Time to get to know me a little.

About Me

- I'm Clinton
- International Student from Ghana
- Undergrad in CS (Systems)
- Doing my Coterm in CS (Computer and Network Security)

What will we do here?

- Recap what we did in lecture last week and Monday
- Go over some helpful tips for assignments
- Practice problems!

Plan for Today

- Bit, Function Decomp, Control Flow
- Functions more in depth
- Images
- Practice problems!

Any Questions?

Bit

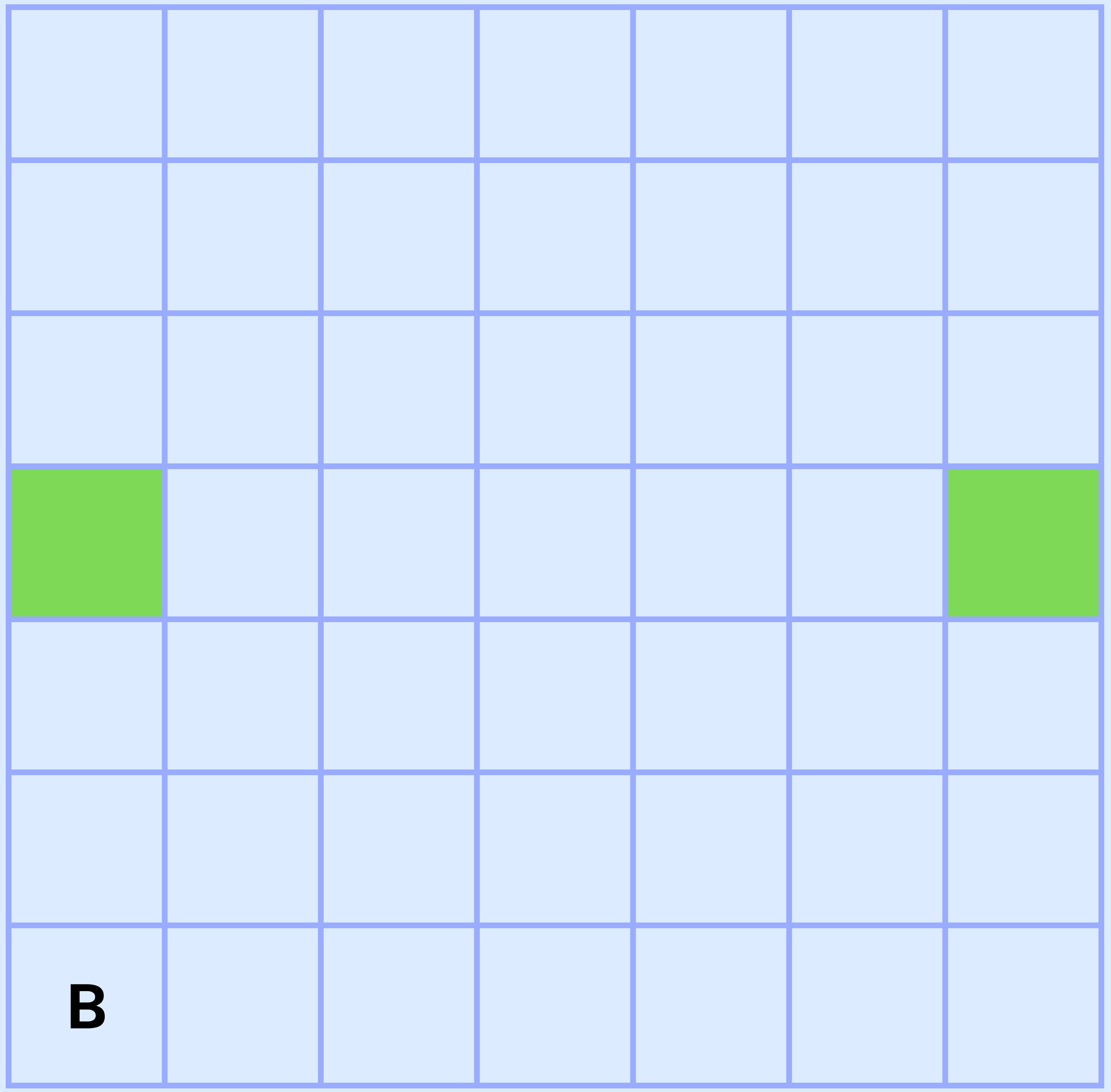
- `bit.front_clear(),`
  `bit.left_clear(),`
  `bit.right_clear()`
- `bit.move()`
- `bit.paint(color)`
- `bit.get_color()`
- `bit.left(), bit.right()`

Function Decomp!

- Break problem down into smaller, logical subproblems to make the solution easy to read and understand

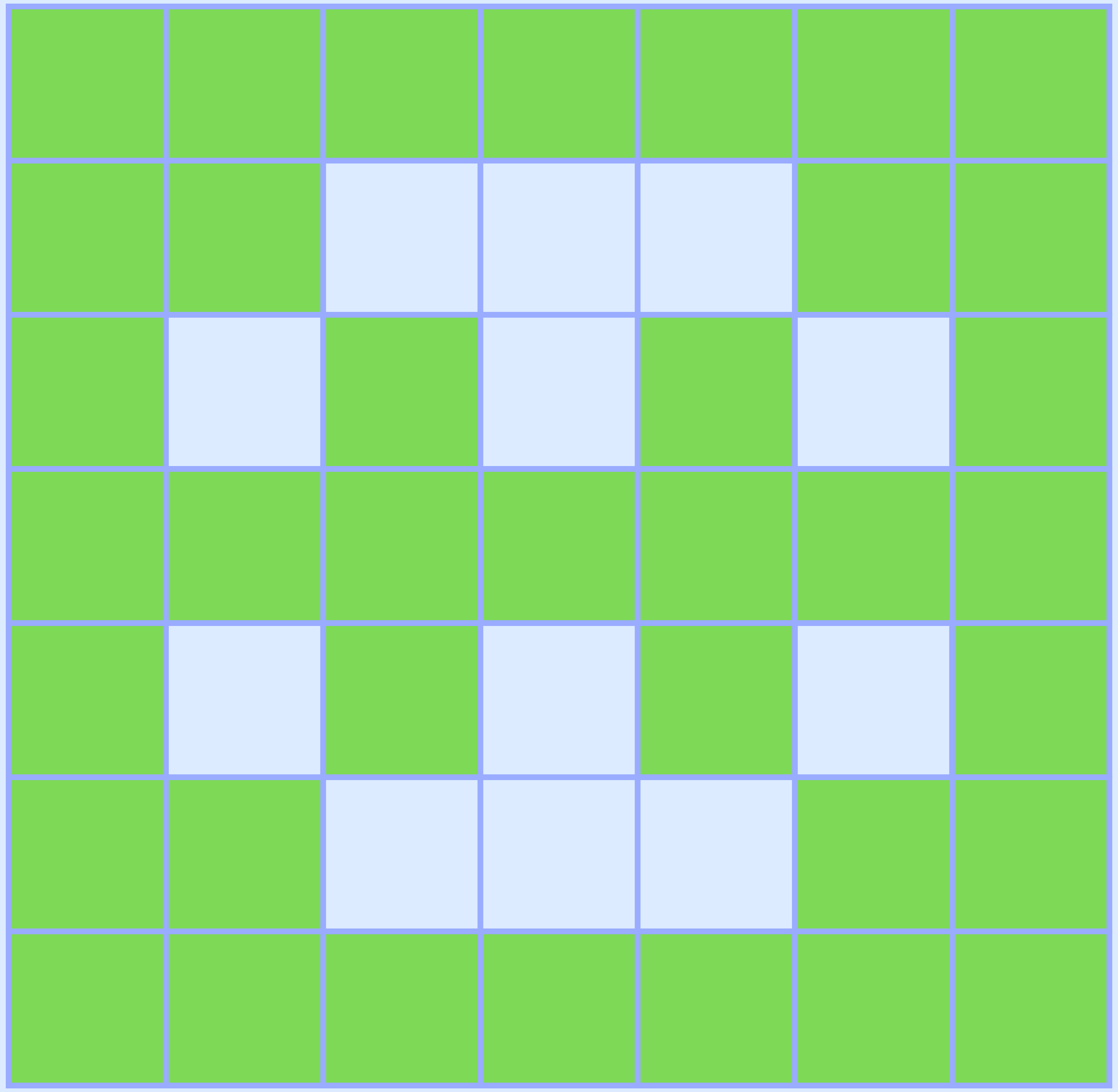
Function Decomp!

The world starts this way



Function Decomp!

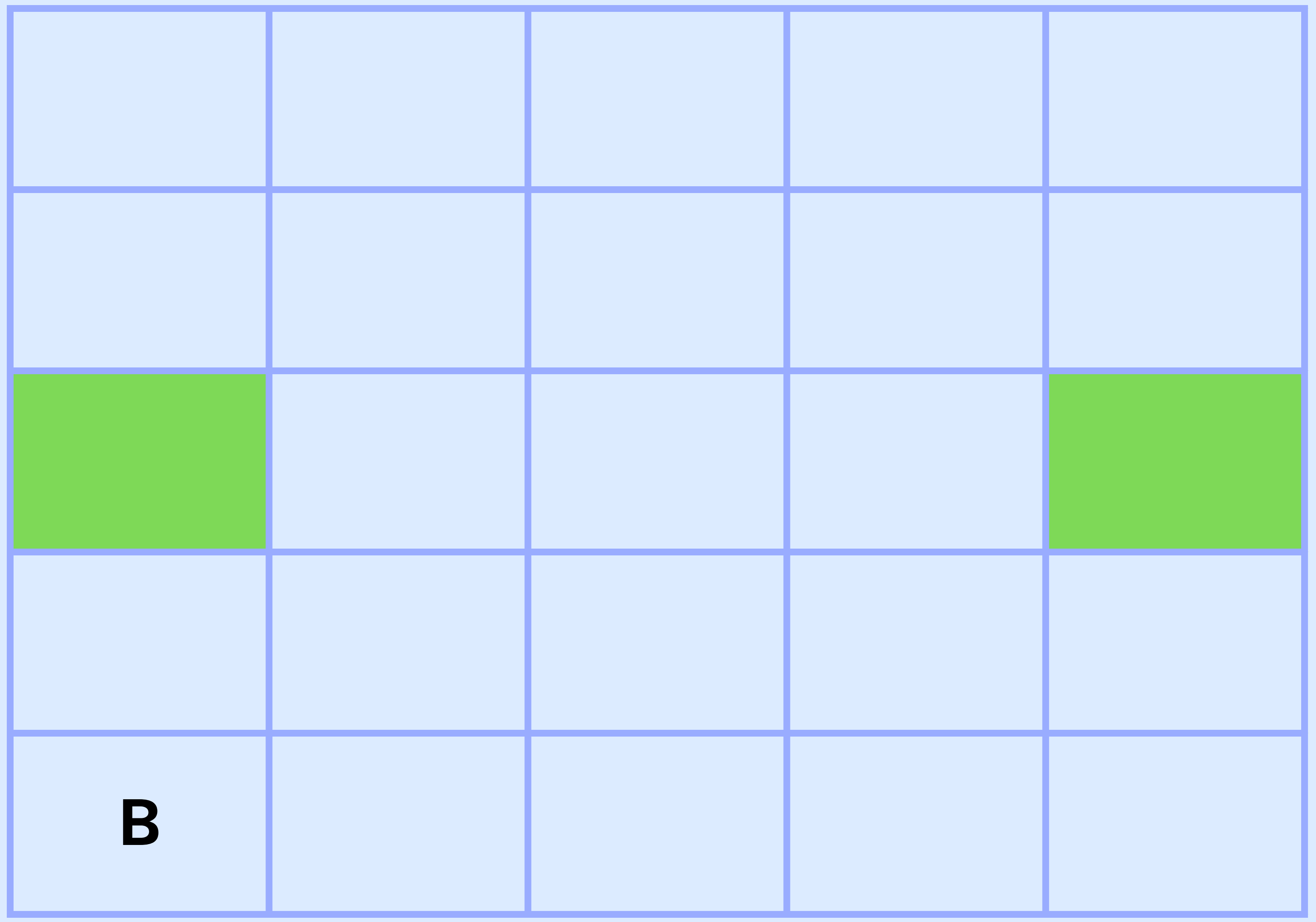
The world ends this way. It doesn't matter where Bit ends up



Function Decomp!

Another example

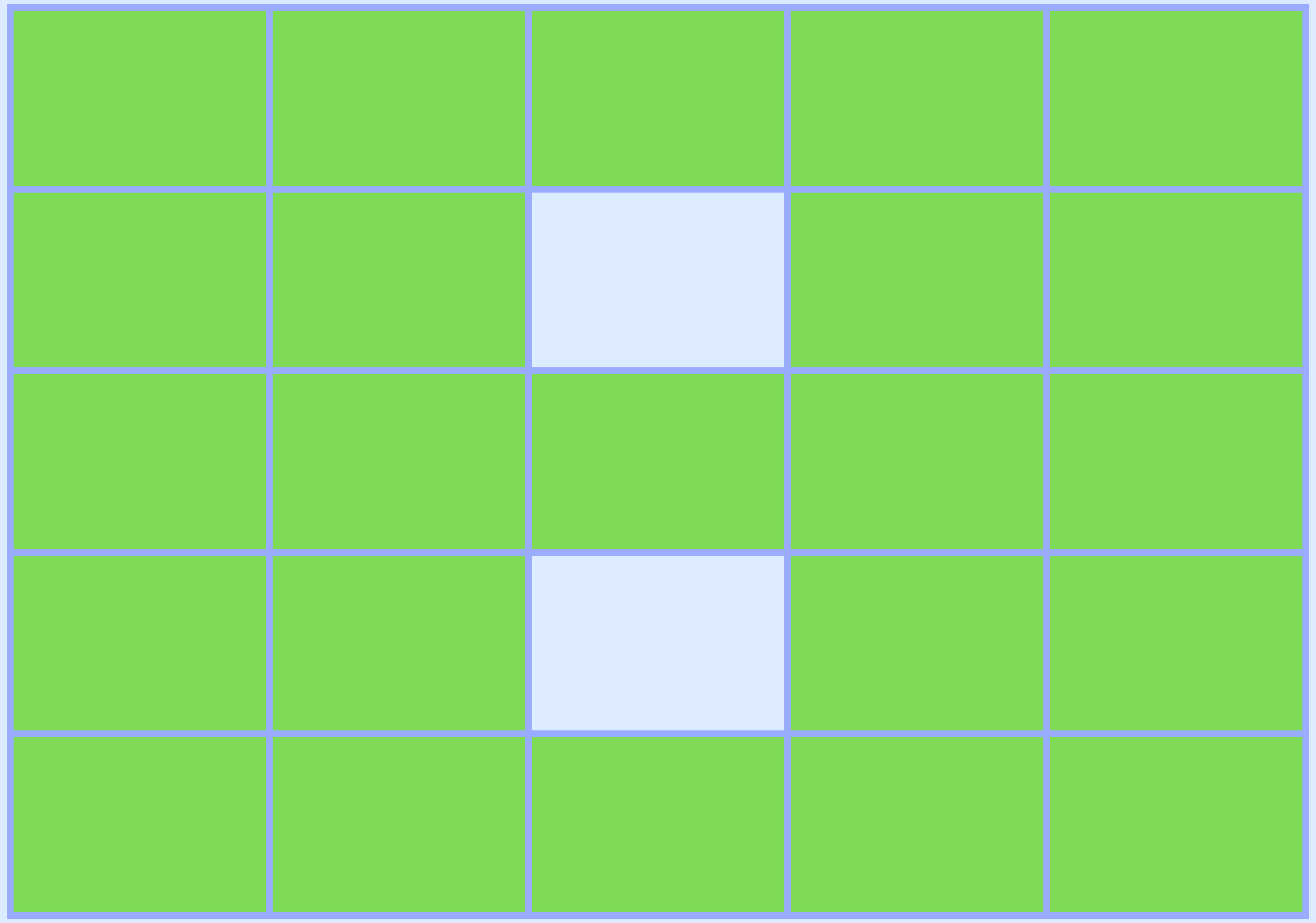
The world starts this way



Function Decomp!

Another example

The world ends this way



Decomposition Strategies

One good idea!

```
def solution(bit):
    move_to_middle(bit)
    color_middle_row(bit)
    color_bounding_box(bit)
    color_left_diagonal(bit)
    color_right_diagonal(bit)
```

Let's dive deeper

Let's think about Pre/Post conditions of each function

Precondition: What we expect to be true about our program **before** our function runs

Postcondition: What we expect to be true about our program **after** our function runs

move_to_middle(bit)

Pre-conditions:

Post-conditions:

color_middle_row(bit)

Pre-conditions:

Post-conditions:

color_bounding_box(bit)

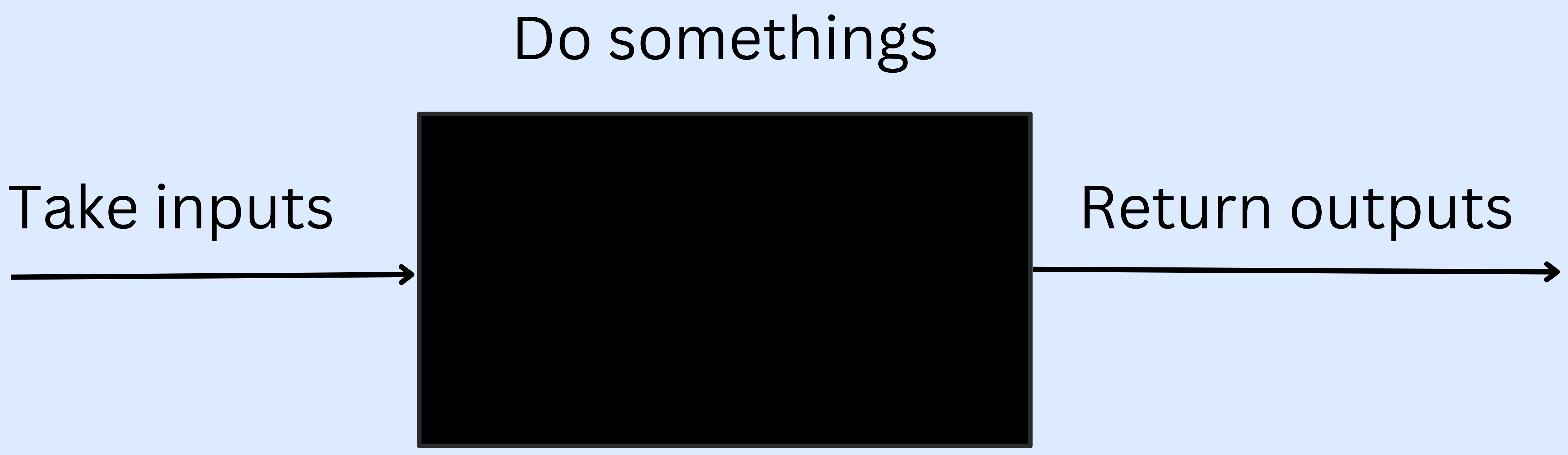
Pre-conditions:

Post-conditions:

Any Questions?

Functions in Depth!

The black box model



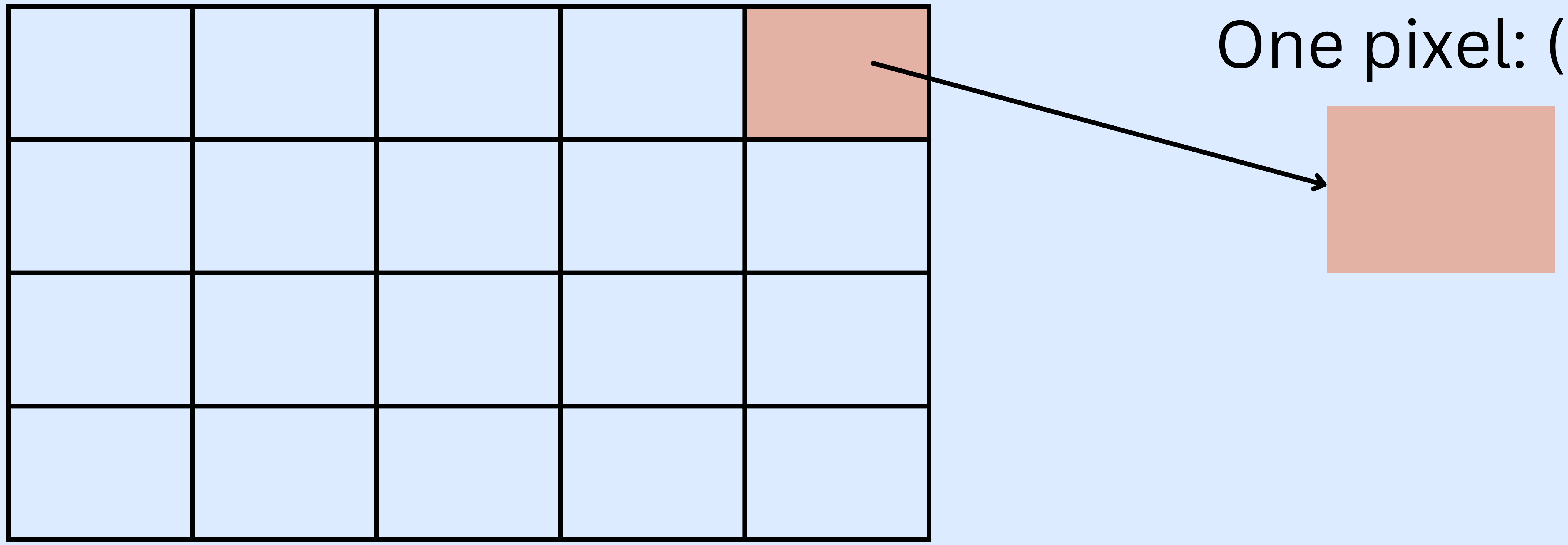
Functions in Depth!

We've seen this before ...

```
bit.front_clear()
bit.move()
bit.paint(color)
color = bit.get_color()
```

Images

Grid of Pixels!



SimpleImage Library

- Create a blank image
  - **blank_image** = SimpleImage.blank(**width**, **height**)
- Load an image
  - **image** = SimpleImage(**filename**)
- Get a pixel from image defined above
  - **pixel** = **image**.get_pixel(**x**, **y**)
- Set value in a pixel
  - **pixel**.green = 255

Looping through an image

Aside on Python For loops...

Syntax: for var_name in range(start, end)
for var_name in range(end)

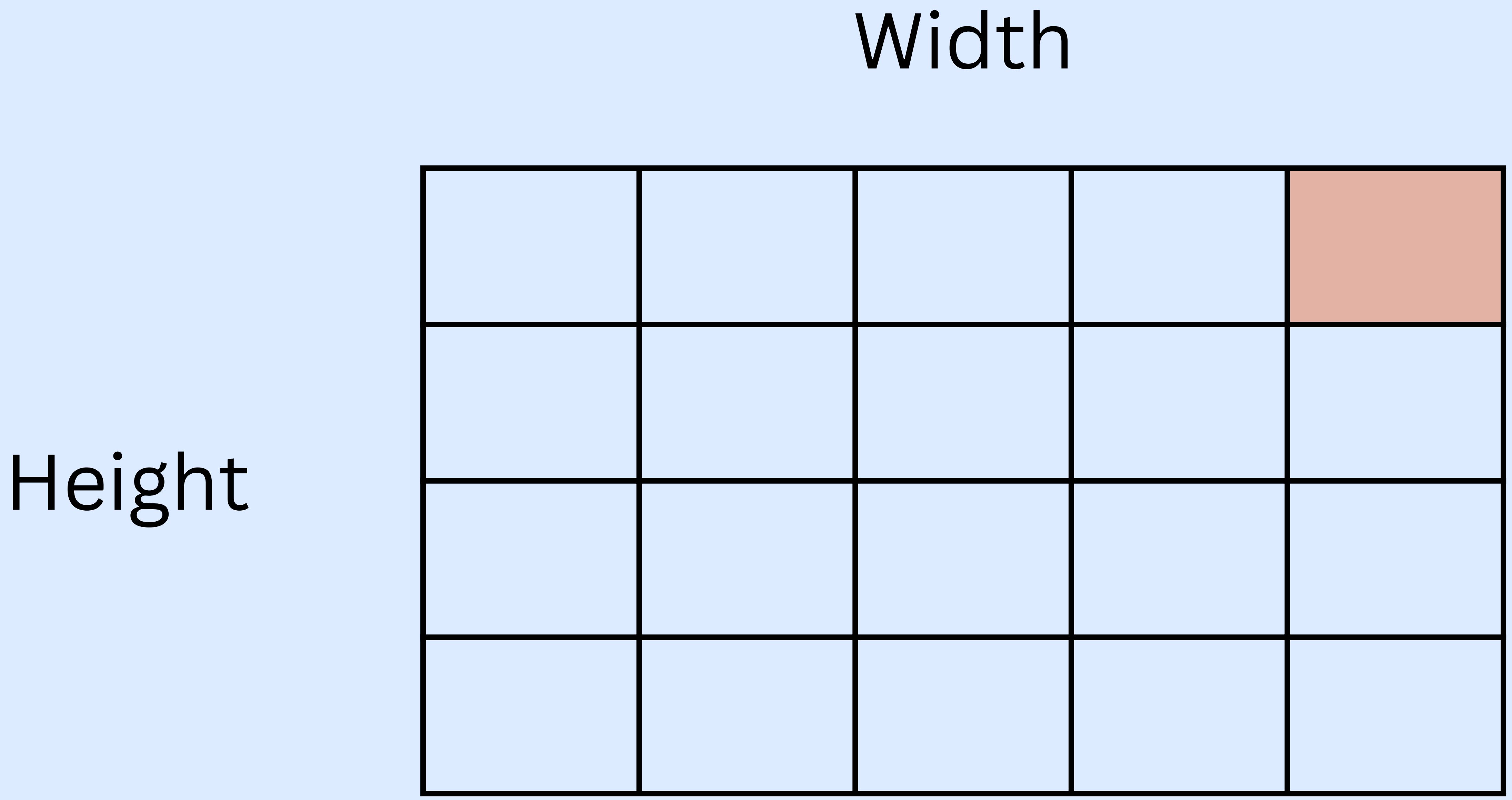
Examples:

```
for x in range(0, 10):
    print(x)

for y in range(0, 10):
    print(y)

start = 12
end = 25
for y in range(start, end):
  print(y)
```

Looping through an image



Looping through an image

Nested loops!

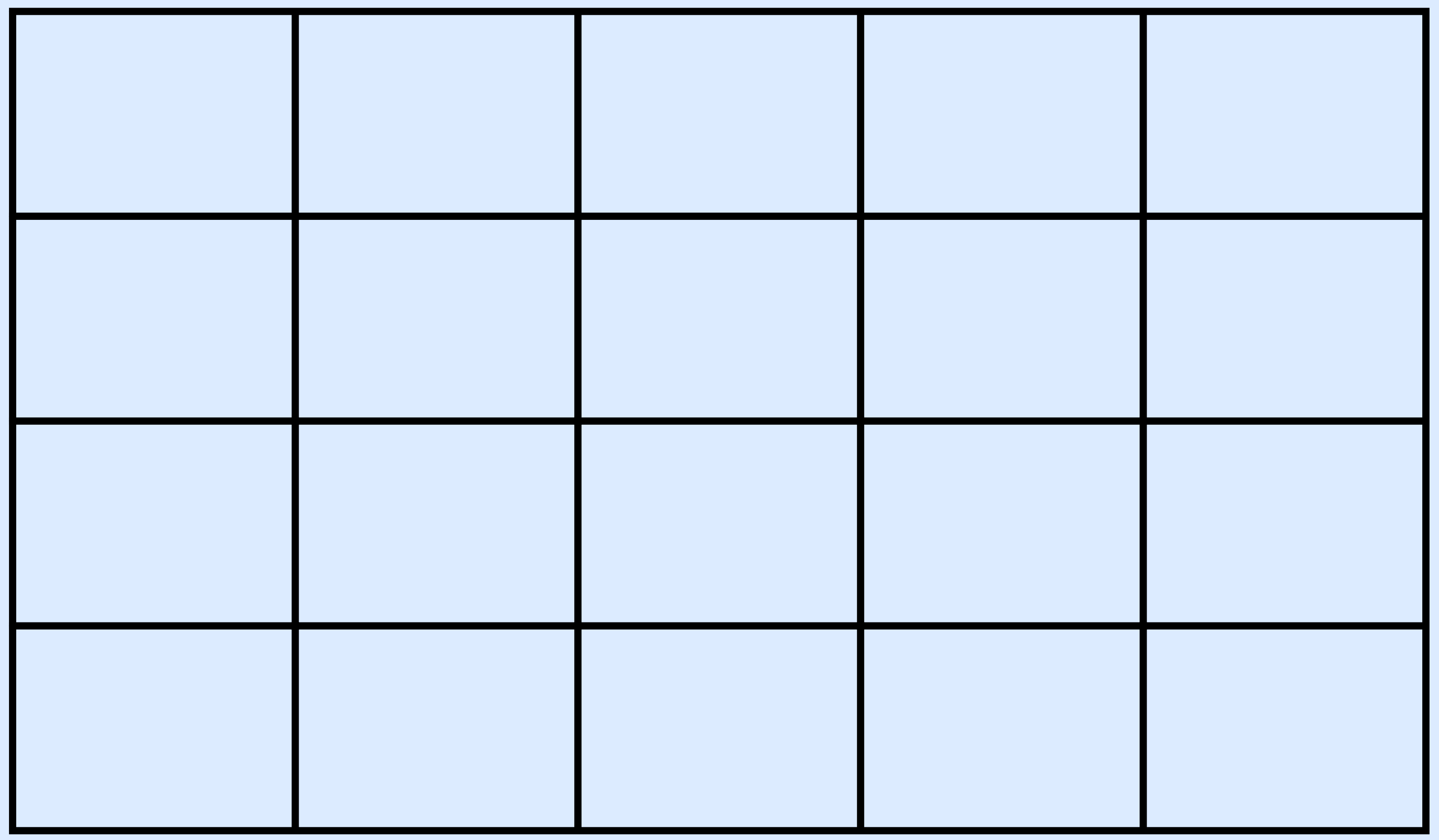
```
for y in range(0, image.height):
    for x in range(0, image.width):
        pixel = image.get_pixel(x, y)
```

Because we start from 0 ...

```
for y in range(image.height):
    for x in range(image.width):
        pixel = image.get_pixel(x, y)
```

Looping through an image

What's actually happening?



We end up processing the first row, then second, ..., to the last

Practice Problems!

Remember this?

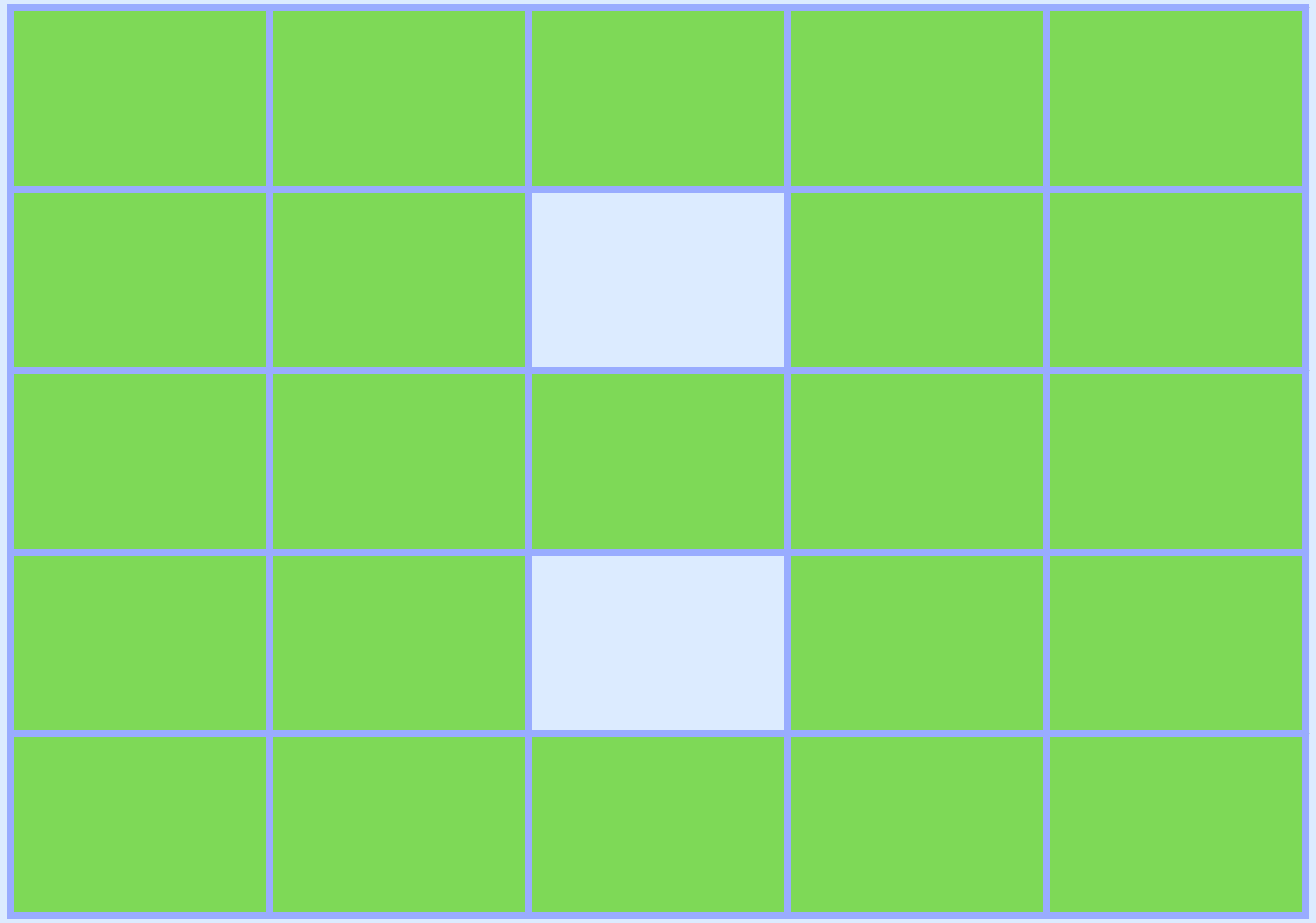
What if the world starts this way?

B				

Practice Problems!

Remember this?

But we still want the same outcome? How would you decompose this problem?



Hint hint! We only need to add one new function to our earlier decomp. What should that function do?

Practice Problems!

Off to the experimental Server!

Bit:

- **beloved**
- **fill_all**

Images:

- **darker**
- **darker left**

Your homework