

## 01 Pixel coordinates

|   | 0   | 1   | 2   | 3   | 4   | 5   |
|---|-----|-----|-----|-----|-----|-----|
| 0 | 137 | 145 | 210 | 110 | 196 | 24  |
| 1 | 135 | 89  | 3   | 159 | 118 | 5   |
| 2 | 76  | 223 | 18  | 218 | 226 | 11  |
| 3 | 34  | 205 | 11  | 47  | 206 | 178 |
| 4 | 133 | 39  | 55  | 177 | 66  | 245 |
| 5 | 111 | 252 | 83  | 169 | 47  | 41  |

Dashed teal: Q4 block. Dashed orange: Q5 block.

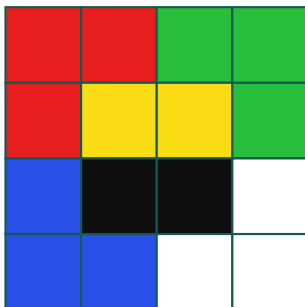
- 1 `pixels[2, 3] = 218`
- 2 Brightest is **252**, at **[5, 1]** (row 5, column 1)
- 3 `pixels[0, 5]` (gives 24)
- 4 Rows 0–1, columns 0–2 (stops before 2 and before 3):

|     |     |     |
|-----|-----|-----|
| 137 | 145 | 210 |
| 135 | 89  | 3   |

- 5 `pixels[4:6, 4:6]` (also accept `pixels[-2:, -2:]`, `pixels[4:, 4:]`). It gives `[[66, 245], [47, 41]]`

## 04 Channel detective

**Part 1** (picture, then red, green, blue):



|     |     |     |     |
|-----|-----|-----|-----|
| 255 | 255 | 0   | 0   |
| 255 | 255 | 255 | 0   |
| 0   | 0   | 0   | 255 |
| 0   | 0   | 255 | 255 |

|   |     |     |     |
|---|-----|-----|-----|
| 0 | 0   | 255 | 255 |
| 0 | 255 | 255 | 255 |
| 0 | 0   | 0   | 255 |
| 0 | 0   | 255 | 255 |

|     |     |     |     |
|-----|-----|-----|-----|
| 0   | 0   | 0   | 0   |
| 0   | 0   | 0   | 0   |
| 255 | 0   | 0   | 255 |
| 255 | 255 | 255 | 255 |

Picture, by row: red red green green; red yellow yellow green; blue black black white; blue blue white white.

**Part 2** (the 3 × 3 picture):

|        |        |        |
|--------|--------|--------|
| green  | yellow | green  |
| yellow | black  | yellow |
| green  | yellow | green  |

Yellow cell: red and green on (255, 255, 0), blue off. Green cells have only the green channel on; black cells have all three off.

## 05 Predict and check: model answers

Every value below was produced by running the operation on `cow-demo.jpg` ( $320 \times 320$ ). The top-left  $100 \times 100$  block is the red-painted wall; its average colour is (125, 60, 48).

|                                                 |                                                                                                                                                                                        |
|-------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>no_red[:, :, 0] = 0</code>                | Every red number becomes 0. The red wall goes dark teal: its average changes from (125, 60, 48) to (0, 60, 48). Reds vanish, so the picture looks blue-green.                          |
| <code>only_blue (red and green set to 0)</code> | Only the blue numbers survive, so the picture is shades of blue. The wall is almost black, (0, 0, 48), because it had little blue.                                                     |
| <code>negative = 255 - pixels</code>            | Every number is flipped, so dark becomes light and colours become their opposites. Pixel [0, 0] goes from (16, 0, 0) to (239, 255, 255); the wall becomes a pale blue (130, 195, 207). |
| <code>pixels[:, ::-1]</code>                    | Mirror image, left and right swapped. The shape stays (320, 320, 3).                                                                                                                   |
| <code>(grey &lt; 50).sum()</code>               | Counts the pixels darker than 50: <b>28501</b> out of 102400 (about 28%), mostly the black cow.                                                                                        |
| <code>corner = pixels[0:100, 0:100]</code>      | A $100 \times 100$ square from the top-left, shape (100, 100, 3): part of the red wall.                                                                                                |

**Reflection 1.** The wall is painted red, so its red number is large (about 125) and its green and blue numbers are small (60 and 48). Large number means bright in the red channel.

**Reflection 2.** White is (255, 255, 255). Without red it is (0, 255, 255), which is cyan (blue-green). Extension: black is unchanged, since 0 stays 0.

**Task 7 values for reference:** average brightness 81.45; pixels brighter than 100: 34315; fraction 0.335.

## Solution image\_explore\_solution.py

```
1 # image_explore_solution.py (INSTRUCTOR COPY: do not hand out before students have tried)
2
3 import numpy as np
4 from PIL import Image
5 import matplotlib.pyplot as plt
6
7 def show(pixels):
8     plt.imshow(pixels)
9     plt.axis("off")
10    plt.show()
11
12 def show_channel(channel, colour):
13     plt.imshow(channel, cmap=colour, vmin=0, vmax=255)
14     plt.axis("off")
15     plt.colorbar()
16     plt.show()
17
18 def show_colour(red, green, blue):
19     # draws one big square of the colour you give it (each number is 0 to 255)
20     plt.imshow([[red, green, blue]])
21     plt.axis("off")
22     plt.show()
23
24 img = Image.open("cow-demo.jpg")
25 pixels = np.array(img)
26 show(pixels)
27 print(pixels.shape)
28
29 print(pixels[0, 0])
30 print(pixels[100, 200])
31
32 corner = pixels[0:100, 0:100]
33 show(corner)
34 other_corner = pixels[220:320, 220:320]
35 show(other_corner)
36
37 show_colour(230, 30, 30)
38 show_colour(255, 255, 0)
39 show_colour(128, 128, 128)
40
41 red = pixels[:, :, 0]
42 green = pixels[:, :, 1]
43 blue = pixels[:, :, 2]
44 show_channel(red, "Reds")
45 show_channel(green, "Greens")
46 show_channel(blue, "Blues")
47
48 no_red = pixels.copy()
49 no_red[:, :, 0] = 0
50 show(no_red)
51 no_green = pixels.copy()
52 no_green[:, :, 1] = 0
53 show(no_green)
54 no_blue = pixels.copy()
55 no_blue[:, :, 2] = 0
56 show(no_blue)
57 only_blue = pixels.copy()
58 only_blue[:, :, 0] = 0
59 only_blue[:, :, 1] = 0
60 show(only_blue)
61
62 grey = np.array(img.convert("L"))
63 show_channel(grey, "gray")
64 negative = 255 - pixels
65 show(negative)
66
67 print(grey.mean())
68 print((grey < 50).sum())
69 bright = (grey > 100).sum()
70 print(bright)
71 print(bright / grey.size)
72
73 plt.hist(grey.flatten(), bins=16)
74 plt.show()
75 plt.hist(red.flatten(), bins=16)
76 plt.show()
```