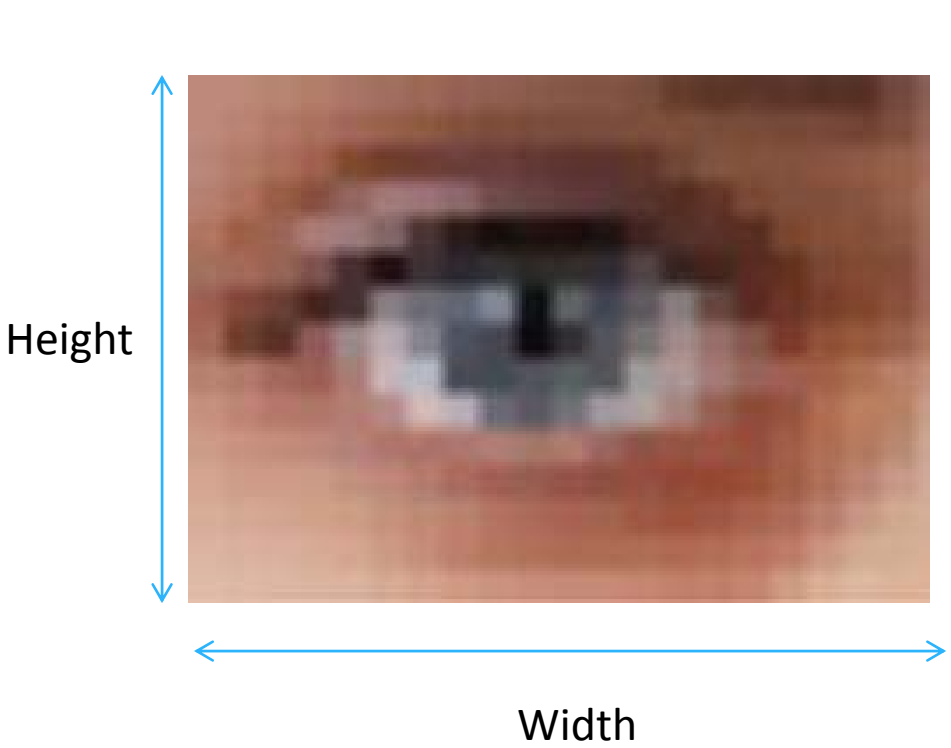
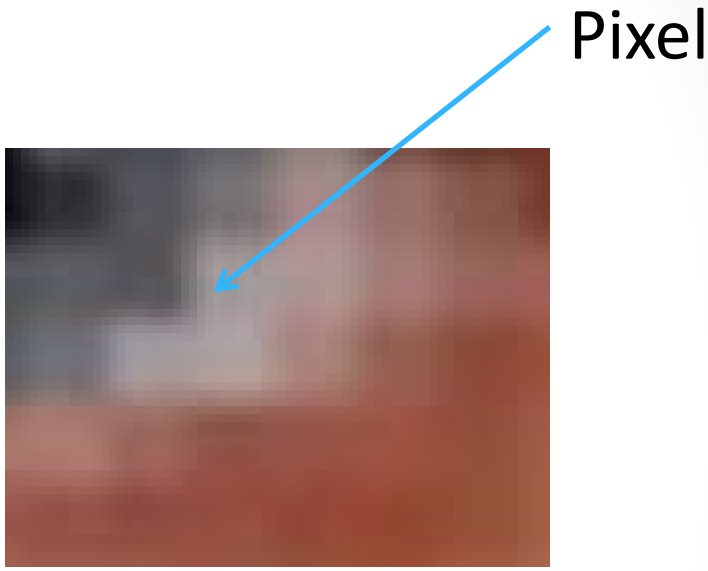
Pictures and Loops

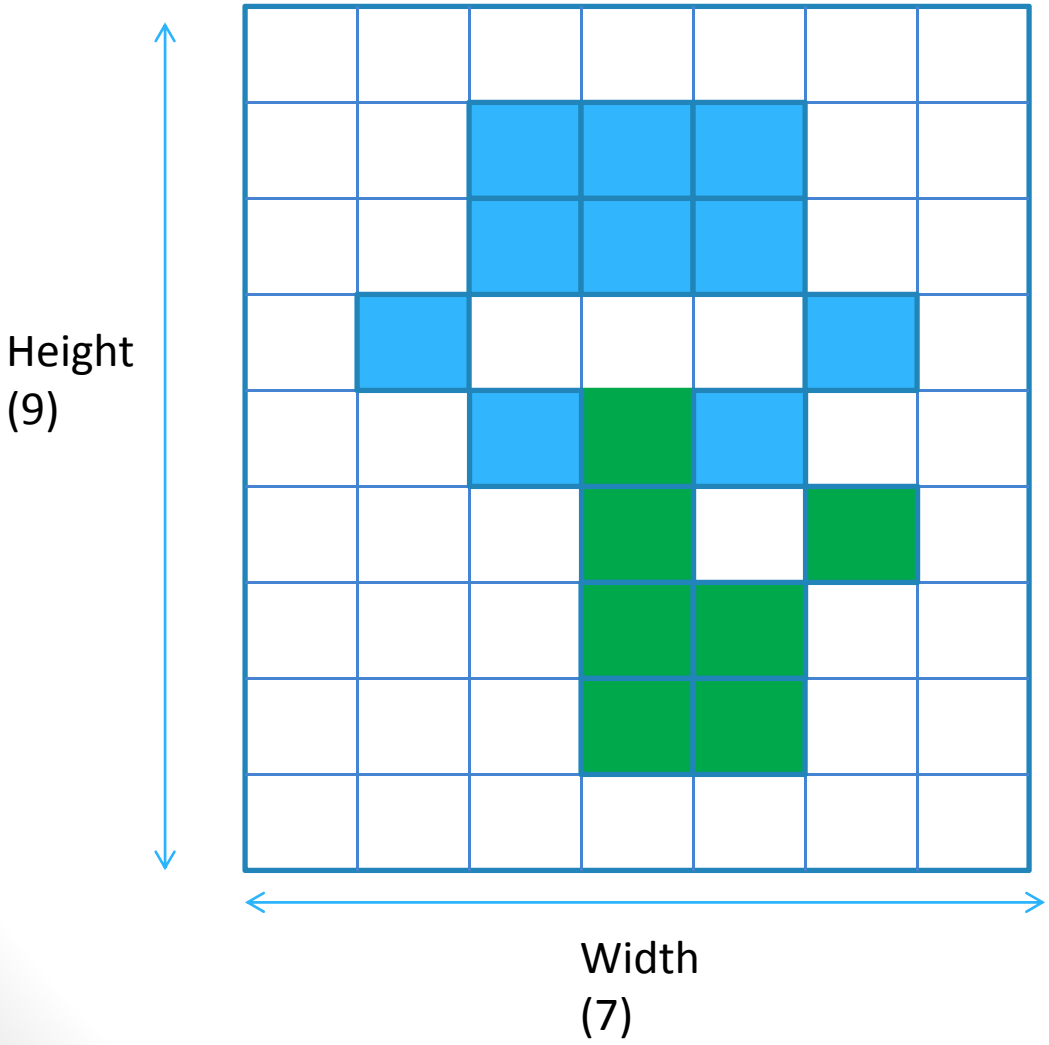
Guttag chapter 4
Guttag chapter 5

How are pictures represented on a computer?

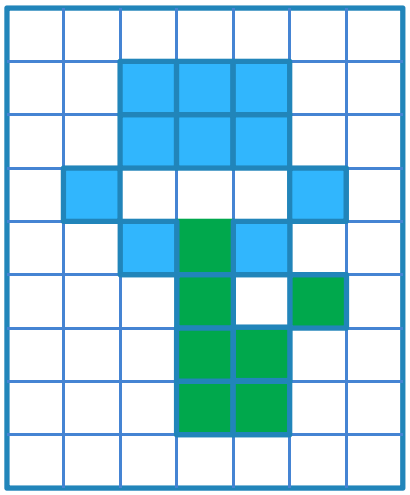




A picture is a grid of “Pixels”



Each Pixel is a single Color… so how is color represented?



RGB Model for color representation

A color is made up of:

- Some amount of Red (0 … 255)
- Some amount of Green (0 … 255)
- Some amount of Blue (0 … 255)

Together these three channels, when combined, describe the entire range of visible colors

What color is represented by (100, 100,100)?

A. Black
B. White
C. Brown
D. Gray
E. Salmon

Python's Etch-a-Sketch

```
from PIL import Image

def firstDraw():
    pic = Image.new('RGB', (200,200), (0, 0, 0))

    # Write code that draws something on this canvas



    # This line will stay the same so that you can see what was drawn
    pic.show()
```

mode (width,height) color

The mode attribute defines the number and names of the bands in the image, and also the pixel type and depth. Common modes are “L” (luminance) for greyscale images and “RGB” for true color images

Creating a new picture

```
>>> pic = Image.new('RGB', (200,400), (255, 255, 255))
>>> pic.show()
```

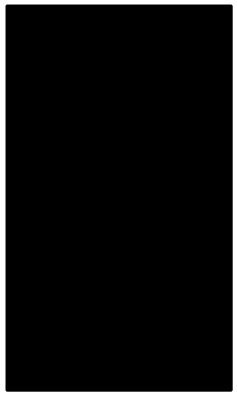
Which of the following is displayed by the above code?

A.

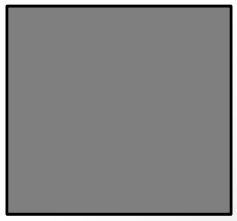
B.

C.

D.



E.



Opening an existing picture

```
>>> pic = Image.open(“butterfly.jpg”)  # Open a file
>>> pic.show()
```

How can we find the width and height of the image?

Getting information about pictures

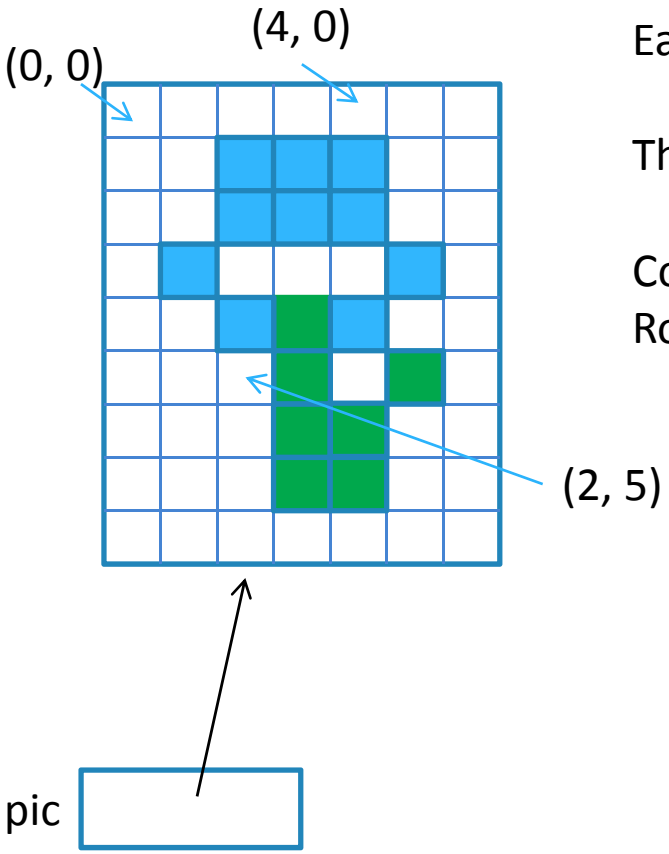
```
>>> pic = Image.open(“butterfly.jpg”)  # Open a file
>>> pic.show()

>>> w = pic.size[0] # get the width of the picture
>>> h = pic.size[1] # get the height of the picture


>>> (w, h) = pic.size
```

pic.size is a variable associated with the Image object. It is a tuple with two elements (width, height)

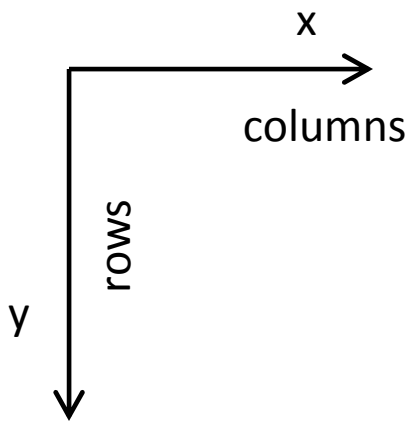
Accessing Pixels in a Picture



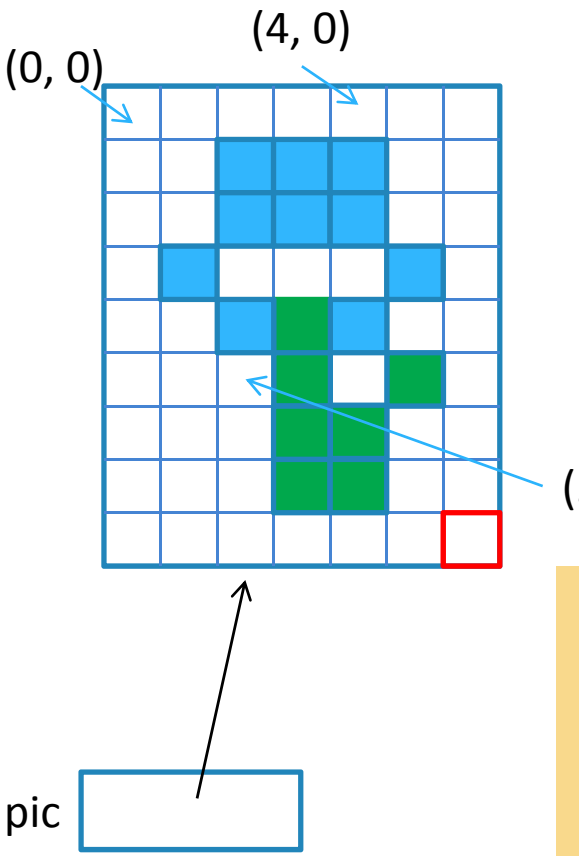
Each pixel can be accessed via its row and column

The pixel in the **upper left** is at row **0**, column **0**.

Columns increase to the right (i.e. x axis)
Rows increase ***down*** (i.e. y axis)



Accessing Pixels in a Picture



Each pixel can be accessed via its row and column

The pixel in the **upper left** is at row **0**, column **0**.

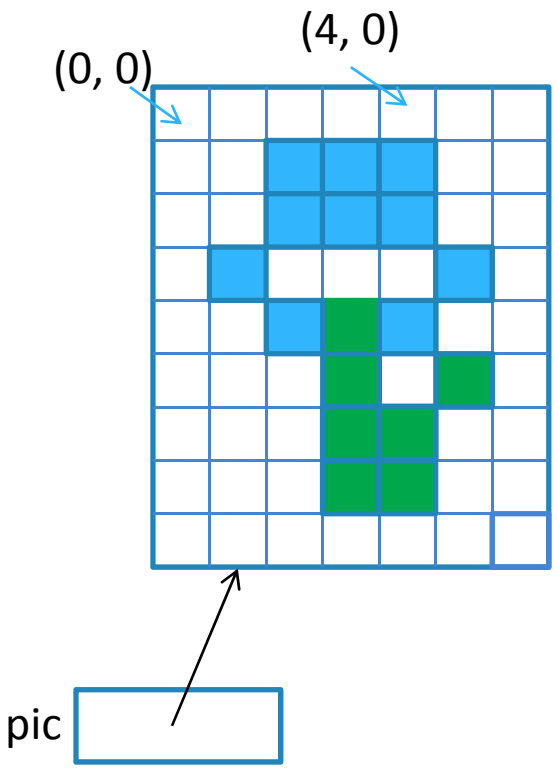
Columns increase to the right (i.e. x axis)
Rows increase ***down*** (i.e. y axis)

What value represents the *row* of the pixel in the lower right corner of any picture, pic?

A. 0
B. pic.size[0]
C. pic.size[1]
D. pic.size[0]-1
E. pic.size[1]-1

pic.size is a tuple with two elements (width, height)

Accessing Pixels in a Picture

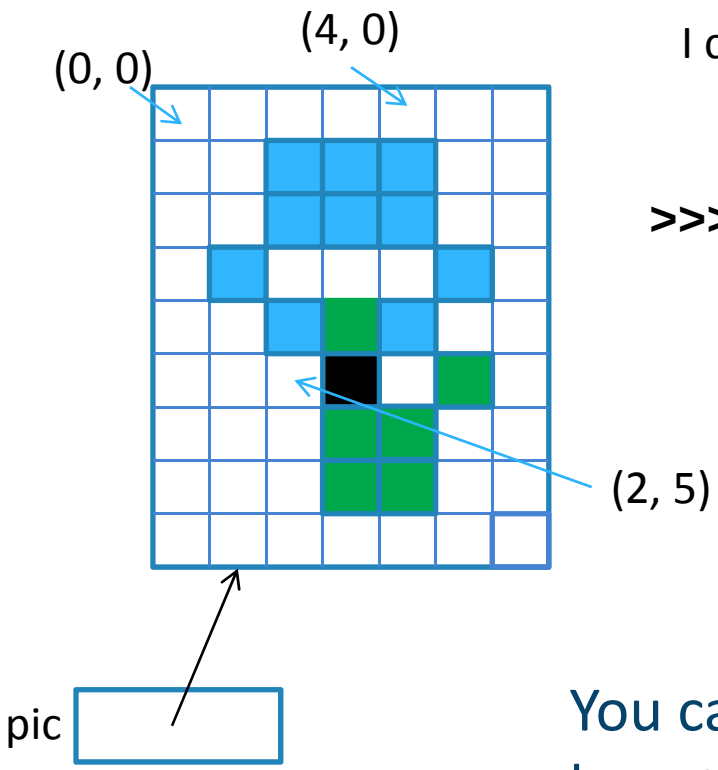


You can retrieve (the color values of) a single pixel

```
>>> pix = pic.getpixel( (3, 5) )
>>> pix
```

A. (255,255,255)
B. (255,0,0)
C. (0,255,0)
D. (0, 0, 255)
E. None of the above.

Modifying Pixels in a Picture



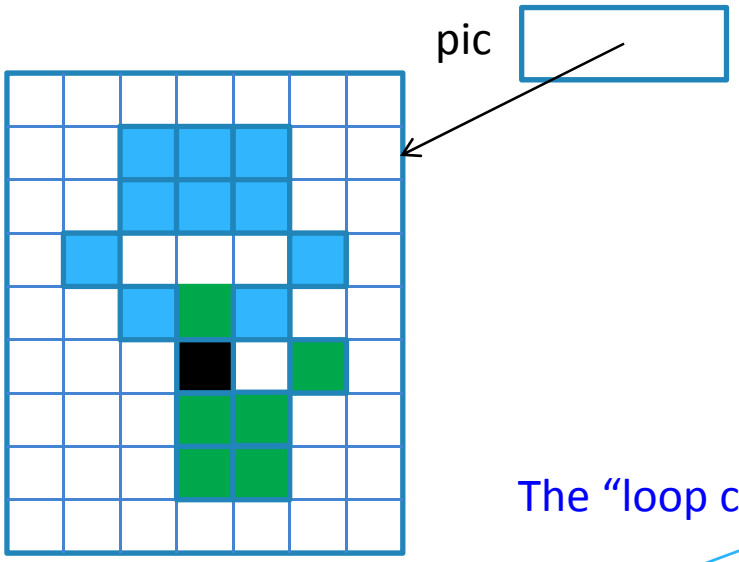
I can set the color of a pixel based on its coordinates:

x y

```
>>> pic.putpixel( (3,5), (0,0,0) )
```

You can programmatically modify a picture by retrieving individual pixels and changing their color! The key is to know *which pixels* to change and *what colors* to change them to…

Loops for pixel modification



The "loop control variable"

Keyword "for"

Keyword "in"

list

```
for x in [0, 1, 2, 3, 4, 5, 6]:
    pic.putpixel( (x,0),(100, 100, 100))
```

Loop body

Loops

```
for x in [0, 1, 2, 3, 4, 5, 6]:
    print x
```

The loop body will execute one time for each element in the list. Each time through the loop, the loop control variable will take the value of the next element in the list.

Lists without listing

```
for x in range(7):
    print x_col
```

```
for x in [0,1,2,3,4,5,6]:
    print x_col
```

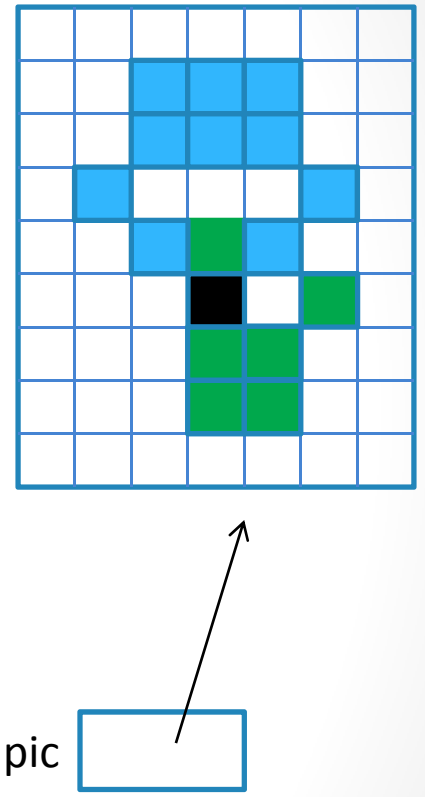
```
>>> list(range(7))
[0, 1, 2, 3, 4, 5, 6]

>>> list(range(1, 7))
[1, 2, 3, 4, 5, 6]
```

Draw a line

```
for x in range(7):
    pic.putpixel( (x,0),(100,100,100))
```

```
pic.putpixel( (0, 0 ),(100,100,100))
pic.putpixel( (1, 0 ),(100,100,100))
pic.putpixel( (2, 0 ),(100,100,100))
pic.putpixel( (3, 0 ),(100,100,100))
pic.putpixel( (4, 0 ),(100,100,100))
pic.putpixel( (5, 0 ),(100,100,100))
pic.putpixel( (6, 0 ),(100,100,100))
```

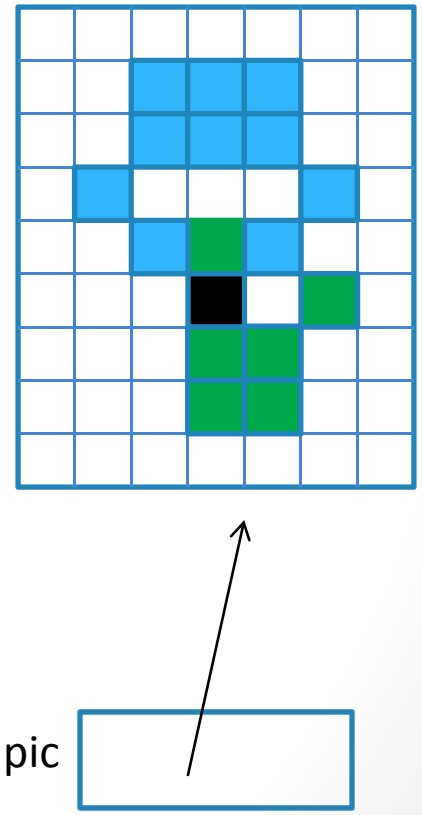


Lists without listing

```
for x in range(7):
    pic.putpixel( (x,0),(100,100,100))
```

```
>>> list(range( pic.size[0] ))
???
```

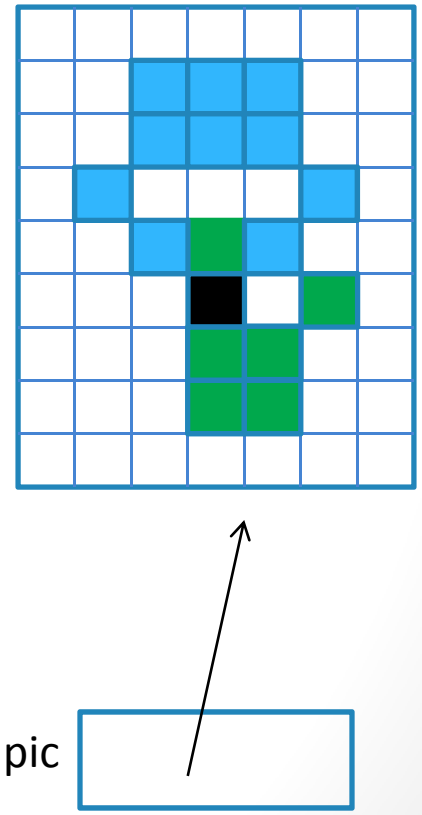
A. [0,1,2,3,4,5,6]
B. [1,2,3,4,5,6,7]
C. [1,2,3,4,5,6]
D. [0,1,2,3,4,5,6,7]
E. None of the above.



Draw a line

```
for x in range(pic.size[0]):
    pic.putpixel( (x,0),(100,100,100))
```

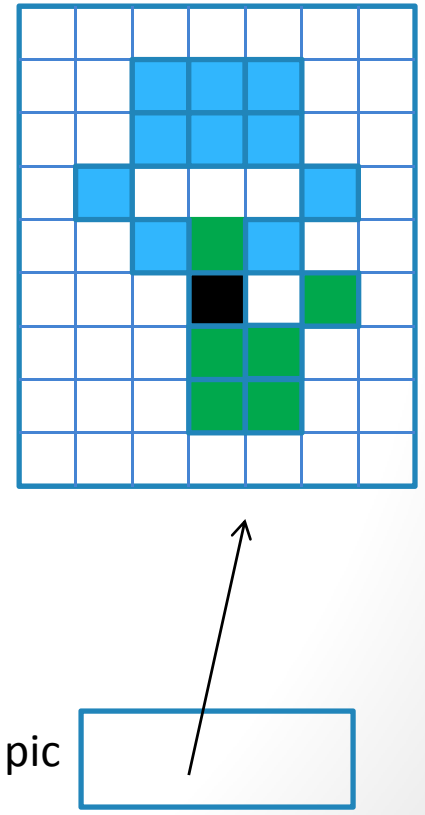
What's going to happen?



Loops for pixel modification

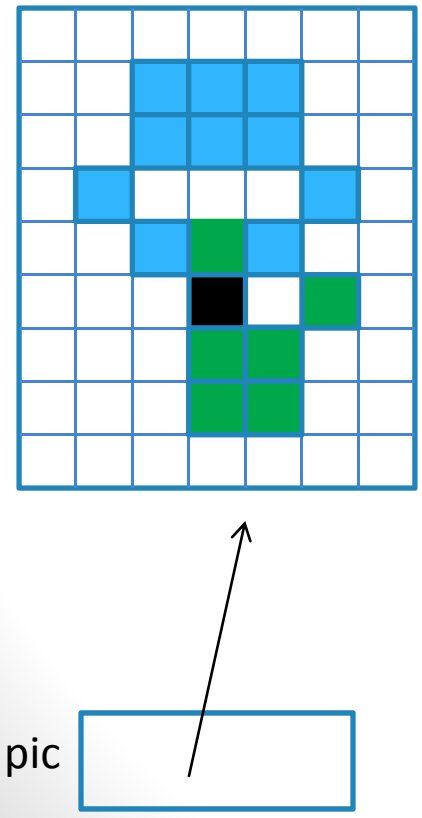
```
for x_col in range(pic.size[0]):
    pic.putpixel( (x_col,0),(100,100,100))
```

```
pic.putpixel( (0, 0 ),(100,100,100))
pic.putpixel( (1, 0 ),(100,100,100))
pic.putpixel( (2, 0 ),(100,100,100))
pic.putpixel( (3, 0 ),(100,100,100))
pic.putpixel( (4, 0 ),(100,100,100))
pic.putpixel( (5, 0 ),(100,100,100))
pic.putpixel( (6, 0 ),(100,100,100))
```



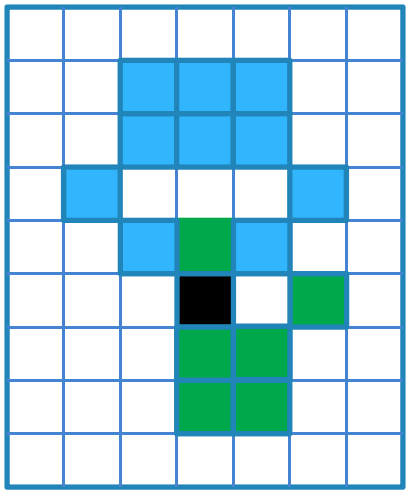
Nested loops

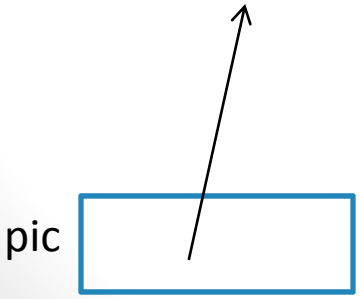
```
for y in [1, 3, 5, 7]:
   for x in range(pic.size[0]):
      pic.putpixel( (x,y),(100, 100, 100))
```



Nested loops for modifying the whole picture

```
for x in range(pic.size[0]//2):
    for y in range(pic.size[1]):
        pic.putpixel( (x,y), (100, 100, 100))
```





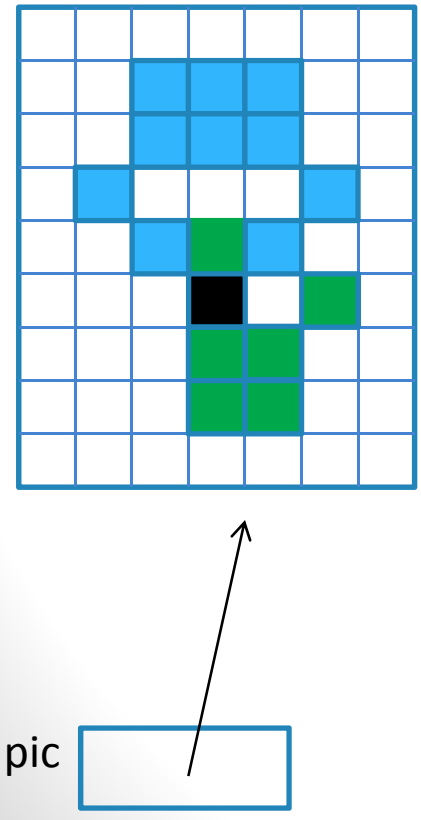
What does the code above do?

A. Turns the whole picture gray
B. Turns the top half of the picture gray
C. Turns the bottom half of the picture gray
D. Turns the right half of the picture gray
E. Turns the left half of the picture gray

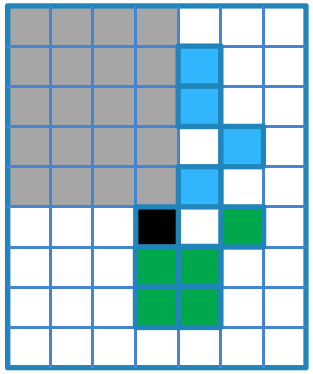
test04

If statements work in loops too!

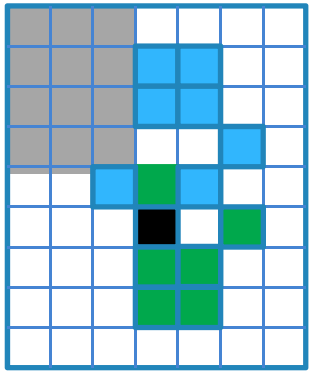
```
for x in range(pic.size[0]):
    for y in range(pic.size[1]):
        if y < pic.size[1]//2 and x < pic.size[0]//2:
            pic.putpixel( (x,y), (100, 100, 100))
```



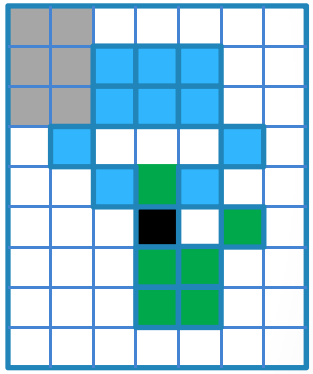
A



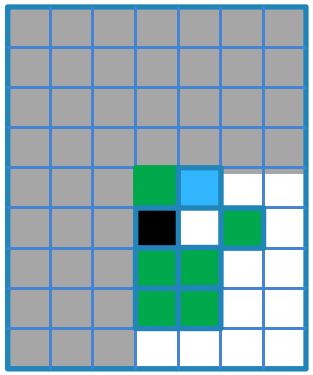
B



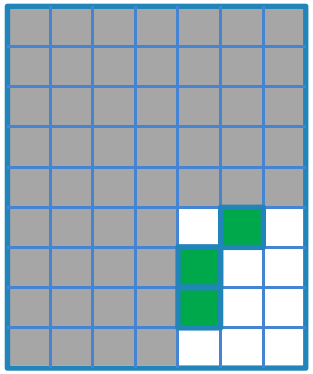
C



D



E

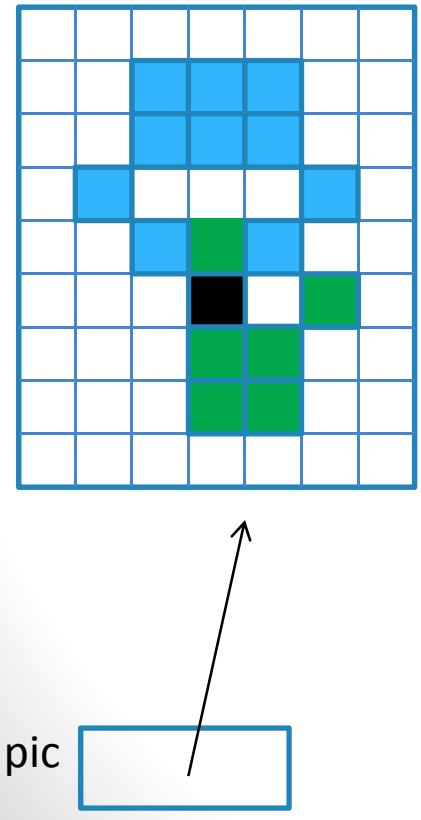


What is the resulting pic?

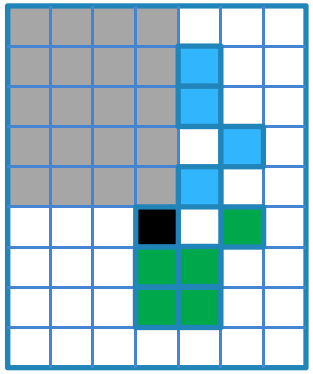
test05

If statements work in loops too!

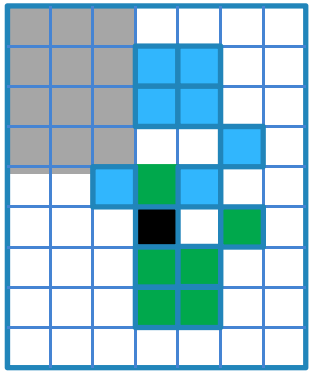
```
for x in range(pic.size[0]):
    for y in range(pic.size[1]):
        if y < pic.size[1]//2 or x < pic.size[0]//2:
            pic.putpixel( (x,y), (100, 100, 100))
```



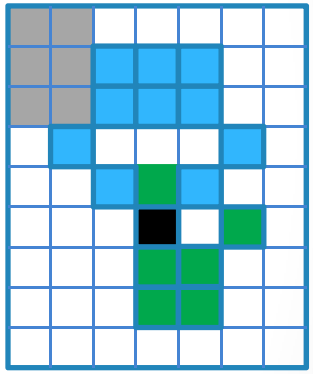
A



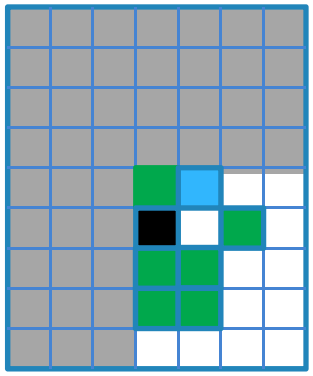
B



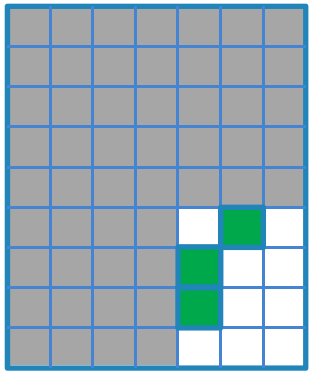
C



D



E



What is the resulting pic?