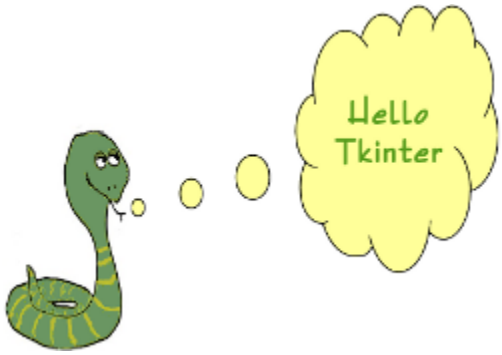
Tkinter Widgets

Hello Tkinter Label



We will start our tutorial with one of the easiest widgets of Tk (Tkinter), i.e. a label. A Label is a Tkinter Widget class, which is used to display text or an image. The label is a widget that the user just views but not interact with.

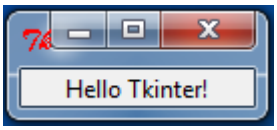
There is hardly any book or introduction into a programming language, which doesn't start with the "Hello World" example. We will draw on tradition but will slightly modify the output to "Hello Tkinter" instead of "Hello World".

The following Python script uses Tkinter to create a window with the text "Hello Tkinter". You can use the Python interpretor to type this script line after line, or you can save it in a file, for example, "hello.py":

```
import tkinter as tk


# if you are still working under a Python 2 version,
# comment out the previous line and uncomment the following line
# import tkinter as tk


root = tk.Tk()


w = tk.Label(root, text="Hello Tkinter!")
w.pack()


root.mainloop()
```

Starting our example

If we save the script under the name hello.py and run the code under Windows it appears like this:



Explanation

The tkinter module, containing the Tk toolkit, has always to be imported. In our example, we imported tkinter by renaming it into tk, which is the preferred way to do it:

```
import tkinter as tk
```

To initialize tkinter, we have to create a Tk root widget, which is a window with a title bar and other decoration provided by the window manager. The root widget has to be created before any other widgets and there can only be one root widget.

```
root = tk.Tk()
```

The next line of code contains the Label widget. The first parameter of the Label call is the name of the parent window, in our case "root". So our Label widget is a child of the root widget. The keyword parameter "text" specifies the text to be shown:

```
w = tk.Label(root, text="Hello Tkinter!")
```

The pack method tells Tk to fit the size of the window to the given text.

```
w.pack()
```

The window won't appear until we enter the Tkinter event loop:

```
root.mainloop()
```

Our script will remain in the event loop until we close the window.

Colorized Labels in various fonts

Some Tk widgets, like the label, text, and canvas widget, allow you to specify the fonts used to display text. This can be achieved by setting the attribute "font". typically via a "font"

configuration option. You have to consider that fonts are one of several areas that are not platform-independent.

The attribute fg can be used to have the text in another colour and the attribute bg can be used to change the background colour of the label.

```
import tkinter as tk


root = tk.Tk()
tk.Label(root,
                text="Red Text in Times Font",
                fg = "red",
                font = "Times").pack()
tk.Label(root,
                text="Green Text in Helvetica Font",
                fg = "light green",
                bg = "dark green",
                font = "Helvetica 16 bold italic").pack()
tk.Label(root,
                text="Blue Text in Verdana bold",
                fg = "blue",
                bg = "yellow",
                font = "Verdana 10 bold").pack()


root.mainloop()
```

The result looks like this:

Dynamical Content in a Label

The following script shows an example, where a label is dynamically incremented by 1 until the stop button is pressed:

```
import tkinter as tk


counter = 0
def counter_label(label):
  def count():
    global counter
    counter += 1
    label.config(text=str(counter))
    label.after(1000, count)
  count()



root = tk.Tk()
root.title("Counting Seconds")
label = tk.Label(root, fg="green")
label.pack()
counter_label(label)
button = tk.Button(root, text='Stop', width=25, command=root.destroy)
button.pack()
root.mainloop()
```

The result of the previous script looks like this: