GUI programming is an art, and like all art, you need a drawing board to capture your ideas. The drawing board you will use is called the root window. Our first goal is to get the root window ready.

The following screenshot depicts the root window we are going to create:

Drawing the root window is easy. You just need the following three lines of code:

```
import tkinter as tk
root = tk.Tk()
# all GUI code goes here
root.mainloop()
```

Running this program should generate a blank root window as shown in the preceding screenshot. This window is furnished with functional minimize, maximize, and close buttons, and a blank frame.

The description of the code is as follows:

- The first line imports all (*) classes, attributes, and methods of Tkinter into the current workspace. (More about this particular import described later in this document)
- The second line creates an instance of the class tkinter.Tk. This creates what is called the "root" window that you see in the screenshot provided. By convention, the root window in Tkinter is usually called "root", but you are free to call it by any other name.
- The third line executes the mainloop (that is, the event loop) method of the root object. The mainloop method is what keeps the root window visible. If you remove the third line, the window created in line 2 will disappear immediately as the script stops running. This will happen so fast that you will not even see the

window appearing on your screen. Keeping the mainloop running also lets you keep the program running until you press the close button, which exits the main loop.

Commit the three lines of code to memory. These three lines generate your root window, which will accommodate all other graphical components. These lines constitute the skeleton of any GUI application that you will develop in Tkinter. All code that will make your GUI application functional will go between line 2 (new object creation) and line 3 (`mainloop`) of this code.

Tkinter Imports

This section describes the different styles of importing Tkinter modules.
In the preceding example, we imported Tkinter using the following command:

```
from tkinter import *
```

This method of import eases the handling of methods defined in the module. That is to say, you can simply access the methods directly. Generally, it is considered a bad practice to import all (`*`) methods of a module like we did here. This is because if you import all methods from some other module with a common method name, it would lead to the overwriting of methods.

There are several ways to import Tkinter in which this overlapping can be avoided, a common one being:

```
import tkinter
```

This style of importing does not pollute the namespace with a list of all methods defined within Tkinter. However, every method within Tkinter will now have to be called using the format `tkinter.methodA` instead of directly calling the method.
Another commonly used import style is as follows:

```
import tkinter as tk
```

Here too, you do not pollute the current namespace with all Tkinter methods and now you can access methods such as `tk.methodA`. "tk" is a convenient, easy-to-type alias commonly used by many developers for importing Tkinter.