

Application Note:

Creating a Python Graphical User Interface

Matthew Roach

March 31st, 2014

Abstract: This document contains 2 portions. First, it provides an introduction into phase and the use of phase measurement to arrive at a calculated angle of arrival. Second, it provides an introduction into creating a Graphical User Interface in Python.

Keywords: Phase, Phase Difference, Angle of Arrival, Python, GUI (Graphical User Interface)

Introduction:

A Graphical User Interface, or GUI for short, allows a user to interact with their program and or hardware as opposed to operating through a command line or strictly running a script. It allows the creator to represent his or her program or hardware interaction in a visual setting, allowing for easier control and usage. A GUI simplifies and creates easier access to more technical less known operations.

Objective:

The objective of this application note is to understand the basics in building a GUI. It will focus on the key portions and building blocks to creating a simple and usable GUI. This application note will be using Python as the programming language. Python is simple and easy to use for a broad variety of applications.

Software Requirements:

To begin, the first requirement needed is the Python software. This can be downloaded free of charge from <https://www.python.org/>. Please be advised that the use of version 2.7.6 or above is highly recommended as these are the most recent and supported versions. After successfully downloading and installing Python, when opening IDLE (Python GUI), you should see a window similar to the one below in Figure 1.

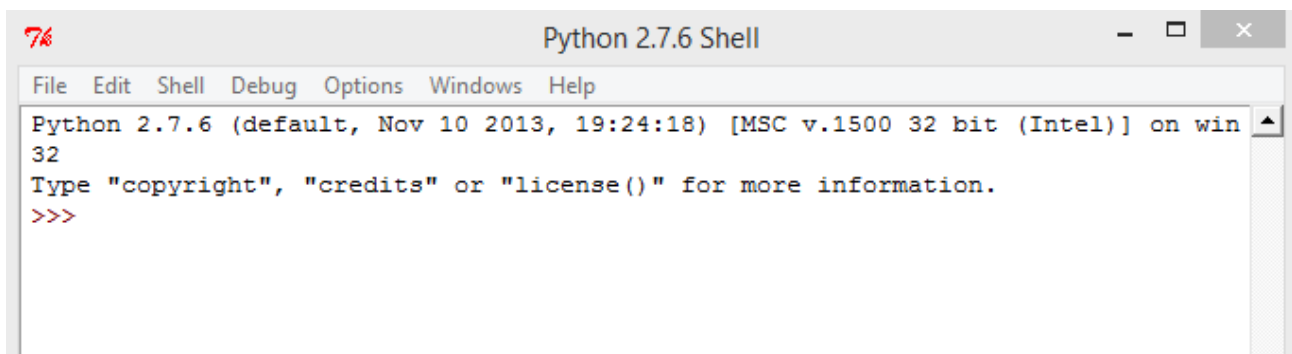


Figure 1 – IDLE (Python GUI) Window

The window seen above in figure 1 is actually a GUI itself, while obviously more complex; it is similar to what one would build when creating a GUI. Note the “Tk” in the upper left of the window, this application note will reference “Tk” later and expand on it.

Tk Module:

As noted above, the “Tk” stands for Tkinter Module, also known as Tk interface. This module provides widget classes and associated constants to use in building your GUI. To use this module simply select File > New File and begin with the following code:

```
from Tkinter import *
```

Using the command “import Tkinter” is also acceptable, but the command given above is the recommended course of action for this application note.

Root Window:

The next step in creating your GUI is to begin by creating your window where your application will appear, as seen in Figure 1 below. To do this we place the following command in your code.

```
root = Tk()  
root.mainloop()
```

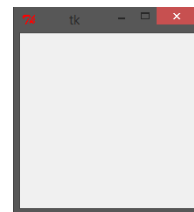


Figure 2 – Blank Window

For test your GUI, in the code editing window select Run > Run Module. Figure 3 above is an example of what you should see after using the three lines of code provided above. The first command is the initial code that creates our initial Tk root widget. This is only needed once for your GUI. The second command used is known as the “event loop”, without this your window will not appear. For this application note, all code examples given should be placed between the Tk root and the event loop.

Widgets:

Widgets, simply put, are all the labels, button, boxes, etc., that will be placed inside your root window. The following portion of this application note will focus on implementing different widgets into the root window constructed in the previous section. There are many widgets that can be implemented and used for a variety of applications, but this application note will focus on a few core building blocks that are commonly used in most GUI.

- Label Widget:

The first and most basic widget is the “Label Widget”. The label widget displays text and images. Insert the code below between the Tk root and event loop as specified above:

```
LabelName = Label(root, text="Label Widget")
LabelName.pack()
```

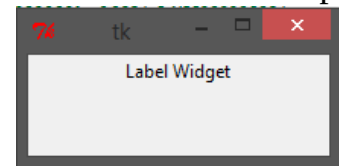


Figure 3 – Label Widget

The first line of the code represents the actual label as displayed in Figure 3 above. It is in this command where extra features would be added such as font, color, etc. Figure 3 represents the default option if no other options are included. The second line is called the “pack method”. This command sizes the widget to the given text.

- Frame Widget:

One aspect of designing a GUI is the decoration of it. One way in which to add accent to your GUI is to implement a frame widget. While purely used for decorative purposes, the frame widget can be very useful in dividing information and separating different sections of your GUI. The following code includes the use of two label widgets above and below the frame widget code for demonstration purposes. Insert the code below:

```
LabelName = Label(root, text="Label Widget")
LabelName.pack()

FrameName = Frame(height=2, bd=1, relief=SUNKEN)
FrameName.pack(fill=X, padx=20, pady=10)

LabelName2 = Label(root, text="Label Widget 2")
LabelName2.pack()
```

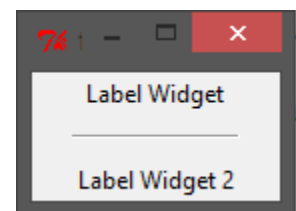


Figure 4 – Frame Widget

There are a couple things to note regarding the frame widget. First it begins just as the label widget does, but it contains information regarding the size and look of the frame. The “height” variable adjusts the thickness of the frame vertically. “Relief” and “bd” adjust the style and to what extent that particular style is applied. In this instance we used “SUNKEN” which will display the frame as “SUNKEN” into the GUI. The next line of code deals with the pack method as referenced above. The “fill” function refers to the background color while the “padx” and “pady” refer to “padding” around the frame. An increase in the latter two would provide more room between the frame and the label widgets as demonstrated above in Figure 4. Similar to the label widget, there are many more options available to the frame widget providing customization for your particular GUI.

- Check Button Widget

The check button widget is another very useful widget that is widely used in a large number of GUIs. When not connected to something though, this widget becomes ultimately useless. From the previous widget example, the second label widget has been replaced by a check button. Insert the code below:

```
LabelName = Label(root, text="Label Widget")
LabelName.pack()

FrameName = Frame(height=2, bd=1, relief=SUNKEN)
FrameName.pack(fill=X, padx=5, pady=5)

var = IntVar()

CheckButtonName = Checkbutton(root, text="Expand", variable=var)
CheckButtonName.pack()
```

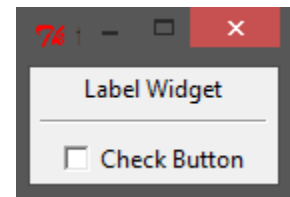


Figure 5 – Check Button Widget

In Figure 5 located above notice how the check button box falls below the frame widget. As with all the other widgets the check button widget too includes customizable features. The one unique feature regarding this widget that differs from the previous two is that this widget is functional. Because of this we will need to keep track of the selection (either a 1 or 0); hence the variable declaration located above the check button statement. Once again we will use the pack method directly after the widget.

- Entry Widget

Similar to the check button widget as discussed above regarding functionality, the entry widget is also a commonly used widget throughout many GUIs. The entry widget provides a text input interface for the user. From the code

in the previous example, the entry widget code has replaced the check button widget. Insert the code below:

```
LabelName = Label(root, text="Label Widget")
LabelName.pack()

FrameName = Frame(height=2, bd=1, relief=SUNKEN)
FrameName.pack(fill=X, padx=5, pady=5)

word = StringVar()
TextWidget = Entry(root, textvariable=word)
TextWidget.pack()
word.set("default")
```

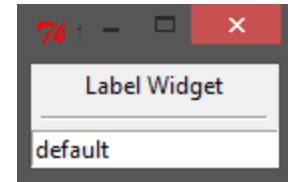


Figure 6 – Entry Widget

As shown in Figure 6 above, the text “default” is where a user would input his or her text if required to do so. The code required for the entry widget is similar to the previous widget in that because it has the functionality of an input there is a need to save that input. In this case the input would be saved as a string. The last line of the segment of code refers to the default text that will be shown inside the entry box initially. In this case it is specified to say “default”.

Conclusion:

The programming language of Python can be a very valuable tool in developing both advanced and basic GUIs. Python allows for easy manipulation and customization to fit whatever end user experience is required. There is a wide variety of widgets and design schemes that were not mentioned in the above that can be used just as easily as demonstrated in this application note.

Recommendations

This application note provides a good base understanding for creating a GUI in python. Though not mentioned, there are many other aspects regarding GUI that would be highly beneficial to educate yourself on before beginning your GUI. Recommendations include placement and mapping of widgets, creating a .exe (executable), and also linking to other sources. These are just three of many aspects

that are commonly used in GUIs and would be beneficial to learn before proceeding further.