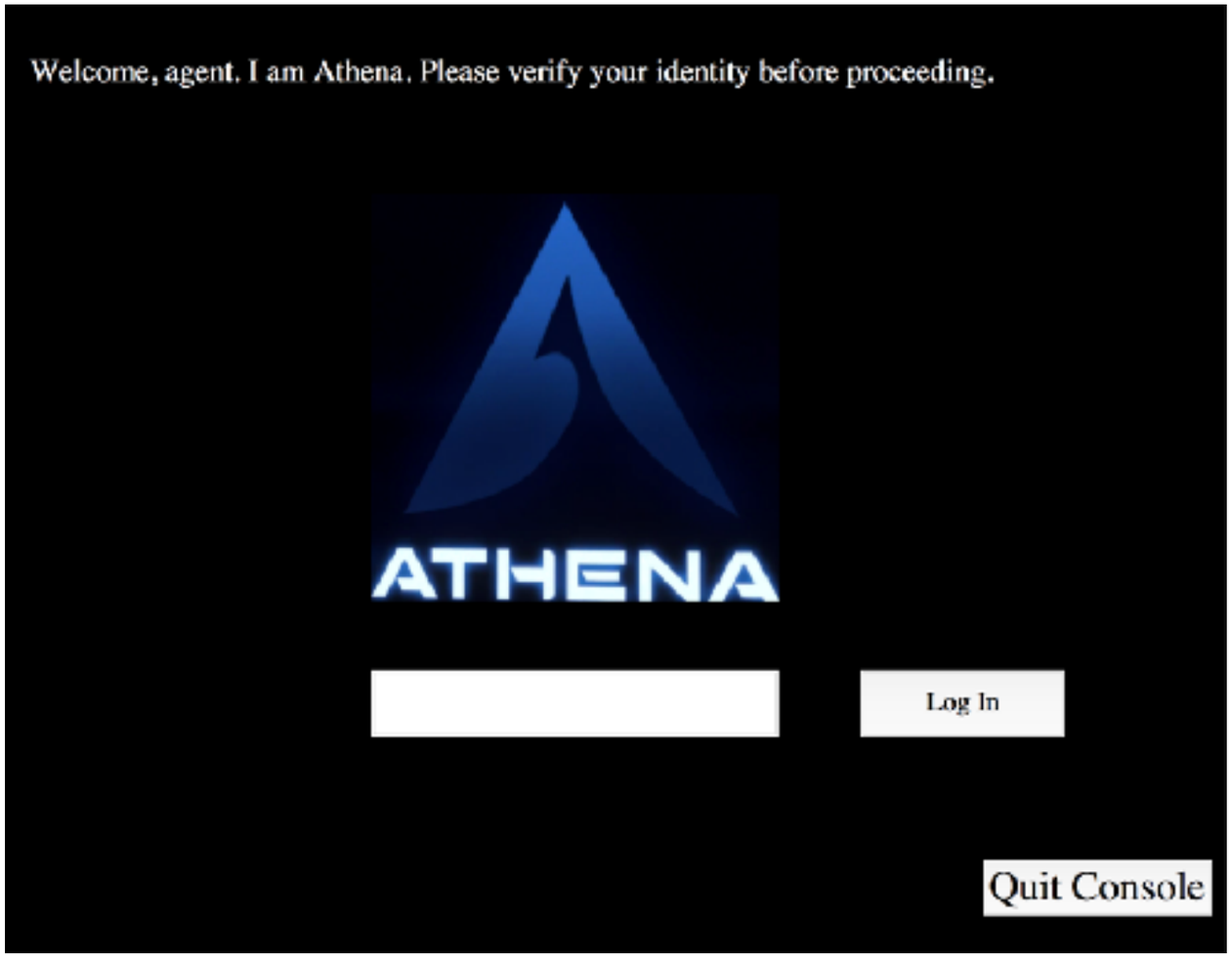
Tkinter A-Z

A Brief Guide to Your First Python GUI

A brief introduction to Tkinter…

Tkinter is a free Python module that is used to create Graphical User Interfaces. It provides the user with a plethora of tools and widgets, such as buttons, input boxes, labels, windows, checklists, and much more. In addition to these widget classes, Tkinter provides the user with a generous helping of customization options regarding color, cursor style, widget placement, and, once again, much more. By the end of this iBook, you will have developed a working basic understanding of how to properly apply Tkinter to make your very own GUI, and you will also be shown where you can acquire further information about the module.



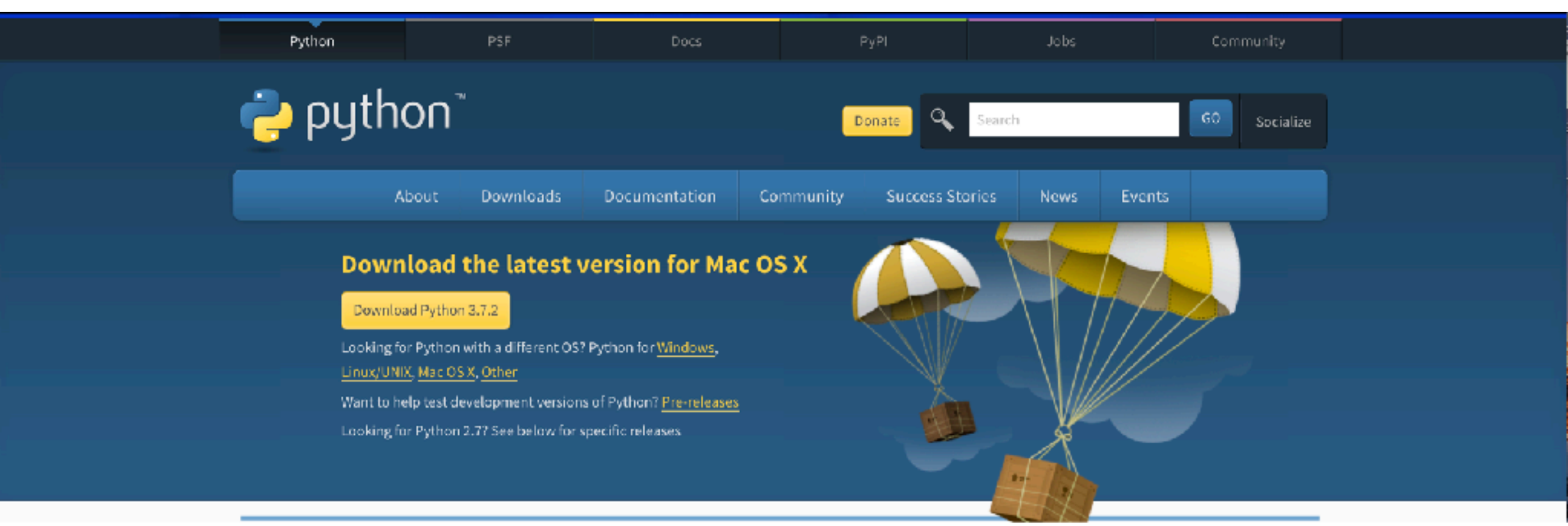
Let's get started…

Installing Tkinter on Macintosh is seamlessly simple. Simply install Python3 and Tkinter comes as an included and in-built module!

Navigate your browser window to the following link:

https://www.python.org/downloads/

And click "Download Python 3.7.2" (note that the Python version number may be different, but as of the time of writing this book it is 3.7.2)



Let's get started…

Installing Tkinter on Macintosh is seamlessly simple. Simply install Python3 and Tkinter comes as an included and in-built module!

Once the package finishes downloading, double-click on the file to engage in the Python3 installation process.

This part should the rather self-explanatory; simply follow the instructions in the wizard, and you have successfully installed Python3 and Tkinter!

Open the syntax-based text editor of your choice (I'd recommend Sublime Text to begin with). Before we can do anything, we need to instruct the computer to include Tkinter commands and classes. To do this, at the very top of the script, we type:

import tkinter

Simple!

```
1   import tkinter
```

However, this method of importing can be a bit annoying. By writing "*import tkinter*", we must include the namespace every time we reference a tkinter function. For example, if we wish to instantiate a button, we must type

myButton = tkinter.Button()

Instead of

myButton = Button()

For the sake of simplicity, it is generally preferred to import tkinter like so:

from tkinter import *

And the rest of this iBook will assume that you have imported tkinter as such.

```
1   from tkinter import *
```

The Main Window

Before we can add widgets of any kind, we must first create the window that the widgets will exist in. For this, we use the Tk() class (AKA the "main window" class). You reference Tk() as you would reference any other class; simply by giving it a name (I generally go with "root") and initializing it.

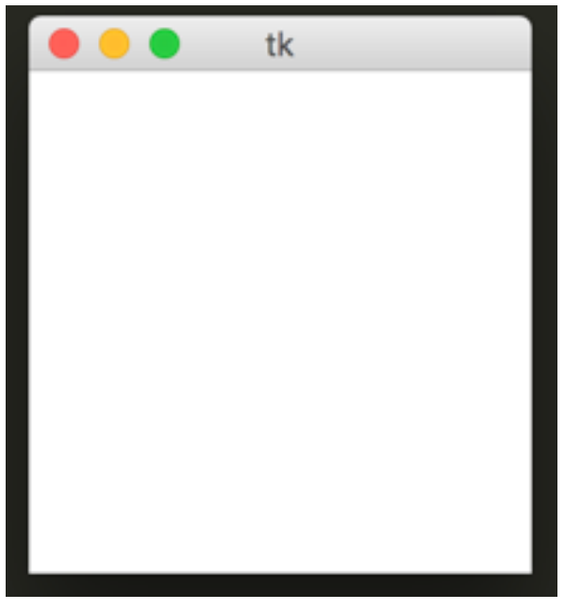
```
root = Tk()
```

Secondly, in order to keep our window, we must apply the *mainloop()* function. Be sure to ALWAYS write this at the BOTTOM of your code, otherwise some of your script may not be applied by the window.

```
root.mainloop()
```

Now let's see what these two lines get us…

OOH! A blank window! That's promising!



Configuring the Root

1. As Tk() is a class, many different parameters can be applied to it. These parameters, as well as the parameters of other widgets, can be discovered via the Tkinter documentation:

http://effbot.org/tkinterbook/tkinter-index.htm#class-reference

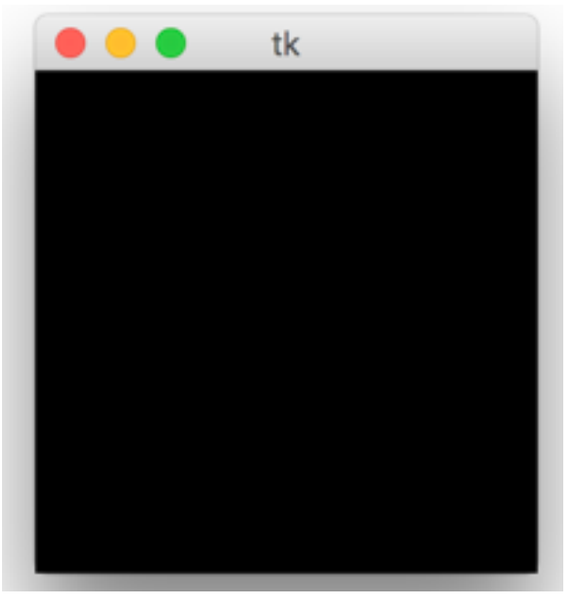
An example of the application of a parameter is shown below.

2. Let's configure our root window, shall we? To do this, we use the *configure()* function. Let's go ahead and tweak the background color of the root window.

root.configure(background="black")

```
from tkinter import *

root = Tk()

root.configure(background="black")

root.mainloop()
```

Voila! A black background window!



3. Some other Tk() parameters to explore are width (integer), height (integer), and cursor (string to determine the cursor style)

REMEMBER: Every class in Tkinter, whether it's a window like the root or a label, is configured in the same way via the same syntax. They each have different configuration options, however, and these are all listed in the documentation link above for your free exploration.

Widgets

Tkinter places a plethora of cool classes at your disposal. These classes are generally referred to as “widgets”. Instantiate these widgets in the same fashion that you instantiated the root window. Simply give it a name, specify the class, enter any configurations, and BOOM!

Specifying the window:
One thing that you absolutely MUST do every time you create a widget is specify its window when configuring it. Without this information, how is the computer supposed to know where you want the widget to appear? We have already created a master window called “root”, so let’s go ahead and specify it when creating a button, one of the many types of widgets.

```
myButton = Button(root)
```

In this line, we are creating a button with the name of myButton, and we are specifying it to be instantiated in the already created “root” master window.

Widgets

Let's explore some parameters of widgets. Each widget has its own parameters. Here's a little table providing you with some parameter examples of three different widgets. All syntax examples, like previous ones, are within a master window that is entitled "root".

Button
Class: Button()

text = (string)

```
myButton = Button(root,text="This is a button!")
```

Set the Button's text.

command = (function name)

```
def buttonfunction():
    print("Button has been pressed!")
myButton = Button(root,command=buttonfunction)
```

The function of the button.

width = (int), height = (int)

```
myButton = Button(root,width=10,height=10)
```

The size of the button.

Input Field
Class: Entry()

textvariable = (name of string variable)

```
var = StringVar()
label = Entry(root,textvariable=var)
```

Set the input of the Entry to a variable

cursor = (cursor type string)

```
var = StringVar()
label = Entry(root,cursor="box_spiral")
```

Set the cursor when hovering over entry

justify = (LEFT, RIGHT, or CENTER)

```
label = Entry(root,justify=CENTER)
```

Set which side of the field the inputed text starts on. Defaulted to LEFT.

Label
Class: Label()

text = (string)

```
label = Label(root,text="This is a label.")
```

Set the label's text.

fg = (color string)

```
label = Label(root,fg="blue")
```

Set the label's text color.

bg = (color string)

```
label = Label(root,bg="red")
```

Set the label's background color.

Unique Variable Types

When looking at the parameter examples on the previous page, you may have noticed a weird class called StringVar() when referencing the textvariable parameter of Entry (or Input) fields. Tkinter has four unique variable classes. These are BooleanVar(), DoubleVar(), StringVar(), and IntVar(). There is nothing separating these from typical Python data types except for the fact that some Tkinter widgets, such as Entry fields and RadioButtons, have the ability to reference them. These are incredibly useful, as these Tkinter widgets are not compatible with standard Python variables. The example of an Entry widget referencing a StringVar as its input-text definition is shown below.

```
var = StringVar()
label = Entry(root,textvariable=var)
```

Widgets

Now let's begin placing these widgets. We haven't ordered our widgets to appear on the screen quite yet!

Let's create a label that says "hi". Simple, yeah?

```
label = Label(text="hi")
```

Window:

What? Why do we only get a blank window when we compile this? Where's the label?

Introducing… geometry managers! These little widget functions will place your widget on the screen!

There are three types of geometry managers. *place(), grid(), and pack().* They all have their own functionalities. Let's go over them, shall we?

Reference these functions in the same way that you would reference any class function.

```
label = Label(text="hi")

label.pack()

label.grid()

label.place()
```

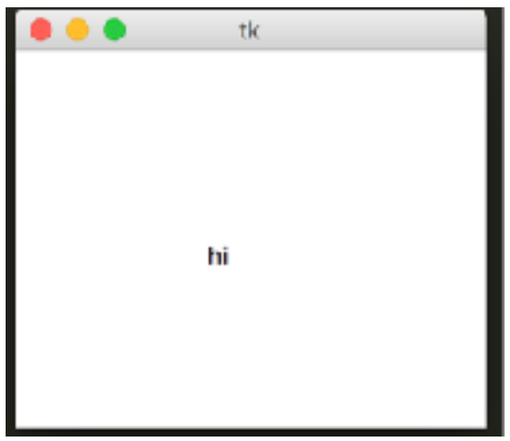
Geometry Managers

place() is easily the most straightforward option of positioning your widget. The main parameters of place() are x = (int in pixels), and y = (int in pixels). The following code places the label 100 pixels to the right and 100 pixels down, starting from the top left corner.

You may also use relx = (float), rely = (float). These parameters take float values between 0 and 1 as their input, and position the widget relative to the size of the window, with 0 being 0% of the window's width/height and 1 being 100% of it.

```
label = Label(text="hi")

label.place(x=100,y=100)
```

Window:



grid() is generally used to organize widgets into a grid. The primarily used parameters of the function are row = (int) and column = (int).

```
label = Label(text="hi").grid(row=0,column=0)

label2 = Label(text="hi").grid(row=1,column=0)

label3 = Label(text="hi").grid(row=0,column=1)

label4 = Label(text="hi").grid(row=1,column=1)
```

Window:

pack() is used to position widgets relative to the edges of the window. The most commonly applied parameter of pack() is side = (LEFT, RIGHT, TOP, or BOTTOM), which places the widget on the specified edge. If a widget is already present on that side, it automatically places the widget next to the preexisting widget. Using parameters such as padx = (int) and pady = (int), you can specify how distant to widgets are to be relative to each other, essentially providing each widget with a personal bubble. The same parameters can be applied to grid(). This makes pack() the most autonomous and convenient, yet least flexible option.

Code example:

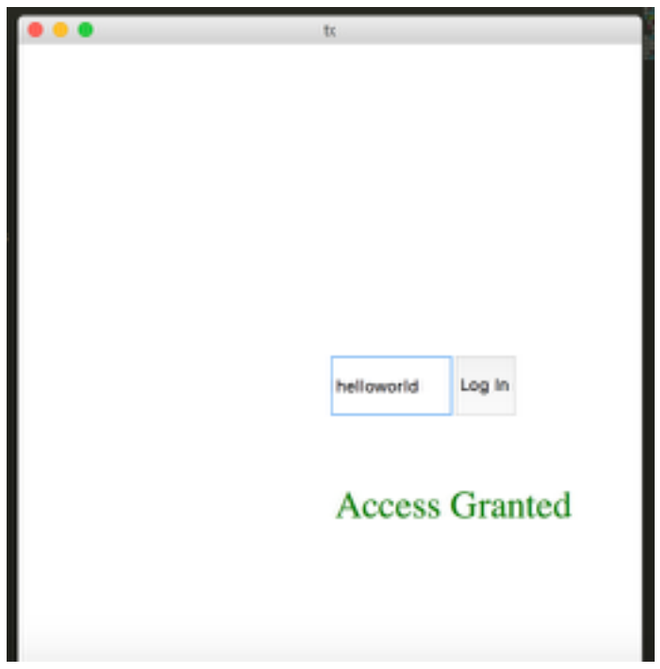
```
label = Label(window,text="hi").pack(side=LEFT,padx=100) # Pack two labels on the left of the screen, total of 200 pixels away from each other.
label2 = Label(window,text="hi2").pack(side=LEFT,padx=100)
```

Window:

Now let's apply it all, using everything we've learned as well as some extra information from the linked Tkinter documentation, namely widget font configurations (one of the many configurations) and some widget functions such as entry.get()

```
from tkinter import *

root = Tk() # Create the window

root.geometry('500x500') # Set window size using one of the many functions in the Tk() class

def LogIn():

    txt = entryfield.get() # Get() function from Entry class, discovered in Tkinter documentation

    if txt == "helloworld": # If the input text is equal to a password

        accessgranted = Label(text="Access Granted",fg="green",font=("times",30)).place(relx=0.5,rely=0.7) # Create a green label that says "Access Granted"

root.configure(background="white",cursor="pirate") # Set background to white and set a custom cursor

entryfield = Entry() # Create entry field

entryfield.place(relx=0.5,rely=0.5,width=100,height=50) # Place entry field

button = Button(text="Log In",command=LogIn).place(rely=0.5,relx=0.7,width=50,height=50) # Create and place button

root.mainloop() # MAIN LOOP
```

Window output:



BONUS TIP: To make a widget disappear, use the grid_forget(), place_forget(), or pack_forget() functions, depending on which geometry manager you chose.

Congratulations! You are now familiar with the basics of Tkinter!

Once again, please reference this Tkinter documentation for more comprehensive information.

http://effbot.org/tkinterbook/

Sources

Tkinter Documentation: http://effbot.org/tkinterbook/

Python Logo Image: https://commons.wikimedia.org/wiki/File:Python-logo-notext.svg

On my honor, I have neither given nor received unauthorized aid.