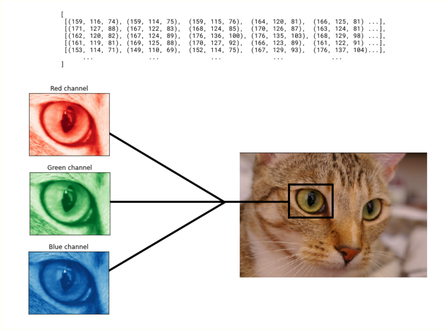
1.8 Application: Representing Colour

The data types we've studied so far are not the only kinds of data that we encounter in the real world, but they do form a basis for representing all kinds of more complex data. We'll study how to represent more complex forms of data later in this course, but here's one teaser: representing image and colour data.



Images can be represented as data in a computer program as a list, where each element corresponds to a very tiny dot on your screen, called a *pixel*. For each pixel, we need a way to represent what colour should be shown. But if we accomplish that, somehow, our computers are able to take these lists and translate them into a sequence of visible lights and if these lights are arranged in a particular way, well, a cat appears!

The physics behind how we perceive colour is incredibly interesting, but also complex. Humans have developed a broad range of names of colours to identify categories like "red" in everyday language.[1] Yet these categories can be fairly broad and imprecise; useful for everyday communication, but not for computer graphics and design. So in this section, we'll learn about how computers represent colour data.

[1] Although the names we use for colours vary widely from language to language!

Mathematics can help us represent colours by a combination of numbers; the rules for how numbers map to colours is called a *colour model*. Many colour models exist, but one of the most common is the *RGB colour model*. At some point in your youth, you may have discovered that mixing two colours together (i.e., with paint, crayons, etc.) produces a different colour. The RGB colour model is based on the same idea: each colour is represented by three numbers, one for the "amount" of red, green, and blue to be mixed together.

A common form of the RGB colour model in a computer is called the **RGB24** colour model, and allows for each of the red, green, and blue amounts to be a number between 0 and 255, inclusive.[2] Formally, we can define the set $S = \{0, 1, \ldots, 255\}$ and $\mathcal{C}$ to be the set of all possible colours in the universe. Then the RGB colour model is a function $RGB_{24} : S \times S \times S \to \mathcal{C}$ that takes in red, green, and blue values from S and returns a colour. This RGB_{24} function is *one-to-one*, as every combination of (red, green, blue) values produces a different colour.

[2] Though RGB24 is quite common, software like Photoshop allow for a larger range of numbers, enabling more granularity it their colour representations. You can look up the term *deep colour* to find out more about more sophisticated colour models

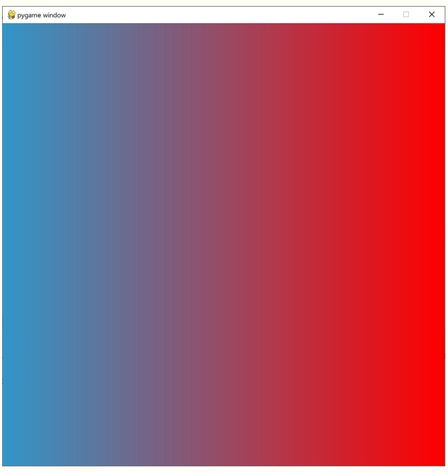
RGB Value	Colour
(0, 0, 0)	
(255, 0, 0)	
(0, 255, 0)	
(0, 0, 255)	
(181, 57, 173)	
(255, 255, 255)	

Colours in Python

The RGB24 colour model translates naturally to Python: we represent a colour value as a list of three integers between 0 and 255 inclusive. For example, we can use `[0, 0, 0]` to represent a pure black and `[181, 57, 173]` to represent a shade of purple. Of course, just representing these values as lists doesn't automatically make them colours:

```
>>> [181, 57, 173]  # This list literal evaluates to... itself
[181, 57, 173]
```

But as you'll see in your first tutorial this year, we can pass these lists to operations that expect colour values, and get remarkable results.



Pygame demo of colour gradient. You will generate this image in Tutorial 1 this year!

References

- Hexadecimal Colors
- Color Theory