window

```
1   from tkinter import*
2
3   myframe = Tk()
4
5   myframe.mainloop()
```

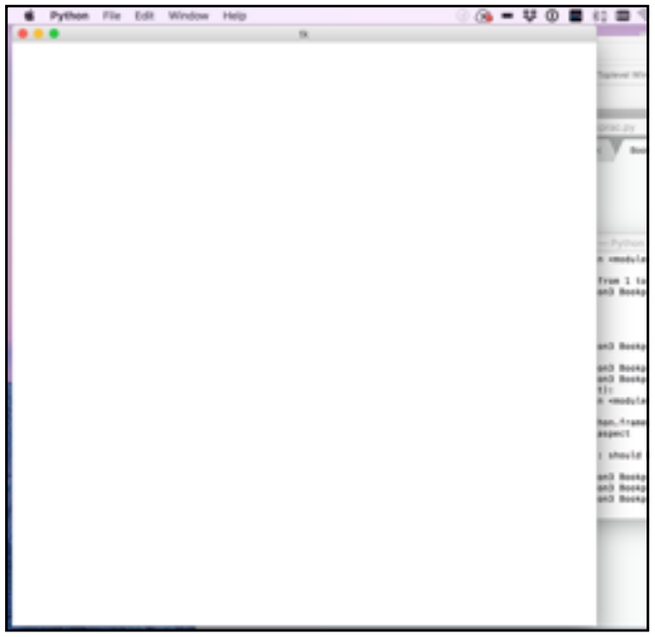


The code to the left will appear as the window on the right when run. myframe is the just the arbitrary name of the window. It is defined as being a window because of the Tk() class. In order to make this window appear on your desktop, you have to use the mainloop() function. Putting the specific window in front of this function will display just this window, while using mainloop() as its own, separate line of code, will display all windows, if you have multiple. In order to exit the program and go back to the shell, you have to "x-out" the window.

This is a good, simple way to establish a window, and its size can be adjusted by the user. However, the size of the window can be established from the beginning with some additional code.

Sizing the Window

```
1  from tkinter import*
2
3  myframe = Tk()
4
5  myframe.aspect(1,1,1,1)
6  myframe.geometry('800x800')
7
8  mainloop()
```



In order to adjust the sizing, we have to add a few other functions. In the code on the left, the aspect ratio (width to height ratio) and starting size are defined. There are many other options for the sizing, which are defined similarly, such as minimum size and maximum size. The aspect() function takes four arguments: a, b, c, and d. The function keeps the pixel ratio of width vs height of the window between a:b and c:d. 1,1,1,1 will keep the height and width equal. The geometry function defines the starting size of the window as (width x height) in pixels. You can see the result in the large window on the right.

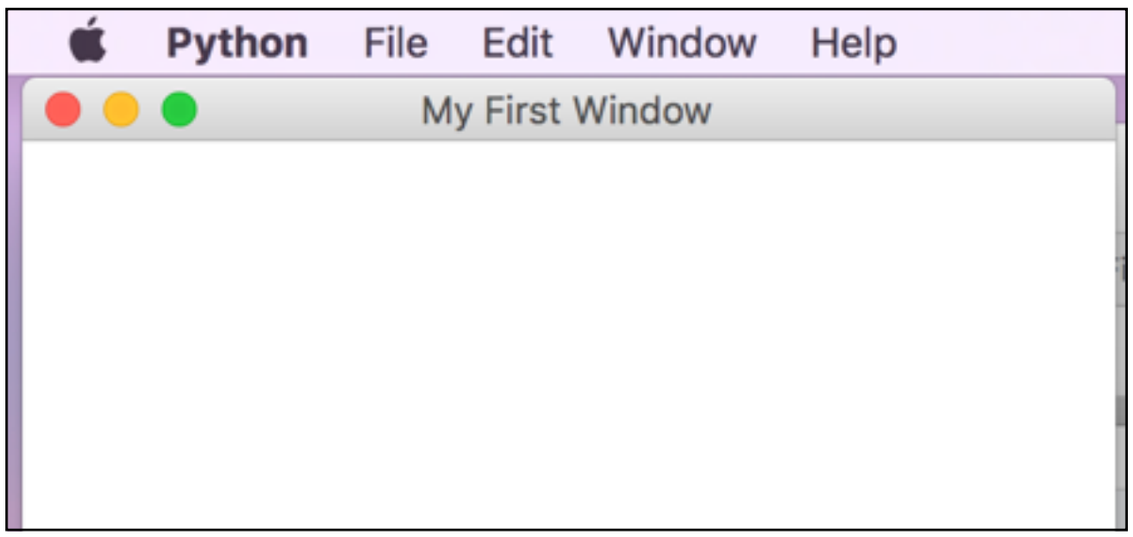
Other Window Options

In addition to sizing, many other features of the window can be customized using the same formatting. For example, the title() function can be used to give a title to the window.

Other customizations to the main window can be found here:

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

```
1   from tkinter import*
2
3   myframe = Tk()
4
5   myframe.aspect(1,2,1,2)
6   myframe.geometry('400x800')
7   myframe.title("My First Window")
8
9   mainloop()
10
```



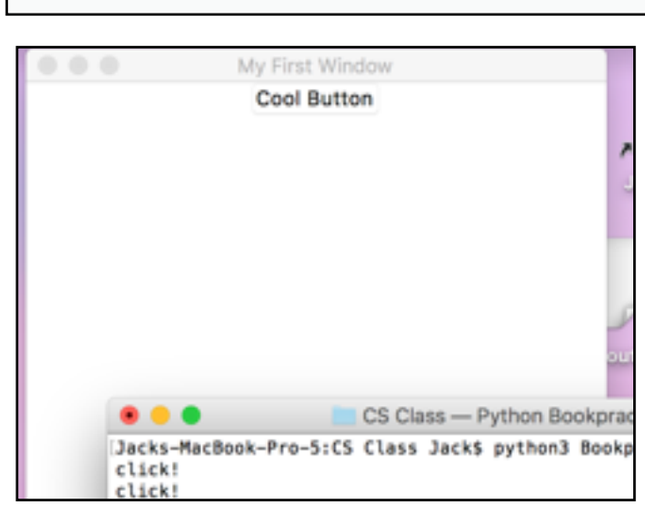
CHAPTER 4

Widgets

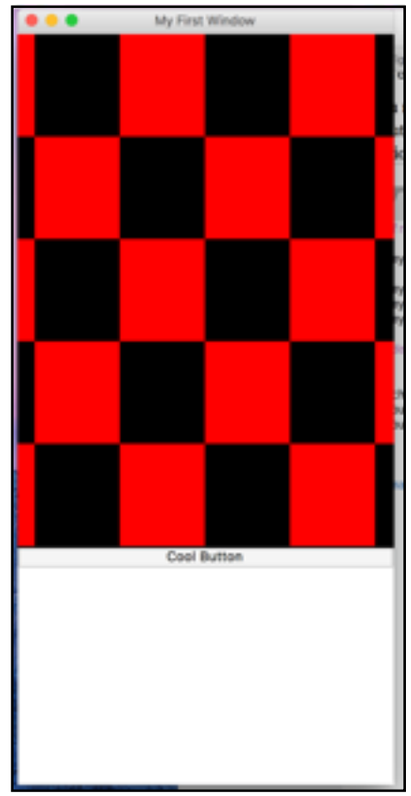
Widgets are what makes your GUI come alive. These include buttons, text boxes, shapes, and checkboxes the user can interact with while running the program. Learning how to implement these effectively will have a drastic impact on the user experience, and are straightforward once you learn the pattern in creating them

Buttons

```
from tkinter import*

myframe = Tk()

myframe.aspect(1,2,1,2)
myframe.geometry('400x800')
myframe.title("My First Window")

def click():
    print("click!")

button1 = Button(myframe, text="Cool Button", command = click)
button1.pack()

mainloop()
```

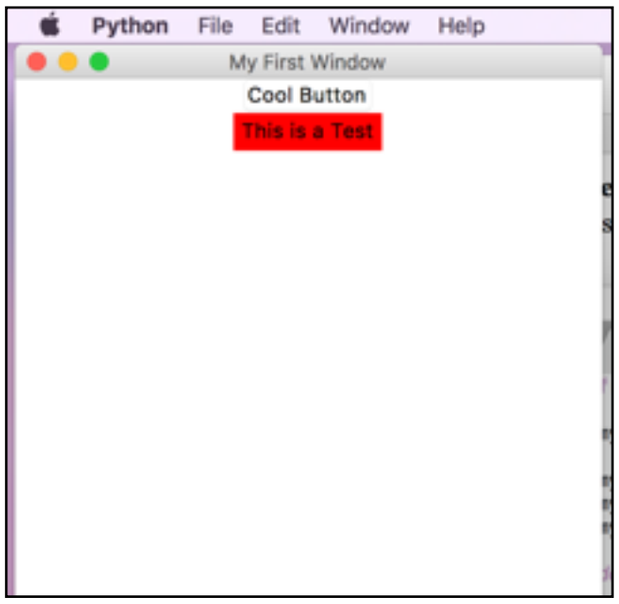



```
1  from tkinter import*
2
3  myframe = Tk()
4
5  myframe.aspect(1,2,1,2)
6  myframe.geometry('400x800')
7  myframe.title("My First Window")
8
9  def click():
10     print("click!")
11
12 checkerpic = PhotoImage(file ="checker2.png")
13 button1 = Button(myframe, compound= TOP,text="Cool Button", image= checkerpic, command = click)
14 button1.pack()
15
16
17
18 mainloop()
19
```



Buttons are one of the easiest widgets to use. When defining your buttons, as seen in the first example, you use the Button() class with a series of parameters. The first parameter is what window/frame to put the button (we will talk about frames later- they are basically a certain area within the window), and other parameters are named by their purpose. Here we have included text, which defines the text shown on the button, and command, which executes a certain function in your code upon click. Packing the button with the pack() function will organize the parameters of the button and display them in the frame. In the second example, other customization options of the buttons are shown - here attaching an image to a button - using the image parameter which holds the image and compound, which determines where the image is in relation to text. It is important to store the photo using PhotoImage() so it is in the correct format. A full list of button parameters can be found here: http://effbot.org/tkinterbook/button.htm

Labels




```
1  from tkinter import*
2
3  myframe = Tk()
4
5  myframe.aspect(1,1,1,1)
6  myframe.geometry('400x400')
7  myframe.title("My First Window")
8
9  def click():
10     print("click!")
11
12 button1 = Button(myframe,text="Cool Button", command = click)
13 button1.pack()
14
15
16 label1 = Label(myframe, text = "This is a Test", cursor = 'pirate', background = 'RED')
17 label1.pack()
18
19 mainloop()
20
```

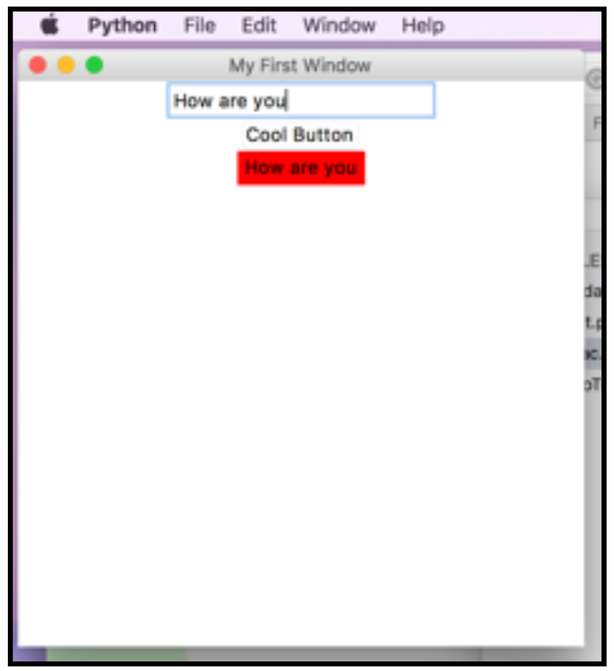
Labels are the simplest way to add text to a GUI, and are basically text-boxes that can be placed anywhere in the frame/window. Similar to buttons, they are defined as a series of parameters in a class, the first one being the window/frame to place them in. Once again, there are many optional parameters to customize several aspects, here including the background color and the cursor that appears when you hover over the label. A full list of parameters can be found here:

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

CHAPTER 4

Text Entries

```
1  from tkinter import*
2
3  myframe = Tk()
4
5  myframe.aspect(1,1,1,1)
6  myframe.geometry('400x400')
7  myframe.title("My First Window")
8
9
10 changeable_string = StringVar()
11 entry1 = Entry(myframe, textvariable = changeable_string)
12 entry1.pack()
13
14 def click():
15     label1 = Label(myframe, text = changeable_string.get(), cursor = 'pirate', background = 'RED')
16     label1.pack()
17
18 button1 = Button(myframe,text="Cool Button", command = click)
19 button1.pack()
20
21 mainloop()
22
```



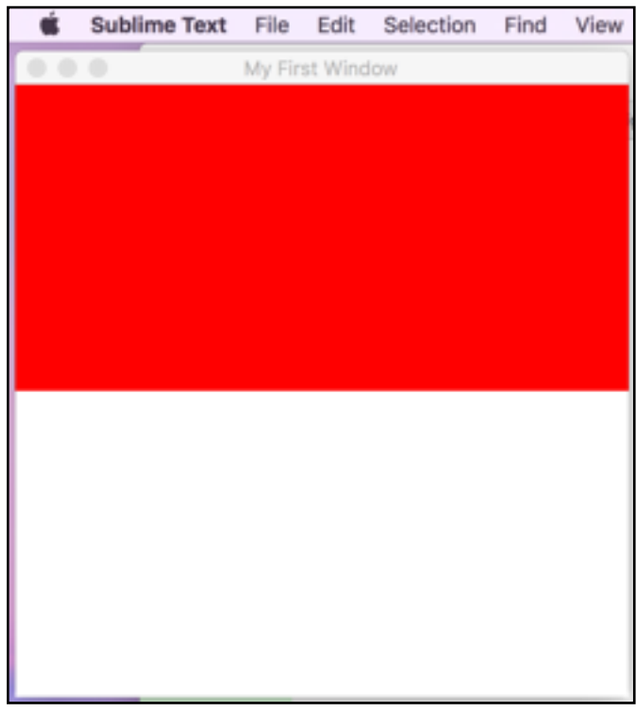
Text entries are in a sense opposite to labels, as they take in text input from the user. An important parameter for the entry class is textvariable. This decides where the string typed into the entry is stored. Because it is changing, you have to define whatever variable you store it in as a StringVar(), so it can change. The code above, upon the button being clicked, retrieves the text in the entry, using the changeable_string.get() function, and displays it as a label. Other entry parameters can be found here:

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

CHAPTER 4

Frames

```
1  from tkinter import*
2
3  master = Tk()
4
5  master.aspect(1,1,1,1)
6  master.geometry('400x400')
7  master.title("My First Window")
8
9  smallerframe = Frame(master, width = 400, height = 200, background = "RED")
10 smallerframe.pack()
11
12
13 mainloop()
```

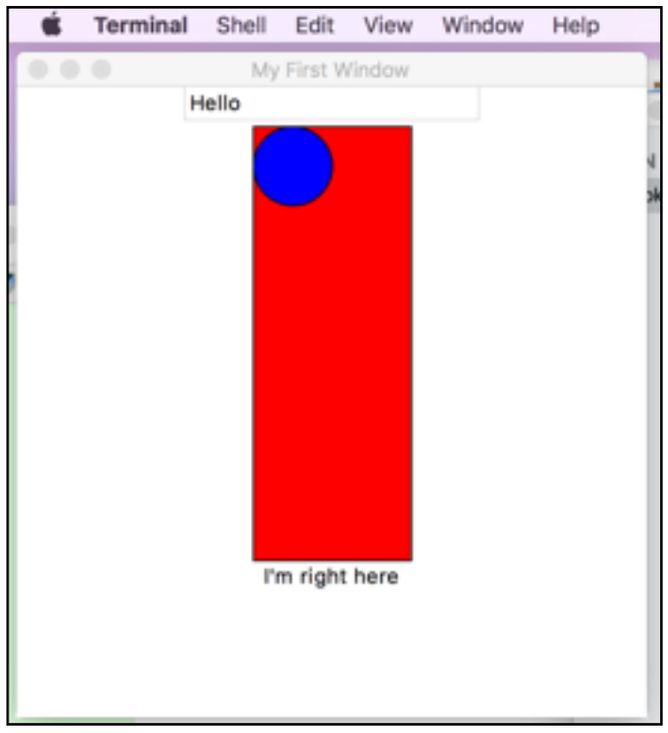


Frames essentially can divide up the master window into smaller sections, perhaps with some special aspects (like background color). These can be useful organizing a large program with a few things going on (like how to bar at the top of a Mac screen is separate from main area of screen) or just make placing widgets easier, as widgets can be placed within a certain frame rather than just the master window. Frame are defined like widgets rather than windows, using the Frame() class that takes parameters. Parameters shown here are the window/frame to place, width and height (in pixels), and background color. Other frame parameters can be found here:

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

Canvas

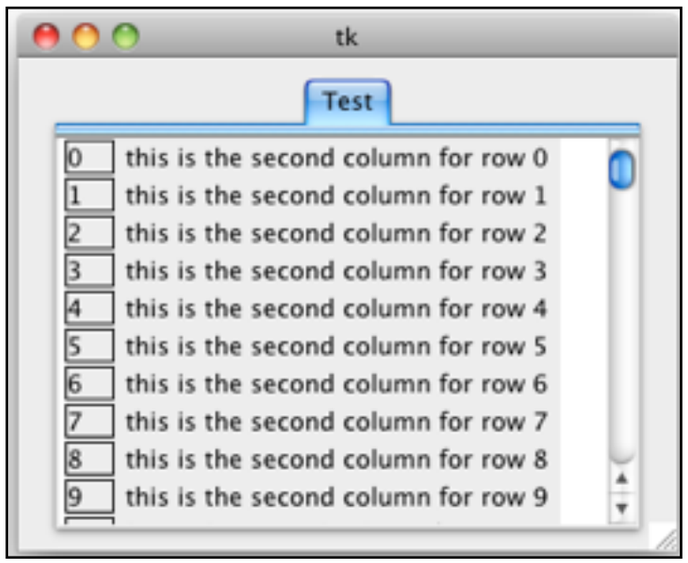
```
1  from tkinter import*
2
3  master = Tk()
4
5  master.aspect(1,1,1,1)
6  master.geometry('400x400')
7  master.title("My First Window")
8
9  draw_area = Canvas(master, width =400, height =400)
10 draw_area.pack()
11
12 draw_area.create_rectangle(150, 25,250,300, fill = 'RED')
13 draw_area.create_oval(150,25,200,75, fill = 'BLUE')
14 draw_area.create_text(200, 310, text="I'm right here")
15
16
17
18 changeable_string = StringVar()
19 entry1 = Entry(master,textvariable = changeable_string)
20
21 draw_area.create_window(200, 10, window=entry1)
22
23 mainloop()
24
```

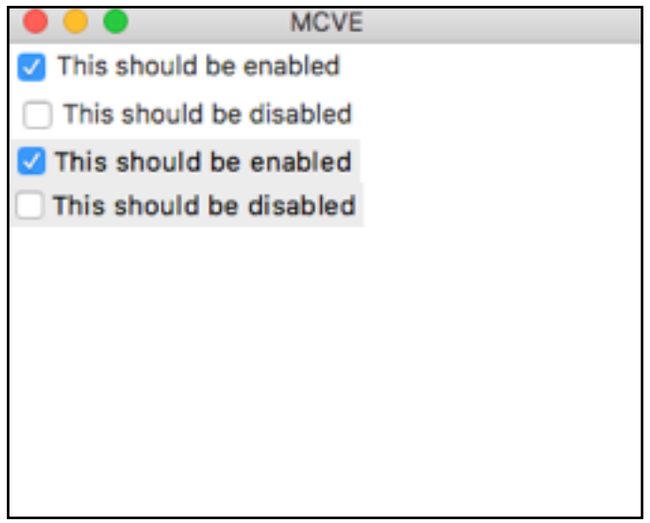


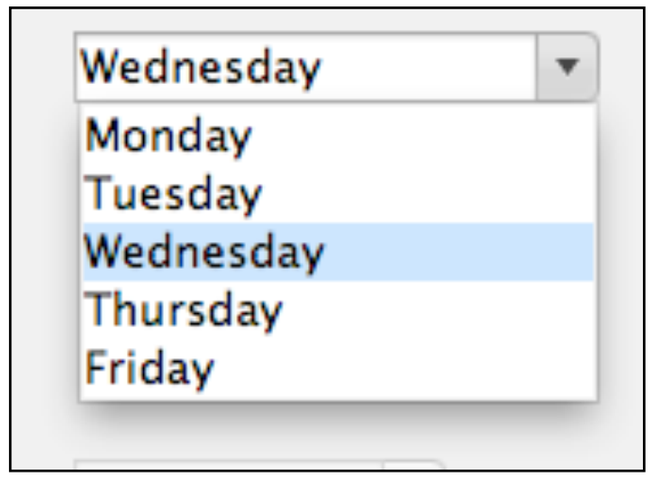
Canvas defines a frame-like area that has special options and capabilities, like drawing shapes or displaying graphs. First, you have to define the canvas area, using the Canvas() class and packing it, similar to other widgets. In this example we made it take up the entire master window. From here, we first drew the rectangle. The mandatory parameters of this function create_rectangle() are four numbers, which define the points the rectangle begins and ends. The first two numbers are the x and y coordinates of the lefthand corner of the rectangle, and the second two are those for the bottom righthand corner. We also choose to use the fill parameter for color. The circle is defined similarly, and the upper lefthand corner and lower righthand corner coordinates can be imagined as those for a box tight around the circle. You can add text to the canvas with the create_text() function, which places text centered on the (X,Y) coordinate provided. In addition, you can add widgets to the canvas not by packing, but by using the create_window() function and declaring the centering coordinate and widget name. Other capabilities of the canvas widget can be found here:

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

Other Widgets







There are many other widgets to work with in Tkinter, such as scrollbars, checkboxes, spinboxes, and radiobuttons, some of which are pictured above. Other widgets work similarly to the ones discussed, having a defining class along with several parameters. As long as you can remember this formatting, you will be able to use any widget the library has to offer. This chapter has provided just a taste of some widgets Tkinter uses and has hopefully inspired interest in you to research more. Information on other widgets can be found here:

http://effbot.org/tkinterbook/

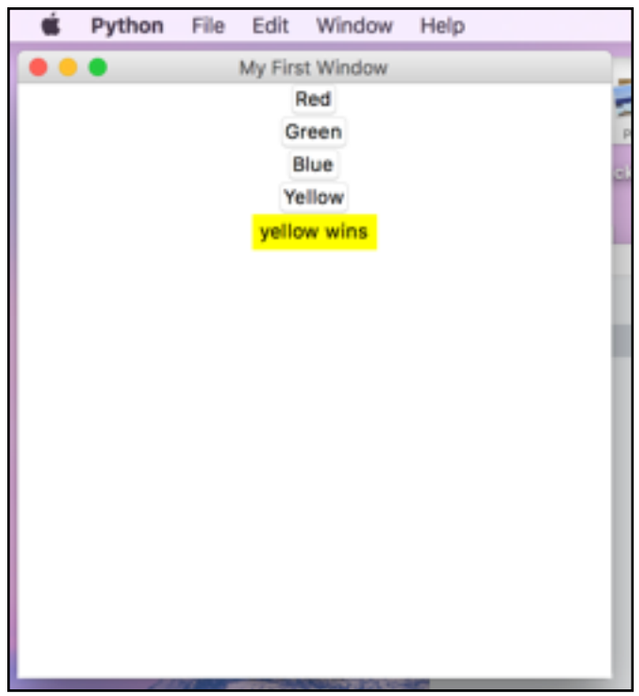
CHAPTER 5

Organizing Widgets

While simply packing widgets can do the job, there are much better ways to organize your GUI. We will cover two of these ways - directional packing and grid layout

Working Example

```
from tkinter import*

master = Tk()

master.aspect(1,1,1,1)
master.geometry('400x400')
master.title("My First Window")


def click(button):
    label1 = Label(master, text = button.lower() + " wins", bg = button)
    label1.pack()



button1 = Button(master,text="Red", command = lambda:click("RED"))
button2 = Button(master,text="Green",command = lambda:click("GREEN"))
button3 = Button(master,text="Blue", command = lambda:click("BLUE"))
button4 = Button(master,text="Yellow",command = lambda:click("YELLOW"))


button1.pack()
button2.pack()
button3.pack()
button4.pack()

mainloop()
```



To show the different ways to organize widgets, we will use an example of a GUI program that includes multiple buttons, and different ways to organize them. The code in the top left corner yields the GUI on the right. There are four different buttons, each defined and packed separately. As you can see, in packing the buttons normally, it will place the first one packed on top, then stack others underneath. The lambda:click() portion of the code is needed rather than just click() for the command parameter because this allows a parameter within the click() function to be sent over. Sending this information eliminates needing a function for each button. This code is further simplified in the right hand code, but yields the same result. x=x is necessary after lambda when defining multiple buttons in a loop, because it makes sure each button gets the correct parameter for click().

```
from tkinter import*

master = Tk()

master.aspect(1,1,1,1)
master.geometry('400x400')
master.title("My First Window")


def click(button):
    label1 = Label(master, text = button.lower() + " wins", bg = button)
    label1.pack()

list1 = ['Red', 'Green', 'Blue', 'Yellow']
button = [0,0,0,0]
for x in range(4):
    button[x] = Button(master, text = str(list1[x]), command = lambda x=x:click(str(list1[x])))
    button[x].pack()

mainloop()
```

CHAPTER 5

Directional Packing

```
1   from tkinter import*
2
3   master = Tk()
4
5   master.aspect(1,1,1,1)
6   master.geometry('400x400')
7   master.title("My First Window")
8
9
10  def click(button):
11      label1 = Label(master, text = button.lower()+ " wins", bg = button)
12      label1.pack(side=RIGHT, fill = Y)
13
14  list1 = ['Red', 'Green', 'Blue', 'Yellow']
15  button = [0,0,0,0]
16  for x in range(4):
17      button[x] = Button(master, text = str(list1[x]), command = lambda x=x:click(str(list1[x])))
18      button[x].pack(side=LEFT, fill = Y)
19
20  mainloop()
21
```

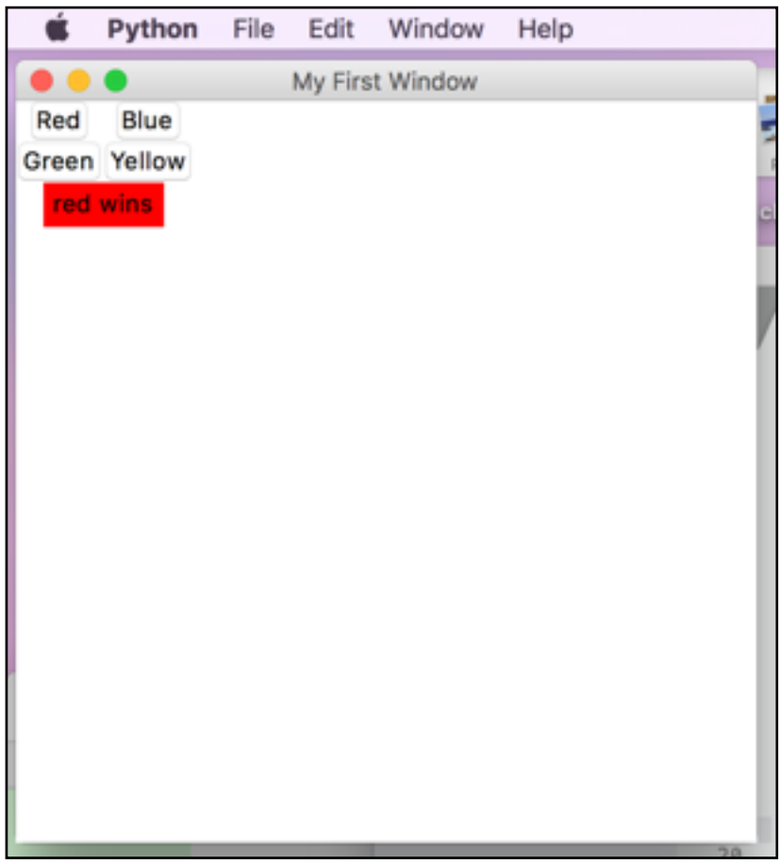
Directional packing permits input on which side to place a widget, not just stacking vertically. As you can see on the right, the layout of the GUI has changed drastically as a result of this organization. Each button is now packed with the parameter side=LEFT. This places each button in the left most space not already taken up, which is why they stack up from the left. In addition, they are packed with the parameter fill = Y, which fills any empty space in the frame/window the button was placed, in this case master, in the y-direction (above or below). Finally, the label that appears upon a button being clicked is packed to the right most space available, and also fills in the y-direction. Directional packing works in every direction (TOP, BOTTOM, LEFT, RIGHT), can fill in both directions, and can take in other parameters, which can be found here:

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

CHAPTER 5

Grid Layout

```
Bookprac.py
1  from tkinter import*
2
3  master = Tk()
4
5  master.aspect(1,1,1,1)
6  master.geometry('400x400')
7  master.title("My First Window")
8
9
10 def click(button):
11     label1 = Label(master, text = button.lower()+ " wins", bg = button)
12     label1.grid(row =2, column =0, columnspan = 2)
13
14 list1 = ['Red', 'Green', 'Blue', 'Yellow']
15 button = [0,0,0,0]
16 for x in range(4):
17     button[x] = Button(master, text = str(list1[x]), command = lambda x=x:click(str(list1[x])))
18
19 button[0].grid(row=0, column = 0)
20 button[1].grid(row=1, column = 0)
21 button[2].grid(row=0, column = 1)
22 button[3].grid(row=1, column = 1)
23
24 mainloop()
25
```



Grid layout is the most versatile organization style. It creates rows and columns, like in an excel sheet, where you can individually place widgets. This allows for more precise arrangement and isn't too complicated. The rows and columns are all relative, that is the lowest row is at the top, and the next highest row is below that. The pattern is the same for colums, except they move from left to right. This allows for as many or as few rows and columns as wanted to be used. Using the grid() function, the code places each button in certain rows and columns, allowing them to be arranged in way impossible with the other organization styles, as shown above. In addition, the label is placed below all the button rows, and spans both columns, as defined with the columnspan parameter. Other grid layout options can be found here:

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

CHAPTER 6

Troubleshooting

Not everything will work perfectly your first try... Here are some helpful solutions to common problems.

My Widgets are Way Too Big!

```
1  from tkinter import *
2
3  window = Tk()
4  frame1 = Frame(window, bg = "black")
5  label1 = Label(frame1, bg = "red", text = "HELLO", height = 3, width = 10)
6  frame1.pack()
7  label1.pack()
8  window.mainloop()
```

This is a very common issue that is caused by the width and height parameters of a widget being defined differently depending on whether there is text within the widget or not. If there is no text within your widget (e.g. a frame) then the width and height are defined in terms of pixels. If your widgets do have text, however, the height and width are defined in terms of character size (e.g. height = 3 would be the height of three characters).

That Dumb Rocket Keeps Bouncing Up and Down and the GUI Won't Launch!

```
1   from tkinter import *
2
3   window = Tk()
4   frame1 = Frame(window, bg = "red", height = 300, width = 300)
5   frame1.pack()
6
7   x = 0 # Perhaps a button changes the value of x
8   while x == 0:
9       frame1.config(bg = "black")
10
11  window.mainloop()
```

This happens when your code never reaches the mainloop() part of your code. The mainloop() is what runs the GUI window, so in order for the window the appear, the code must reach the mainloop(). Because of this, the use of infinite loops as gates to prevent the user from doing certain actions does not work in Tkinter, as it will stop the GUI from running altogether

Function in Button Running Immediately/ Not Running When Button is Clicked!

```
1  from tkinter import *
2
3  def click():
4      print("It worked!")
5
6  window = Tk()
7  but1 = Button(window, text = "Click Me!", command = click())
8  but1.pack()
9  window.mainloop()
```

Not Working

```
1  from tkinter import *
2
3  def click():
4      print("It worked!")
5
6  window = Tk()
7  but1 = Button(window, text = "Click Me!", command = click)
8  but1.pack()
9  window.mainloop()
```

Working

Do not forget that when putting functions in the command = parameter for your button, the parentheses are not needed in your function. This has the side effect of not being able to pass parameters in functions assigned to buttons this way, however. To find a fix for this, please refer to chapter 5.

My Image Will Not Show Up in My Widgets After Using the PhotoImage Function!

```
1  from tkinter import *
2
3  window = Tk()
4  label1 = Label(window, image = PhotoImage(file="dog.png"))
5  label1.pack()
6  window.mainloop()
```

Not Working

```
1  from tkinter import *
2
3  window = Tk()
4  img1 = PhotoImage(file="dog.png")
5  label1 = Label(window, image = img1)
6  label1.pack()
7  window.mainloop()
```

Working

This is because Tkinter does not save the images with widgets. Because of this, the image can be potentially erased if not saved separately from the widget. An easy solution to this problem is to assign PhotoImage() to a variable, and then assign that variable as the image in your widgets.

Sources

- http://effbot.org/tkinterbook/
- https://i.stack.imgur.com/EeNFa.png
- https://i.stack.imgur.com/Qlesu.png
- https://tkdocs.com/images/w_combobox_all.png
- https://stackoverflow.com/questions/27198287/tkinter-create-multiple-buttons-with-different-command-function
- https://stackoverflow.com/questions/1529847/how-to-change-the-foreground-or-background-colour-of-a-tkinter-button-on-mac-os
- https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiDx__k5ZPgAhXsrIMKHQS3BgoQjRx6BAgBEAQ&url=https%3A%2F%2Fcodeburst.io%2Fhow-to-not-suck-at-coding-part-1-bf00502326b0&psig=AOvVaw3h6ND3LRYJ3rOwyv-3NmqU&ust=1548878927637884