Name:	Date:

Colors

Photo by Lucas George Wendt on Unsplash

Pick your favorite color from this selection of colored pencils.

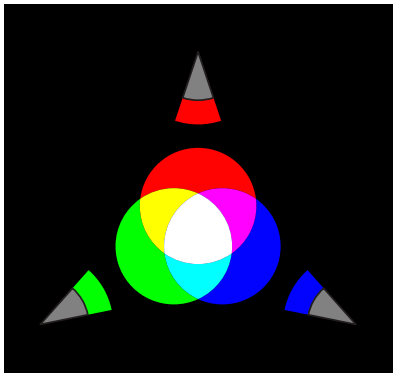
Describe your favorite color in a way that your classmate can figure out your choice.

Have they been able to guess your favorite color, with the correct shade?

In programming we have to be accurate without any ambiguity.

Creating Colors by Adding Lights

Imagine you have three colored flashlights: **red**, **green**, and **blue**.



Which color do you get by overlaying the lights in the following way?
Complete the table:

Red	Green	Blue	Mix
off	off	off	black
off	off	on	blue
off	on	off	
off	on	on	
on			
on			

With three flashlights, each of which can be turned off or on, you can get exactly 8 different colors. The table above has 8 rows, one for each possible color.

Why can one get 8 different colors?
Each light has two states, on or off. In total, we have three lights.

$$2 \cdot 2 \cdot 2 = 2^3 = 8$$

If now each flashlight instead of having only two states (on and off) is dimmable (has an adjustable brightness), obviously the situation changes.
Let's say our flashlight has 256 different brightness levels, where 0 means off and 255 means on with the brightest light.
How many different colors can we get with 3 flashlights of this type?

$$256 \cdot 256 \cdot 256 = 256^3 = 16'777'216$$

That's more than 16 million different colors! Your computer screen is capable of creating all these 16'777'216 combinations.
If you buy a box of colored pencils you end up with maybe 12 different colors... obviously-you could produce other shades with them... but what a lot of work...

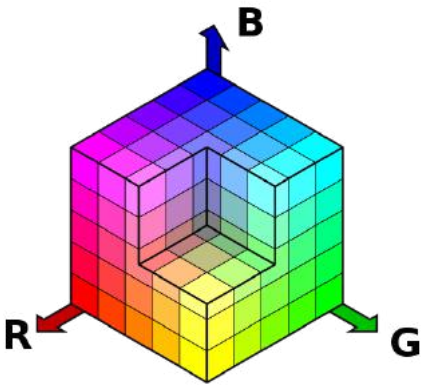
The RGB Color Cube

We can define a color by combining the various shades of the 3 colors red, green and blue. This color specification consists of three numbers, here is an example:

76, 172, 35

How can these three numbers describe a color?
We can imagine a color as a point in a three-dimensional Cartesian reference system, with the x-axis (R = red), y-axis (G= green) and z-axis (B= blue):
A possible command could be:

- go 76 steps in the R direction (red, x axis)
- go 35 steps in the G direction (green, y axis)
- go 172 steps in the B direction (blue, z axis)



The space of the set of possible colors represents a cube, where each dimension ranges from 0 to 255.

Complete the following table and identify the colors of the vertices of the cube:

Red	Green	Blue	Mix
0	0	0	Black
255	255	255	White
255	0	0	
255	128	0	Orange
255	255	0	
0	255	0	
			Cyan
			Blue
	0	255	Purple
255	0	255	
255	128	255	Pink

RGB Colors in Office Programs

In the various office applications (Word, Excel, PowerPoint) you can select predefined colors or define your personalized colors:

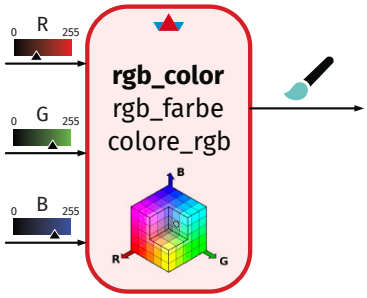
Predefined Colors	Definition of Your RGB Color
	

Open an office application and choose three different colors to your liking.
Use the RGB slider to set the three values (red, green, blue).
Write them in the table:

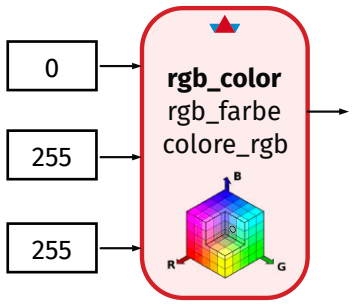
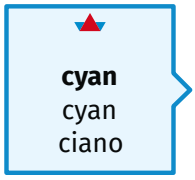
Red	Green	Blue	Name
			Favorite Color #1:
			Favorite Color #2:
			Favorite Color #3:

RGB Colors with PyTamaro

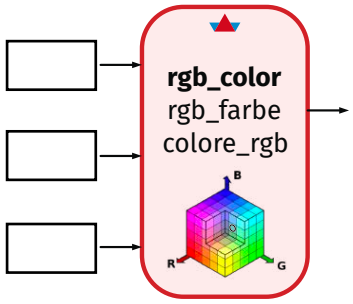
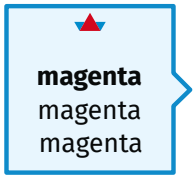
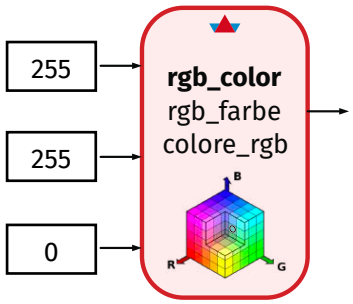
To create a color using red, green, and blue as components, PyTamaro provides a **function** named `rgb_color` with three **parameters**:



The following programs are equivalent:

Calling a Function:	**Using a Constant:**
 `rgb_color(0, 255, 255)`	 `cyan`

Complete the following table:

Calling a Function:	**Using a Constant:**
 `rgb_color(  ,  ,  )`	 `magenta`
 `rgb_color`	

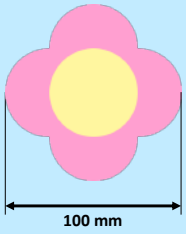
Naming Colors (Defining Constants)

Let's program a pink flower:

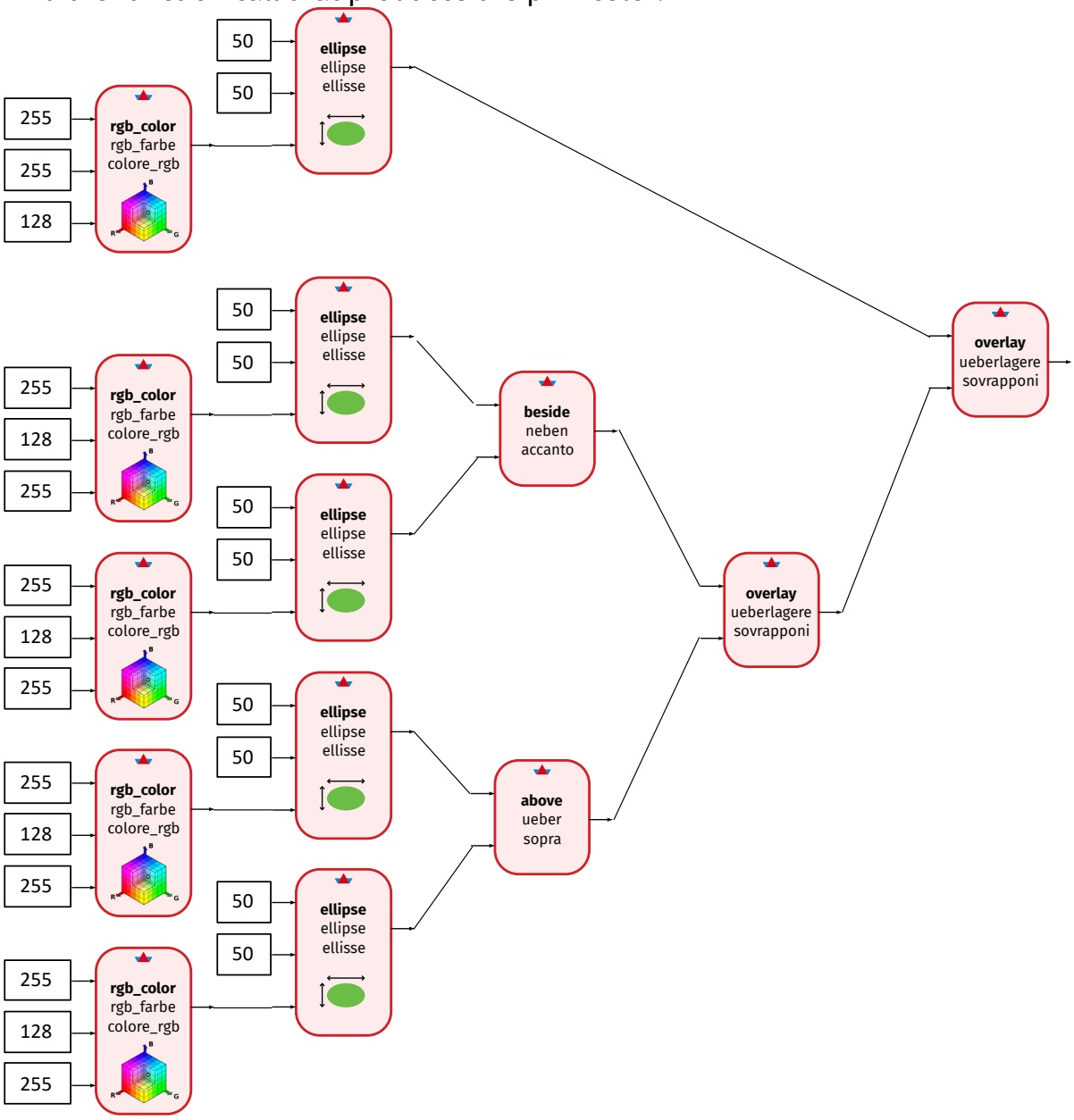
There is no constant for the pink color, thus we have to call the **function** `rgb_color` to create the pink color.

Let's use the same approach for the light yellow color.

Desired Graphic

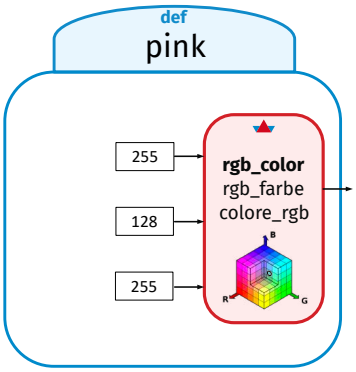


Here is the program that produces the flower.
Find the function call that produces the pink color:



It is quite tedious to rewrite the same color function several times, each time we need it. In the flower example we wrote the pink function as many as 4 times!
Of course there is a more effective solution!
In programming we can **define a name** and use it later when we need it.

Let's define the names of our pink and light-yellow colors!
Complete the following table:

Definition of the name `pink`**:**	**Definition of the name** `light_yellow`**:**
	
`pink = rgb_color(255, 128, 255)`	

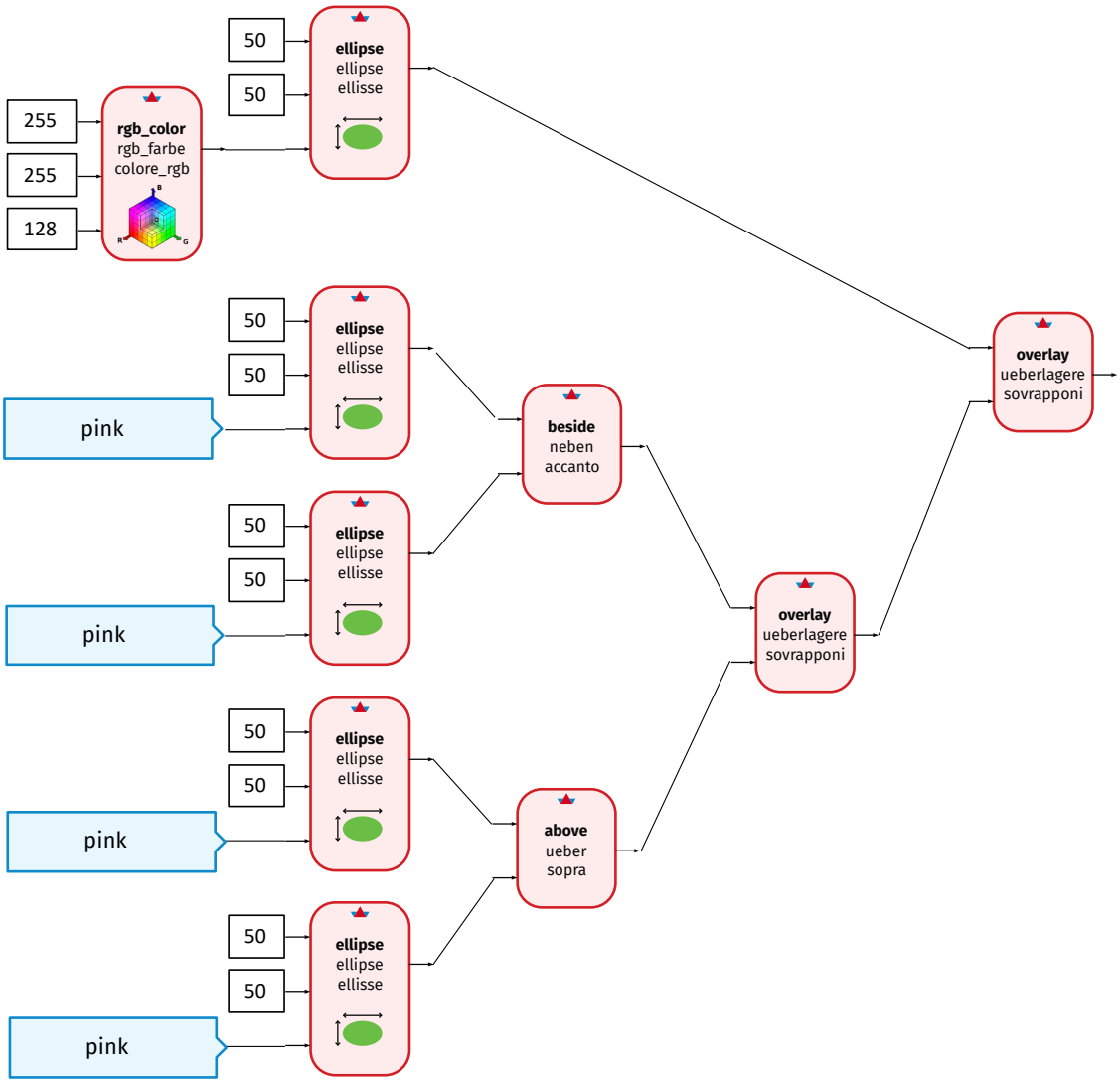
We **defined** a **constant**. In Python we need to write the **name**, followed by an **equals** sign, followed by the **expression** that produces the value of our constant:

name = expression

```
pink = rgb_color(255, 128, 255)
```

Once a **name** is defined, we can simply use that **name** (**constant**), instead of having to write the entire expression (that could be very complex).

Here is our flower program, where we substituted the calls to `rgb_color` with the name `pink`:



Use the constant `pink` in the "*Utilizzo delle costanti*" activity on the PyTamaro site.

What do you have to change in the program above to use the name `light_yellow`?

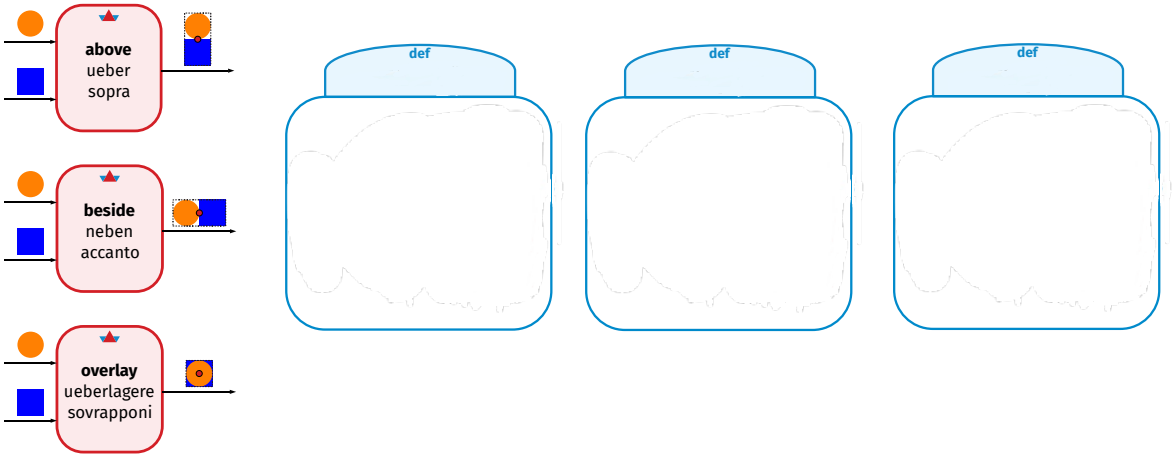
You learned to create constants. Are there other parts that could be simplified in the flower program? Are there other pieces of code that show up multiple times and could be replaced by giving them a name?

Write the Python code to create the constants `light_yellow`, `petal`, and `pistil`.

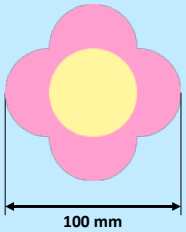
```
light_yellow =

petal =

pistil =
```

Now program the flower again, using these new constants!

Library



Desired Graphic



Create the program with the cards:

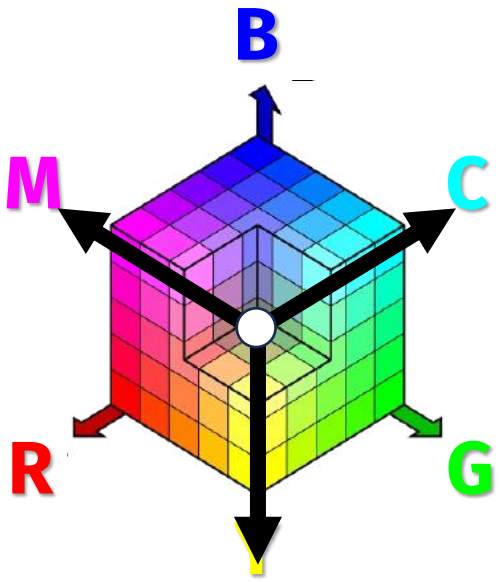
Rewrite the code of the flower using all the new constants on the PyTamaro site in the "*Utilizzo delle costanti*" activity.

CMY Color Model – How Printers Create Colors

The RGB model works with the **additive** method: we start at the **black** corner (origin in the Cartesian three-dimensional reference system) by **adding** the lights produced by the three different flashlights.

We can also start in the color cube from the opposite corner: we start in the **white** corner, **subtracting** lights (covering a white page with colors). This model is based on **subtraction** and is used in the CMY color model found inside printers.

We start with white and go ahead with: - cyan (C) - magenta (M) - yellow (Y) And that's how a printer works!	 Photo by tookapic from Pixabay



The RGB and CMY models are two different ways to navigate the same cube!

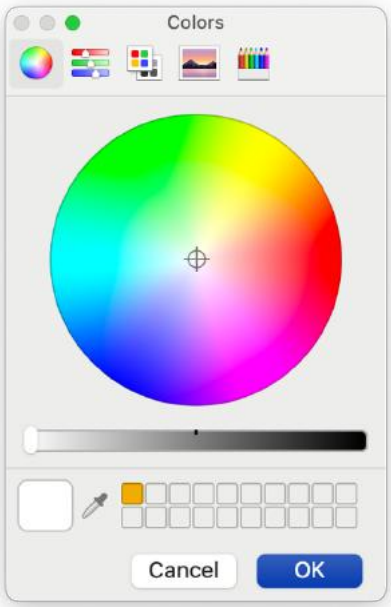
Model	Each step...	Starting point	Dimen-sion	Dimen-sion	Dimen-sion
RGB	**add** colors	**black** RGB: 0,0,0	red	green	blue
CMY	**subtract** colors	**white** RGB: 255, 255, 255	cyan	magenta	yellow

Models Based on Hue

Besides the RGB and CMY models there are other ways to define colors. For example, there are ways based on the **hue**.

The hue is specified as an angle, from 0 to 360 degrees.

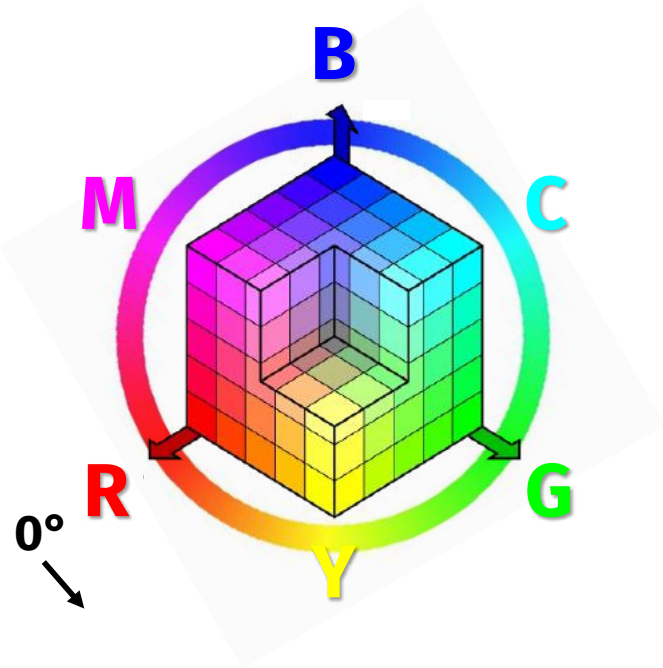
Color Circle



Angle: hue
Radius: saturation
Slider: value

Slider 1: hue
Slider 2: saturation
Slider 3: value

Relationship Between Color Circle and RGB Cube



Complete the table with the angles (in degrees) for each corner of the RGB / CMY color cube:

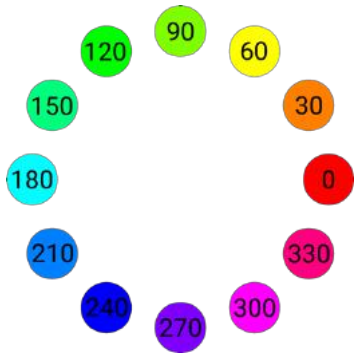
Color	Angle
R red	0°
G green	
B blue	
C cyan	
M magenta	
Y yellow	

Hue-Saturation-Value (HSV) = Hue-Saturation-Brightness (HSB)

A model based on the hue is the hue-saturation-value (HSV) model, also known as hue-saturation-brightness (HSB).

To better understand the 3 parameters *Hue*, *Saturation*, and *Value* you can use the following explanation:

Hue (H)
the colors of the rainbow
(on the circumference of the circle, in degrees)

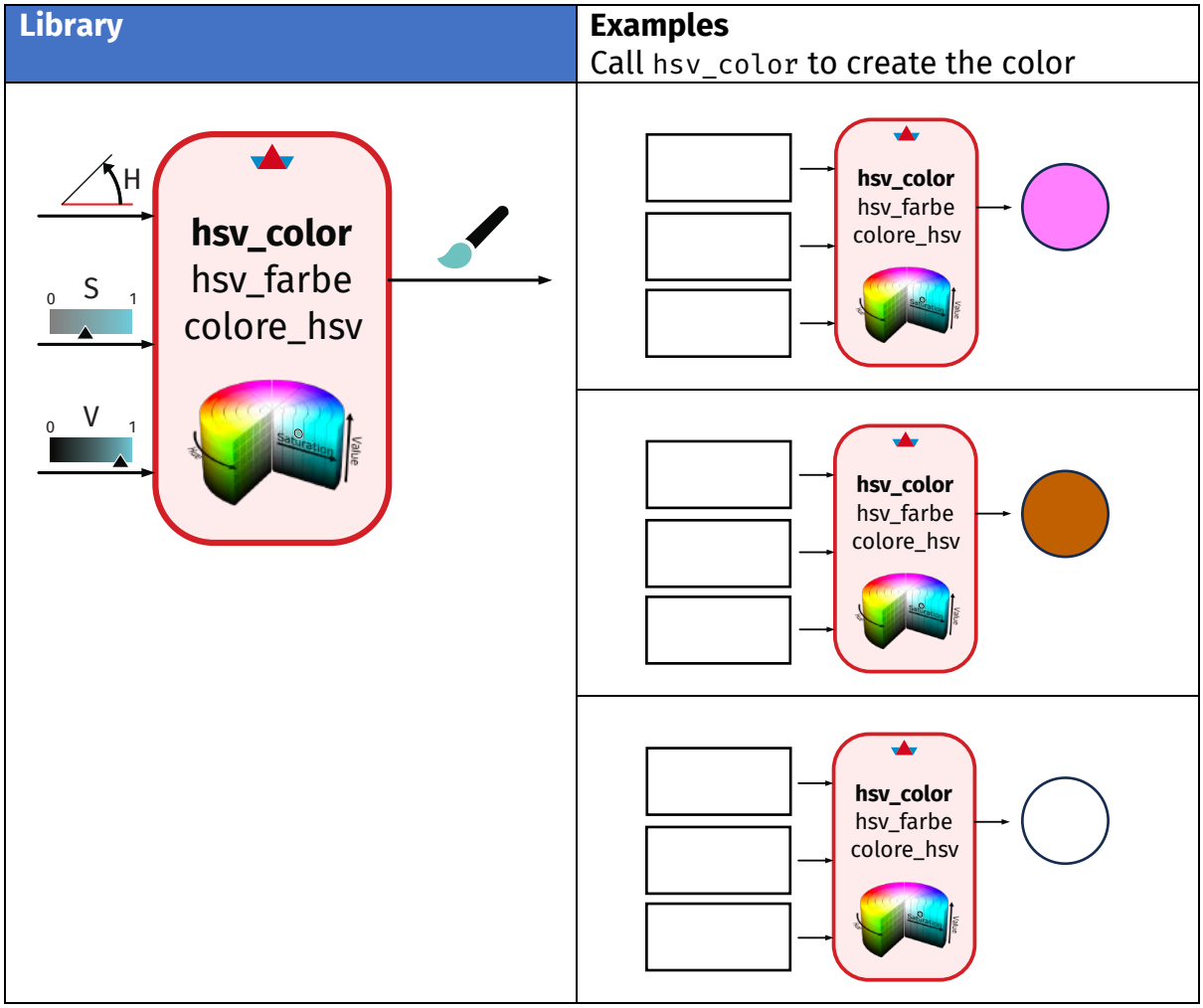


Saturation (S) and Value (V)
Table with S and V, for hue 0° / red:

Saturation \ Value	0.0 0%	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0 100%
1.0 100%	S: 1.0 V: 0.0	S: 1.0 V: 0.1	S: 1.0 V: 0.2	S: 1.0 V: 0.3	S: 1.0 V: 0.4	S: 1.0 V: 0.5	S: 1.0 V: 0.6	S: 1.0 V: 0.7	S: 1.0 V: 0.8	S: 1.0 V: 0.9	S: 1.0 V: 1.0
0.9	S: 0.9 V: 0.0	S: 0.9 V: 0.1	S: 0.9 V: 0.2	S: 0.9 V: 0.3	S: 0.9 V: 0.4	S: 0.9 V: 0.5	S: 0.9 V: 0.6	S: 0.9 V: 0.7	S: 0.9 V: 0.8	S: 0.9 V: 0.9	S: 0.9 V: 1.0
0.8	S: 0.8 V: 0.0	S: 0.8 V: 0.1	S: 0.8 V: 0.2	S: 0.8 V: 0.3	S: 0.8 V: 0.4	S: 0.8 V: 0.5	S: 0.8 V: 0.6	S: 0.8 V: 0.7	S: 0.8 V: 0.8	S: 0.8 V: 0.9	S: 0.8 V: 1.0
0.7	S: 0.7 V: 0.0	S: 0.7 V: 0.1	S: 0.7 V: 0.2	S: 0.7 V: 0.3	S: 0.7 V: 0.4	S: 0.7 V: 0.5	S: 0.7 V: 0.6	S: 0.7 V: 0.7	S: 0.7 V: 0.8	S: 0.7 V: 0.9	S: 0.7 V: 1.0
0.6	S: 0.6 V: 0.0	S: 0.6 V: 0.1	S: 0.6 V: 0.2	S: 0.6 V: 0.3	S: 0.6 V: 0.4	S: 0.6 V: 0.5	S: 0.6 V: 0.6	S: 0.6 V: 0.7	S: 0.6 V: 0.8	S: 0.6 V: 0.9	S: 0.6 V: 1.0
0.5	S: 0.5 V: 0.0	S: 0.5 V: 0.1	S: 0.5 V: 0.2	S: 0.5 V: 0.3	S: 0.5 V: 0.4	S: 0.5 V: 0.5	S: 0.5 V: 0.6	S: 0.5 V: 0.7	S: 0.5 V: 0.8	S: 0.5 V: 0.9	S: 0.5 V: 1.0
0.4	S: 0.4 V: 0.0	S: 0.4 V: 0.1	S: 0.4 V: 0.2	S: 0.4 V: 0.3	S: 0.4 V: 0.4	S: 0.4 V: 0.5	S: 0.4 V: 0.6	S: 0.4 V: 0.7	S: 0.4 V: 0.8	S: 0.4 V: 0.9	S: 0.4 V: 1.0
0.3	S: 0.3 V: 0.0	S: 0.3 V: 0.1	S: 0.3 V: 0.2	S: 0.3 V: 0.3	S: 0.3 V: 0.4	S: 0.3 V: 0.5	S: 0.3 V: 0.6	S: 0.3 V: 0.7	S: 0.3 V: 0.8	S: 0.3 V: 0.9	S: 0.3 V: 1.0
0.2	S: 0.2 V: 0.0	S: 0.2 V: 0.1	S: 0.2 V: 0.2	S: 0.2 V: 0.3	S: 0.2 V: 0.4	S: 0.2 V: 0.5	S: 0.2 V: 0.6	S: 0.2 V: 0.7	S: 0.2 V: 0.8	S: 0.2 V: 0.9	S: 0.2 V: 1.0
0.1	S: 0.1 V: 0.0	S: 0.1 V: 0.1	S: 0.1 V: 0.2	S: 0.1 V: 0.3	S: 0.1 V: 0.4	S: 0.1 V: 0.5	S: 0.1 V: 0.6	S: 0.1 V: 0.7	S: 0.1 V: 0.8	S: 0.1 V: 0.9	S: 0.1 V: 1.0
0.0 0%	S: 0.0 V: 0.0	S: 0.0 V: 0.1	S: 0.0 V: 0.2	S: 0.0 V: 0.3	S: 0.0 V: 0.4	S: 0.0 V: 0.5	S: 0.0 V: 0.6	S: 0.0 V: 0.7	S: 0.0 V: 0.8	S: 0.0 V: 0.9	S: 0.0 V: 1.0

0.0 0% ***Value*** **1.0 100%**

Specifying Colors With the HSV Model in PyTamaro



Check whether you guessed the colors in the "*Il modello HSV*" activity on the PyTamaro web site!

You can imagine the HSV color model as a cylinder: