1. The SimpleImage toolbox

a. Fill in the empty boxes. Use the word bank. Keep this table for the lab. (A canvas is just a variable, so we can name it `canvas`, `image`, `img`, or anything we want.)

```
from simpleimage import SimpleImage        canvas.height          canvas.show()
canvas = SimpleImage.blank(width, height)    canvas.get_rgb(x, y)
canvas.show(resize_width=200)
```

What I want to do	Code
Get the `SimpleImage` tool out of the file `simpleimage.py`	
Make a new white canvas that is `width` wide and `height` tall	
How many columns?	`canvas.width`
How many rows?	
Read the color of the pixel at column `x`, row `y`	
Paint the pixel at column `x`, row `y` with the color `(r, g, b)`	`canvas.set_rgb(x, y, r, g, b)`
See the picture	
See a very small picture bigger	

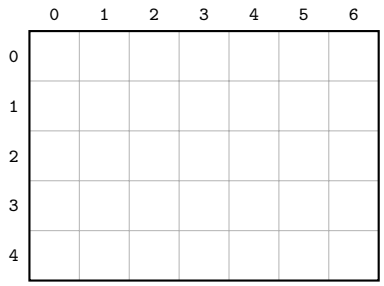
Solution: Row 1: `from simpleimage import SimpleImage`
Row 2: `canvas = SimpleImage.blank(width, height)`
Row 4: `canvas.height`
Row 5: `canvas.get_rgb(x, y)`
Row 7: `canvas.show()`
Row 8: `canvas.show(resize_width=200)`

Note for the TA: columns → `width` and rows → `height` is the pairing students most often get backwards.

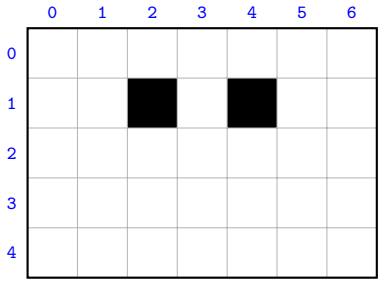
2. Painting pixels

```
image = SimpleImage.blank(7, 5)  # blank pixels are white (255,255,255)
image.set_rgb(2, 1, 0, 0, 0)   # color (0,0,0) is black
image.set_rgb(4, 1, 0, 0, 0)
```

a. What does the image look like after the above code? Shade in the correct pixels.



Solution: Shade `(2, 1)` and `(4, 1)`. Watch out: `(2, 1)` is black but `(1, 2)` is still white — they are different pixels, because `x` comes first.



3. Painting with loops

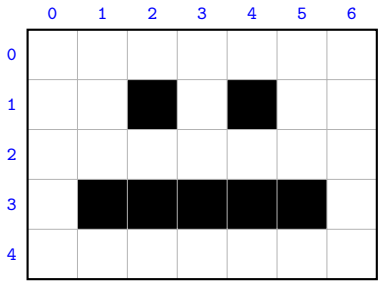
This code runs *after* the code in question 2, on the same `image`.

```
for x in range(image.width):
    image.set_rgb(x, 3, 0, 0, 0)

image.set_rgb(0, 3, 255, 255, 255)
image.set_rgb(6, 3, 255, 255, 255)
```

a. Use a pencil to shade the new black pixels on the same grid in question **2**. What do you see?

Solution:



It's a face!

A pixel keeps only the *last* color it is given. The loop paints all of row 3 black, then the two white lines run afterwards, so white wins on those two pixels.

b. To paint *every* pixel we need a loop inside a loop. Fill in the blanks for this function that paints the given color `(r, g, b)` on every pixel on the given canvas of *any* size.

```
def fill_all(canvas, r, g, b):
    # canvas is a SimpleImage
    for x in range(_________________________):
        for y in range(_________________________):
            canvas._____________________________________
    return canvas
```

Solution:

```
def fill_all(canvas, r, g, b):
    for x in range(canvas.width):
        for y in range(canvas.height):
            canvas.set_rgb(x, y, r, g, b)
    return canvas
```

The outer loop picks a column, and the inner loop walks down every row of that column, so every pixel is visited exactly once.