1.4 Coloring your widgets

Adding colors

Nearly everything you put inside your windows may be **colored**. Most widgets have dedicated properties to handle their colors and we will tell you about them while discussing the widgets themselves. Currently, the most important thing is getting to know how the colors are described in `tkinter`, in other words, what means can you use to order the button to be red or blue.

There are at least three methods designed to meet your needs. We will tell you about them on the example of `Button` but don't forget that these ways are universal and can be used virtually everywhere.

Let's check if tkinter understands **English** – look at the code in the editor, our test is there.

```
import tkinter as tk

window = tk.Tk()
button = tk.Button(window, text="Button #1", bg="red", fg="yellow")
button.pack()
window.mainloop()
```

Note the two new arguments we use in the constructor invocation: `bg` (what is a short form of *"background-color"*) and `fg` (*"foreground-color"*). We went along the line of least resistance here – we've just made use of regular English names of colors and packed them inside strings.

Does it work? Let's check.

Yes, it definitely does:

We encourage to you make some clicks on the **button** – it will unveil its little secret.

Can you see? The colors of the lowered (pressed) button are **gray** still. Why?

Because `fb` and `bg` refer to **raised** buttons only. There two additional parameters describing the second set of colors named `activeforeground` and `activebackground` respectively used by the event controller when the button is **pressed**. Do you want to check how they work? Do it boldly!

We can summarize the test's result saying that any of commonly used English color name can be used with `tkinter`. Don't bother if you want some of your widgets to be simply **white**, **black**, **green**, **gray** or even **grey**. It's easy and handy although not very precise.

Tkinter can do something more for you.

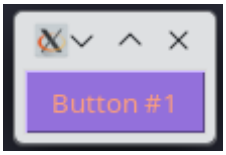
Tkinter recognizes over 750 predefined color names – all of them can be found here.

Feel free to use them – we've showed our try in the editor.

```
import tkinter as tk
```

```
window = tk.Tk()
button = tk.Button(window, text="Button #1",
                   bg="MediumPurple",
                   fg="LightSalmon",
                   activeforeground="LavenderBlush",
                   activebackground="HotPink")
button.pack()
window.mainloop()
```

This is what we get:



Yes, it definitely does:

The third method you can use is based on the fact that each color can be obtained by **mixing** (adding) **three primary colors**: **red** (R), **green** (G) and **blue** (B). The phenomenon is utilized by the so-called **RGB color model** which is one of the additive color models and it's widely used in many application, e.g. in color displays of different kinds.

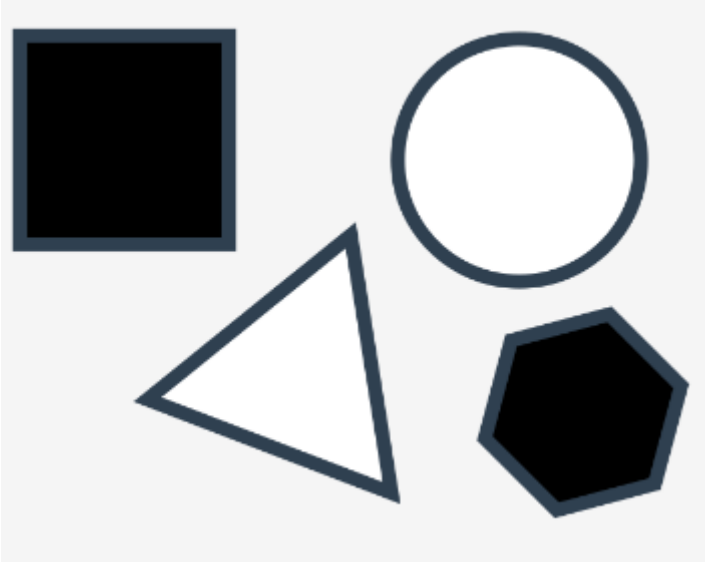
One of the RGB model implementations allows you to set the **saturation** of every of primary colors in the range <0..255> what gives **256 different saturation levels**, from color's absence (saturation 0) to full color's presence (saturation 255).

Do you think it's not too much? Maybe, but don't forget that you mix three different colors (so-called **color components**) so the full spectrum consists of 256*256*256 = 16,777,216 colors. An average human can distinguish about 7 million colors, consequently, the model should work well and it really does.

Let's take a closer look at this.

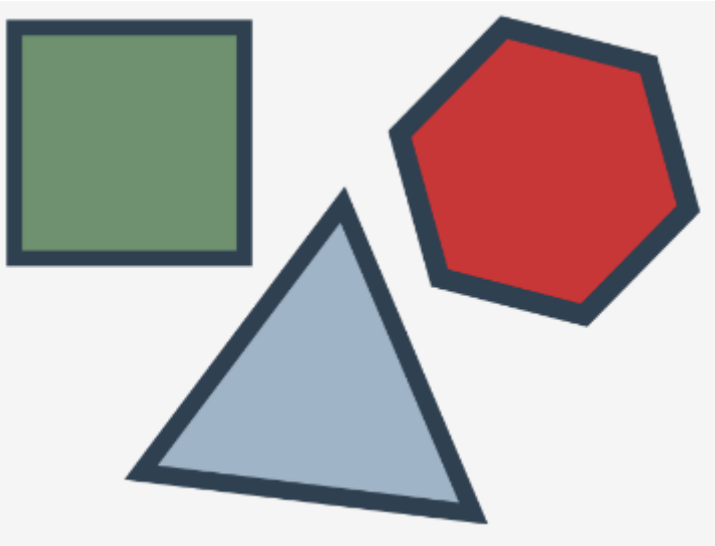
When all the components are set to zero (absence of the colors), we get black as a result.

When all the components are set to 255 (full presence of the colors), we get white as a result.



When one of the components is set to 255 while others are set to 0, we get one of the primary colors – red,

green or blue.



Setting non-zero values for more than one component produces **intermediate colors**, e.g. red and blue set to 255 with green set to 0 gives the **violet**.

It seems clear, doesn't it?

OK, but how to specify all these (more than 16 million) mixes in a comprehensive way?

It's is done by a trick used extensively in web pages – as a **string** starting with a **hash** (#) followed by **6 hexadecimal** digits. Each p**air of the digits** forms the value from range **0x00..0xFF** (0..255) what determines the specific component level.

This is how it works:

All three pairs (RR, GG and GG) are **two-digit hexadecimal number** so:

- #000000 is **black**
- #FFFFFF is **white**
- #FF0000 is **red**
- #00FF00 is **green**
- #0000FF is **blue**
- #00FFFF is **turquoise**
- #FF00FF is **violet**
- ...

Please, forgive us that we made that list so short. To tell the truth, we have no idea how to name each of these 16 million colors – do you?

Note: when all the components are set to the **same value**, equal neither to zero nor to 0xFF (e.g. #0F0F0F0), you will get **254 shades of gray**.

Look at the code in the editor. We want to show you a little test showing how the RGB works:

```
import tkinter as tk

window = tk.Tk()
button = tk.Button(window, text="Button #1",
                   bg="#9370DB",
                   fg="#FFA07A",
                   activeforeground="#FFF0F5",
                   activebackground="#FF69B4")
button.pack()
window.mainloop()
```

Do the colors look the same as in the previous code?

They should look the same as we used RGB equivalents of the previously used `tkinter` color names.

Now try to find RGB codes for all your favorite colors. There are so many choices – don't feel lost!