

# Pixel coordinates

This grid is a tiny picture stored as `pixels` . Each number is a brightness.

Name: \_\_\_\_\_

Date: \_\_\_\_\_

|   | 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  |

1 What is `pixels[2, 3]` ?

Answer

2 Which pixel is the brightest? Give its `[row, column]` .

Answer

3 Write the code to get row 0, column 5.

Code

4 Slicing. What does `pixels[0:2, 0:3]` give? Shade that block on the grid, then write its numbers here.

|  |  |  |
|--|--|--|
|  |  |  |
|  |  |  |

**Remember:** `0:2` starts at 0 and stops *before* 2, so it gives rows 0 and 1 only.

5 Write the slice that would give the bottom-right  $2 \times 2$  block.

Code

# What Python sees

A picture is a grid of numbers. Each pixel is R G B (red, green, blue).

Name: \_\_\_\_\_

Date: \_\_\_\_\_

## Part 1 A tiny $4 \times 4$ patch of the cow

|                                                                                             |                                                                                             |                                                                                                |                                                                                               |
|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
|  113 73 63 |  123 80 71 |  136 92 81  |  141 95 80 |
|  125 83 71 |  139 95 84 |  150 104 91 |  147 99 85 |
|  132 88 77 |  140 93 83 |  145 97 85  |  142 93 78 |
|  125 81 68 |  130 82 70 |  138 89 75  |  144 92 78 |

## Part 2 Questions

1 How many numbers are in this patch, in total?

2 Which pixel has the biggest *red* number? (row, column)

3 Add up all 16 red numbers. Use the box for your working.

Working

4 Are neighbouring pixels similar, or very different? Say how you can tell.

5 Predict! The whole picture is  $320 \times 320$  pixels. How many numbers does Python store?

# Mix a colour

Every colour is a mix of red, green and blue. Each number goes from 0 to 255.

Name: \_\_\_\_\_

Date: \_\_\_\_\_

## Part 1 Predict the colour

Write your guess in each box.

**Word bank:** red green blue yellow orange pink purple cyan magenta black white grey

(255, 0, 0)

Checked ☐

(0, 0, 255)

Checked ☐

(255, 255, 0)

Checked ☐

(0, 0, 0)

Checked ☐

(255, 0, 255)

Checked ☐

(128, 128, 128)

Checked ☐

## Part 2 Write the RGB numbers

Fill in three numbers for each colour.

**Bright green**

R

G

B

☐

**White**

R

G

B

☐

**Black**

R

G

B

☐

**Orange (guess!)**

R

G

B

☐

## Part 3 Check it on the computer

Open `image_explore.py` and find the **TRY IT: mix a colour** block. Change the three numbers in `show_colour(230, 30, 30)` to test each colour above, then tick the boxes you checked.

```
show_colour(255, 0, 0)
```

What surprised you?

# Channel detective

A picture is three grids of numbers stacked up: red, green and blue.

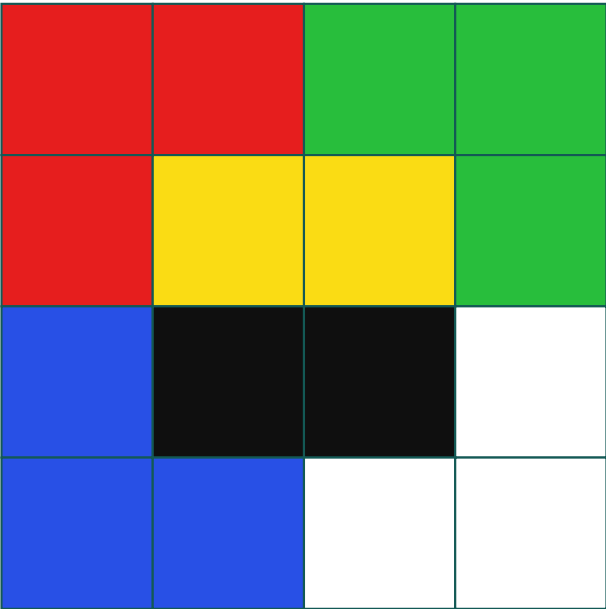
Name: \_\_\_\_\_

Date: \_\_\_\_\_

## Part 1 Split the picture into its channels

For each cell, write **0** (off) or **255** (full on) in each channel grid.

The picture (4 × 4)



Red channel

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Green channel

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Blue channel

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Colour mixing:** red is (255, 0, 0) . Yellow is red + green. White is all three on. Black is all three off.

## Part 2 Now go backwards

Here are the three channels of a different  $3 \times 3$  picture. Write the colour of each cell in the grid below.

**Red channel**

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

**Green channel**

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

**Blue channel**

|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |

**The picture (write a colour name in each cell)**

|  |  |  |
|--|--|--|
|  |  |  |
|  |  |  |
|  |  |  |

**Word bank**

red green blue  
yellow white black

**? Which channels are on for a yellow cell?**

---

---

# Predict and check

Guess first, then run the line in Spyder and write what really happened.

Name: \_\_\_\_\_

Date: \_\_\_\_\_

**Slicing backwards:** `::-1` means “go backwards”, so `[:, ::-1]` reads the columns in reverse order.

| code                                                                           | what I think it does                                       | what actually happened             | ✓ |
|--------------------------------------------------------------------------------|------------------------------------------------------------|------------------------------------|---|
| <code>print(pixels.shape)</code>                                               | Example: three numbers, rows then columns then 3 channels. | Example: it printed (320, 320, 3). | ✓ |
| <code>no_red[:, :, 0] = 0</code>                                               |                                                            |                                    |   |
| <code>only_blue[:, :, 0] = 0</code><br><br><code>only_blue[:, :, 1] = 0</code> |                                                            |                                    |   |
| <code>negative = 255 - pixels</code>                                           |                                                            |                                    |   |
| <code>pixels[:, ::-1]</code>                                                   |                                                            |                                    |   |
| <code>(grey &lt; 50).sum()</code>                                              |                                                            |                                    |   |
| <code>corner = pixels[0:100, 0:100]</code>                                     |                                                            |                                    |   |

## Part 2 Think about it

1 Why does the wall look bright in the red channel?

2 What happens to white if you take away the red channel?

# The explore code

Your printed copy of `image_explore.py`. Type it or open the file.

Name: \_\_\_\_\_

Date: \_\_\_\_\_

**Set up:** keep `image_explore.py` and `cow-demo.jpg` in the *same folder*. Open the file in Spyder and press **F5** to run everything, or select a block and press **F9**.

```
8 import numpy as np
9 from PIL import Image
10 import matplotlib.pyplot as plt
11
12 # ----- GIVEN: you do not need to change these three functions -----
13 def show(pixels):
14     # draws a colour picture from a grid of numbers
15     plt.imshow(pixels)
16     plt.axis("off")
17     plt.show()
18
19 def show_channel(channel, colour):
20     # draws ONE channel, using the colour scale you give: "Reds", "Greens", "Blues" or
    "gray"
21     plt.imshow(channel, cmap=colour, vmin=0, vmax=255)
22     plt.axis("off")
23     plt.colorbar()
24     plt.show()
25
26 def show_colour(red, green, blue):
27     # draws one big square of the colour you give it (each number is 0 to 255)
28     plt.imshow([[red, green, blue]])
29     plt.axis("off")
30     plt.show()
31
32 # ----- TASK 1: open a picture -----
33 img = Image.open("cow-demo.jpg")
34 pixels = np.array(img)
35 show(pixels)
36 print(pixels.shape)          # predict first: how many rows? how many columns?
37
38 # ----- TASK 2: one pixel -----
39 print(pixels[0, 0])          # the pixel in row 0, column 0 (top left)
40 # TODO: print the pixel in row 100, column 200
41 #   hint: pixels[row, column]
42
43 # ----- TASK 3: crop a corner -----
44 corner = pixels[0:100, 0:100]    # rows 0 to 99, columns 0 to 99
45 show(corner)
46 # TODO: crop a different 100 x 100 square, then show it
47 #   hint: change the numbers, keep the colon and the comma
48
49 # ----- TRY IT: mix a colour (after the RGB slides) -----
50 show_colour(230, 30, 30)        # red, green, blue. Try changing these three numbers!
51 # TODO: make yellow, then make a grey
```

```

61 # ----- TASK 4: split the channels -----
62 red = pixels[:, :, 0]
63 show_channel(red, "Reds")
64 # TODO: do the same for green (channel 1) and blue (channel 2)
65 # hint: show_channel(green, "Greens")
66 # Which channel is brightest on the wall? Which is brightest on the cow?
67
68 # ----- TASK 5: switch a channel off -----
69 no_red = pixels.copy() # .copy() makes a second picture to change
70 no_red[:, :, 0] = 0 # set every red value to 0
71 show(no_red)
72 # TODO: make a picture with NO green, and one with NO blue
73 # TODO: make a picture that is ONLY blue (red and green both 0)
74
75 # ----- TASK 6: grey and negative -----
76 grey = np.array(img.convert("L")) # "L" means one brightness number per pixel
77 show_channel(grey, "gray")
78 negative = 255 - pixels # flips every number: 0 becomes 255, 255 becomes 0
79 show(negative)
80
81 # ----- TASK 7: count and average -----
82 print(grey.mean()) # the average brightness, 0 to 255
83 print((grey < 50).sum()) # how many pixels are darker than 50?
84 # TODO: how many pixels are brighter than 100?
85 # hint: change the < and the number
86 # TODO: what fraction of the whole picture is that?
87 # hint: divide by grey.size
88
89 # ----- TASK 8: a histogram -----
90 plt.hist(grey.flatten(), bins=16)
91 plt.show()
92 # TODO: draw a histogram of just the red channel: plt.hist(red.flatten(), bins=16)
93
94 # ----- STRETCH -----
95 # Mirror the picture: show(pixels[:, ::-1]) # ::-1 means "go backwards"
96 # Upside down: show(pixels[::-1])
97 # Brighten it: add 60 to every number. Careful, what happens above 255?

```

### Tick when done:

- ☐ Task 1   ☐ Task 2   ☐ Task 3   ☐ Try it: mix a colour  
☐ Task 4   ☐ Task 5   ☐ Task 6   ☐ Task 7   ☐ Task 8

# Be the computer

Add up the red numbers, take the remainder after dividing by 26, find the letter.

Name: \_\_\_\_\_

Date: \_\_\_\_\_

|          |          |          |          |          |          |          |          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>A</b> | <b>B</b> | <b>C</b> | <b>D</b> | <b>E</b> | <b>F</b> | <b>G</b> | <b>H</b> | <b>I</b> | <b>J</b> | <b>K</b> | <b>L</b> | <b>M</b> |
| 0        | 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        | 10       | 11       | 12       |
| <b>N</b> | <b>O</b> | <b>P</b> | <b>Q</b> | <b>R</b> | <b>S</b> | <b>T</b> | <b>U</b> | <b>V</b> | <b>W</b> | <b>X</b> | <b>Y</b> | <b>Z</b> |
| 13       | 14       | 15       | 16       | 17       | 18       | 19       | 20       | 21       | 22       | 23       | 24       | 25       |

% 26 by hand: keep taking 26 off until you cannot. **26 times table:**

|       |      |      |      |      |      |      |      |      |      |      |
|-------|------|------|------|------|------|------|------|------|------|------|
| 1–10  | 26   | 52   | 78   | 104  | 130  | 156  | 182  | 208  | 234  | 260  |
| 11–20 | 286  | 312  | 338  | 364  | 390  | 416  | 442  | 468  | 494  | 520  |
| 21–30 | 546  | 572  | 598  | 624  | 650  | 676  | 702  | 728  | 754  | 780  |
| 31–40 | 806  | 832  | 858  | 884  | 910  | 936  | 962  | 988  | 1014 | 1040 |
| 41–50 | 1066 | 1092 | 1118 | 1144 | 1170 | 1196 | 1222 | 1248 | 1274 | 1300 |

## Part 1 Decode three tiny images

**Image 1** (worked example)

|     |     |     |
|-----|-----|-----|
| 60  | 58  | 146 |
| 179 | 176 | 119 |
| 164 | 185 | 85  |

Add up the 9 red numbers (working)

$$60 + 58 + 146 + 179 + 176 + 119 + 164 + 185 + 85 = 1172$$

Total

1172

Total % 26

2

$$1172 - 26 \times 45 = 2$$

Letter

C

**Image 2** (your turn)

|     |     |     |
|-----|-----|-----|
| 48  | 212 | 202 |
| 35  | 126 | 187 |
| 177 | 61  | 162 |

Add up the 9 red numbers (working)

Total

Total % 26

Letter

**Image 3** (your turn)

|     |     |     |
|-----|-----|-----|
| 44  | 180 | 113 |
| 178 | 98  | 155 |
| 165 | 160 | 47  |

Add up the 9 red numbers (working)

Total

Total % 26

Letter

## Part 2 How long?

1 One tiny image took me  minutes.

2 The real challenge has 37 images, each  $640 \times 640 = 409,600$  pixels. Roughly how long by hand?

**Hint:** a tiny image has 9 numbers.  $409,600 \div 9$  is about 45,000 tiny images *per picture*. Then multiply by 37!

---

---

---

---

3 What should we get the computer to do?

---

---

---

---

# Plan before you type

Write it in plain English first. No code yet!

Name: \_\_\_\_\_

Date: \_\_\_\_\_

## Part 1 One letter from ONE picture

Write the steps to turn one picture into one letter, one per line.

**Word bank:** open • red channel • add up • % 26 • look up • letter

|   |       |
|---|-------|
| 1 | _____ |
| 2 | _____ |
| 3 | _____ |
| 4 | _____ |
| 5 | _____ |
| 6 | _____ |

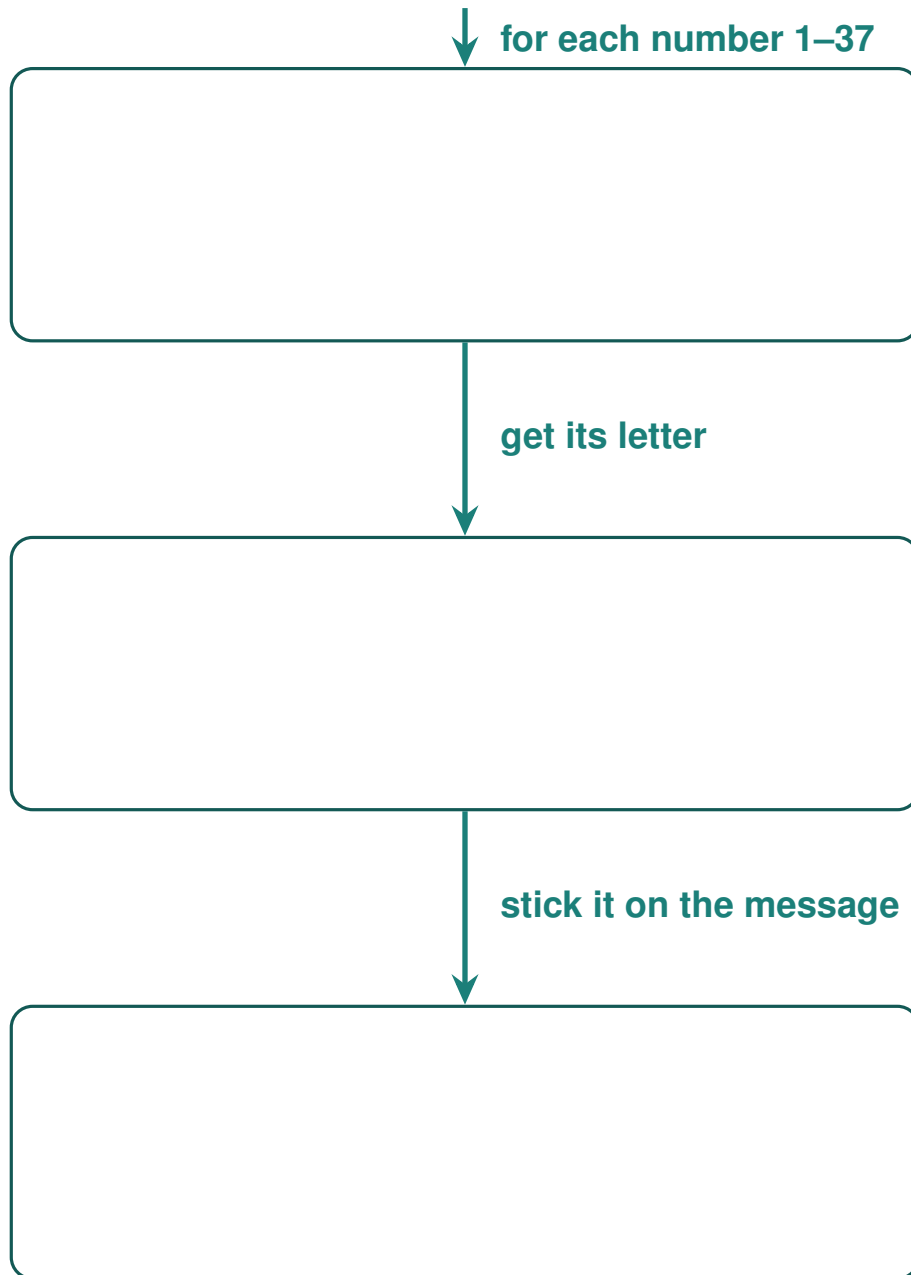
## Part 2 All 37 pictures

How do we get the whole message from all 37 pictures?

|   |       |
|---|-------|
| 1 | _____ |
| 2 | _____ |
| 3 | _____ |
| 4 | _____ |

## Part 3 Boxes and arrows

Write what happens in each box.



# Fill in the functions

Finish the code, run it, and check it.

Name: \_\_\_\_\_

Date: \_\_\_\_\_

## Part 1 Add up the red

```
def total_red(path):  
    img = Image.open(path)  
    pixels = np.array(img)
```

### Write your code here

1. get the red channel: `pixels[:, :, ___]`
2. add up every value in it: `.sum()`

```
return _____ # what do we give back?
```

`total_red("cow-demo.jpg")` expected: 10218857

What did YOU get? \_\_\_\_\_ ☐ it matches

## Part 2 One picture, one letter

```
def letter_for(path):  
    total = _____(path) # the red total (your function!)
```

### Write your code here

1. remainder: `total __ 26`
2. pick that letter from `ALPHABET`

```
return _____ # the letter
```

`letter_for("cow-demo.jpg")` expected: Z

What did YOU get? \_\_\_\_\_ ☐ it matches

## Part 3 The whole message

```
def decode_message():  
    message = ""          # starts empty  
    for _____ in range(____, ____):
```

### Write your code here

1. find the picture: `image_path(...)`
2. get its letter (your function!)
3. stick it on the end: `message += ...`

```
return _____
```

### Debugging checklist

- ☐ Did I `return` at the end?
- ☐ Is the loop body indented?
- ☐ Did I use `range(1, 38)` ? Why 38?
- ☐ Is my file in the same folder as `cow_message` ?
- ☐ Did I run the file with F5?
- ☐ Did I use `% 26` , not `/ 26` ?

| Function       | test worked? | notes |
|----------------|--------------|-------|
| total_red      |              |       |
| letter_for     |              |       |
| decode_message |              |       |

### Did the message make sense? How do you know it is right?

---

---

**Stretch: other colours!** Add up the GREEN (channel 1) or BLUE (channel 2) instead of red. Copy your functions, change the `0` , and see what comes out.

# The starter code

Your printed copy of `image_starter.py`.

Name: \_\_\_\_\_

Date: \_\_\_\_\_

**Set up:** keep `image_starter.py`, `cow-demo.jpg` and the `cow_message` folder in the *same folder*.  
Open `image_starter.py` in Spyder and press **F5**.

```
7 import numpy as np
8 from PIL import Image
9 import matplotlib.pyplot as plt
10 ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"      # A is 0, B is 1, ... Z is 25
11
12
13 # ----- GIVEN: you do not need to change these two functions -----
14 def image_path(number):
15     # image_path(1) gives "cow_message/01.png", image_path(12) gives
16     # "cow_message/12.png"
17     return "cow_message/" + str(number).zfill(2) + ".png"
18
19 def show_colour(red, green, blue):
20     # draws one big square of the colour you give it (each number is 0 to 255)
21     plt.imshow([[red, green, blue]])
22     plt.axis("off")
23     plt.show()
24
25
26 # ----- STEP 1: look at a picture (run this and read the output) -----
27 img = Image.open("cow-demo.jpg")
28 pixels = np.array(img)
29 print("size:", img.size)
30 print("shape:", pixels.shape)
31 print("first pixel:", pixels[0, 0])
32 show_colour(230, 30, 30)      # try changing these three numbers!
```

```
35 # ----- STEP 2 (TODO): add up all the red in ONE picture -----
36 def total_red(path):
37     img = Image.open(path)
38     pixels = np.array(img)
39
40     # TODO: get just the red channel
41     # hint: pixels[:, :, 0] (every row, every column, channel 0 = red)
42     # TODO: add up every red value in it
43     # hint: .sum()
44     # TODO: return that total
45
46     return 0
47
48
49 # Test it. It should print 10218857
50 print(total_red("cow-demo.jpg"))
```

```

53 # ----- STEP 3 (TODO): turn a picture into ONE letter -----
54 def letter_for(path):
55     # TODO: 1. get the red total for this picture (you wrote a function for this!)
56     #         2. take it % 26    (the remainder, a number from 0 to 25)
57     #         3. use that number to pick a letter from ALPHABET
58     #             hint: ALPHABET[2] is "C"
59     #         4. return the letter
60     return ""
61
62
63 # Test it. It should print Z
64 print(letter_for("cow-demo.jpg"))

```

```

67 # ----- STEP 4 (TODO): decode the WHOLE message -----
68 def decode_message():
69     message = ""           # the message we are building. It starts empty.
70
71     # TODO: for each number from 1 to 37 (hint: for number in range(1, 38))
72     #     1. find the picture:  image_path(number)
73     #     2. work out its letter (you already wrote a function for this!)
74     #     3. stick the letter on the end of message:  message += letter
75
76     return message
77
78
79 # Run it. Does the message make sense? If not, check your steps 2 and 3.
80 # print(decode_message())

```

```

83 # ----- STRETCH: a different channel -----
84 # What if you add up the GREEN values (channel 1) or BLUE values (channel 2) instead?
85 # Copy your functions, change the 0, and see what message you get.

```

Tick when done:   ☐ colour square shows   ☐ total\_red prints 10218857   ☐ letter\_for prints Z  
☐ decode\_message works

# Message tracker

Write each image's letter in its box. Then copy the letters into the message.

Name: \_\_\_\_\_

Date: \_\_\_\_\_

Part 1    Image by image

|    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |    |    |

Part 2    The message says:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Does it make sense?    Yes ☐    No ☐

What would change if we used GREEN instead of RED?