

On Line 24 we manipulate the top-left pixel in the image, which is located at coordinate (0,0), and set it to have a value of (0, 0, 255). If we were reading this pixel value in BGR format, we would have a value of 2 for red, 0 for green, and 255 for blue, thus making it a pure blue color.

However, as I mentioned above, we need to take special care when working with OpenCV. Our pixels are actually stored in BGR format, not RGB format.

We actually read this pixel as 255 for red, 0 for green, and 0 for blue, making it a red color, not a blue color.

After setting the top-left pixel to have a red color on Line 24, we then grab the pixel value and print it back to console on Lines 25 and 26, just to demonstrate that we have indeed successfully changed the color of the pixel.

Accessing and setting a single pixel value is simple enough, but what if we wanted to use NumPy's array slicing capabilities to access larger rectangular portions of the image? The code below demonstrates how we can do this.

*Listing 2.3: getting and setting*

```
17 image = image[0:100, 0:100]
18 cv.imshow('image', image)
19
20 image[0:100, 0:100] = [0, 0, 0]
21
22 cv.imshow('image', image)
23 cv.waitKey(0)
```

On Line 17 we grab a 100 × 100 pixel region of the image. In fact, this is the top-left corner of the image! In order to grab chunks of an image, NumPy expects we provide four