Week 3: Depth

Mutable vs. immutable data, nested loops, geometric transformations

How comfortable are you with mutable data types in Python?

A. Very comfortable

B. Know how they work, but have trouble using them with functions

C. Not sure what they are

How comfortable are you with loops?

A. Very comfortable

B. Know how they work, but have a hard time determining the range of loop variable when working on pictures

C. Have trouble with nested loops

D. Don’t know much about them

How comfortable are you with image manipulations?

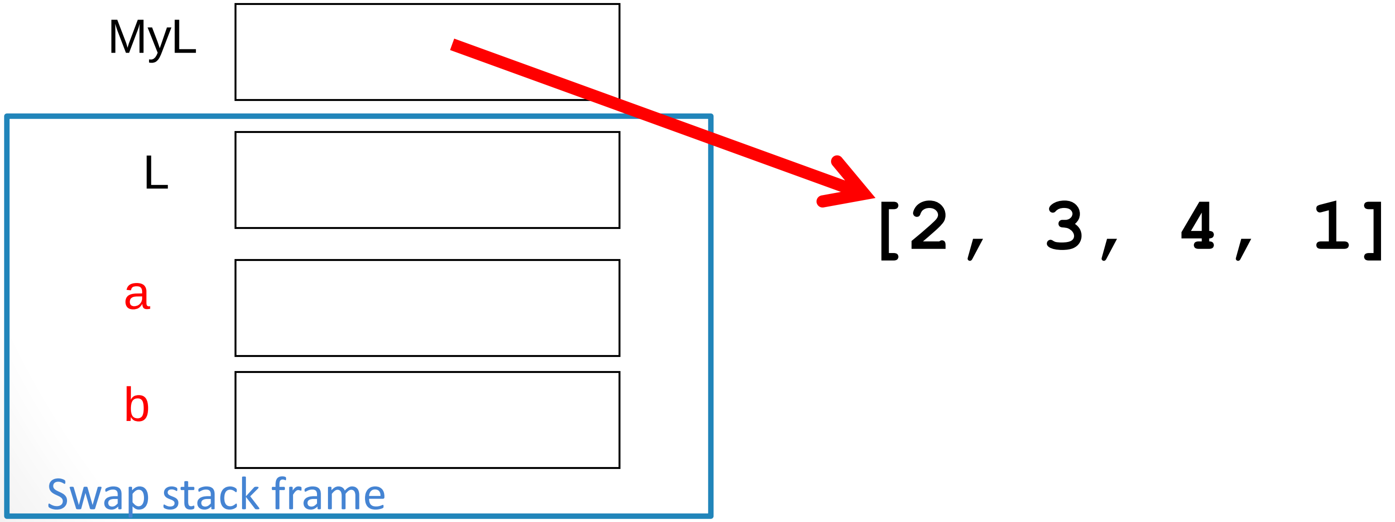
A. Very comfortable
B. Fair understanding, not sure about shifting images up and down
C. Have a hard time with color and geometric transformations
D. Not comfortable at all

Functions and Mutable Types

```
def swap(L, a, b):
    temp = a
    a = b
    b = temp

>>> myL = [2, 3, 4, 1]
>>> swap(myL, myL[0], myL[3])
>>> print(myL)
??
```

What gets printed?

A. [2, 3, 4, 1]
B. [1, 2, 3, 4]
C. [1, 3, 4, 2]
D. Something else

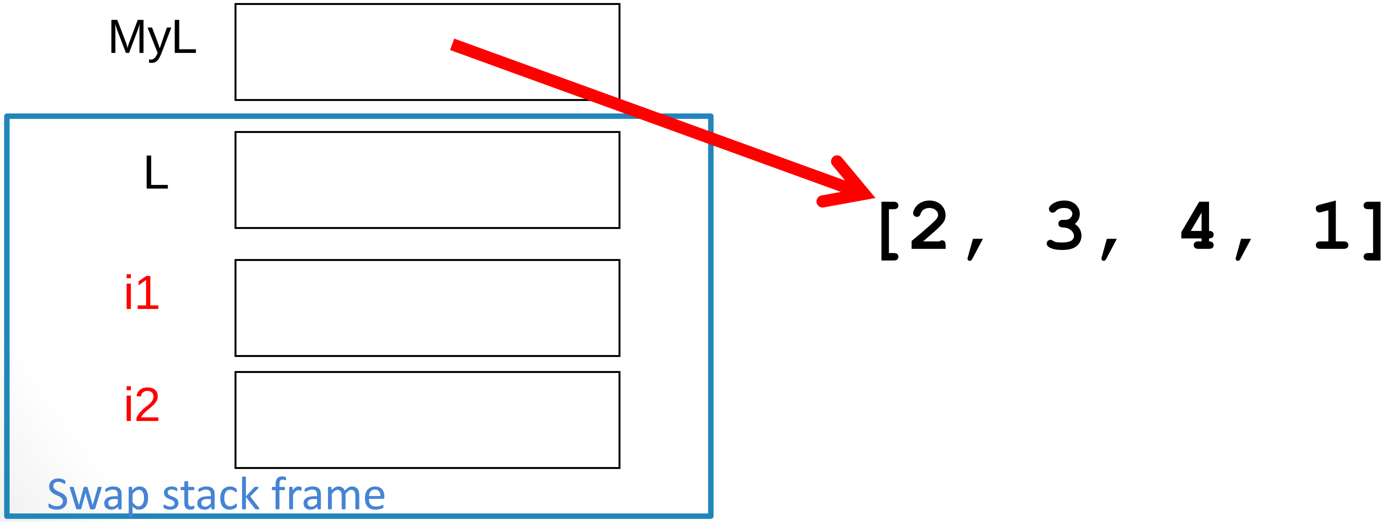


Functions and Mutable Types

```
def swap(L, i1, i2):
    temp = L[i1]
    L[i1] = L[i2]
    L[i2] = temp

>>> myL = [2, 3, 4, 1]
>>> swap(myL, 0, 3)
>>> print(myL)
??
```

What gets printed?

A. [2, 3, 4, 1]
B. [1, 2, 3, 4]
C. [1, 3, 4, 2]
D. Something else



Practice with Pictures

```
def doStuff( pic ):
  pic = Image.new('RGB',(pic.size[0],pic.size[1]),(255,255,255))
  for x in range( pic.size[0] ):
    for y in range( pic.size[1] ):
       pic.putpixel((x,y), ( 100, 100, 100 ) )

>>> myP = Image.open( "butterfly.gif" )
>>> doStuff( myP )
>>> myP.show()
```

Assume that the user chose a butterfly picture when the dialog box came up. What picture will be displayed?

A. A butterfly
B. A picture that is all gray
C. A gray-colored butterfly
D. Something else

Practice with Pictures

```
def doStuff( pic ):
  pic = Image.new('RGB',(pic.size[0],pic.size[1]),(255,255,255))
  for x in range( pic.size[0] ):
    for y in range( pic.size[1] ):
       pic.putpixel((x,y), ( 100, 100, 100 ) )

>>> pic = Image.open( “butterfly.gif” )
>>> doStuff( pic )
>>> pic.show()
```

Assume that the user chose a butterfly picture when the dialog box came up. What picture will be displayed?

A. A butterfly
B. A picture that is all gray
C. A gray-colored butterfly
D. Something else

Practice with Pictures

```
def doStuff( pic ):
  pic = Image.new('RGB',(pic.size[0],pic.size[1]),(255,255,255))
  for x in range( pic.size[0] ):
    for y in range( pic.size[1] ):
        pic.putpixel((x,y), ( 100, 100, 100 ) )
  return pic

>>> pic = Image.open( “butterfly.gif” )
>>> doStuff( pic )
>>> pic.show()
```

Assume that the user chose a butterfly picture when the dialog box came up. What picture will be displayed?

A. A butterfly

B. A picture that is all gray

C. A gray-colored butterfly

D. Something else

Practice with Pictures

```
def doStuff( pic ):
  pic = Image.new('RGB',(pic.size[0],pic.size[1]),(255,255,255))
  for x in range( pic.size[0] ):
    for y in range( pic.size[1] ):
       pic.putpixel((x,y), ( 100, 100, 100 ) )
  return pic

>>> pic = Image.open( "butterfly.gif" )
>>> pic = doStuff( pic )
>>> pic.show()
```

Assume that the user chose a butterfly picture when the dialog box came up. What picture will be displayed?

A. A butterfly
B. A picture that is all gray
C. A gray-colored butterfly
D. Something else

Practice with Pictures

```
def doStuff( pic ):
 pic = Image.new(`RGB’,(pic.size[0],pic.size[1]),(255,255,255))
 for x in range( pic.size[0] ):
  for y in range( pic.size[1] ):
    pic.putpixel((x,y), ( 100, 100, 100 ) )

>>> myP = Image.open( “butterfly.gif” )
>>> doStuff( myP )
>>> myP.show()
```

How else can we modify doStuff(pic) in order to make myP gray ?

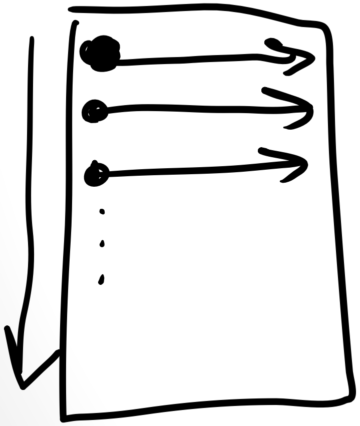
A. The only way is to add a return statement to doStuff
B. Comment out the first line of doStuff: pic= …..
C. Something else

Order matters...?

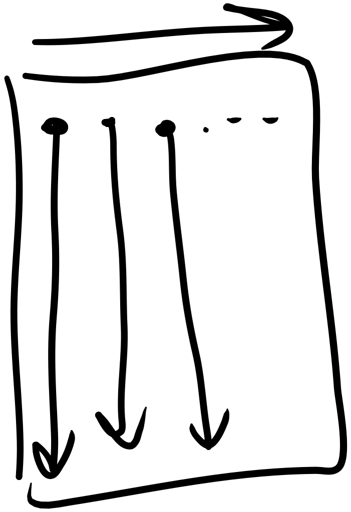
```
for x in range( pic.size[0] ):
   for y in range( pic.size[1] ):
     pic.putpixel((x,y), ( 100, 100, 100 ) )
```

In what order does the above code visit the pixels?

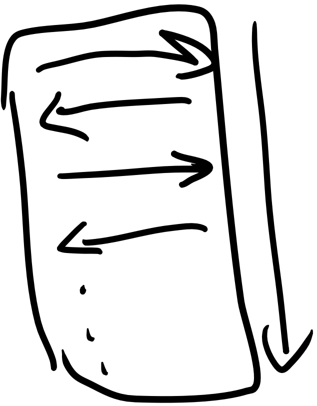
A.



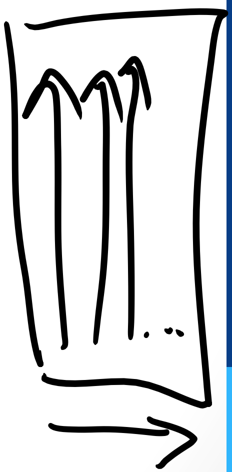
B



C



D

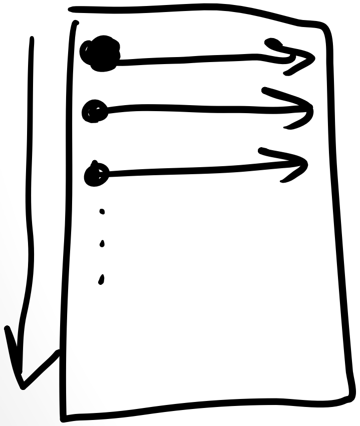


Order matters…?

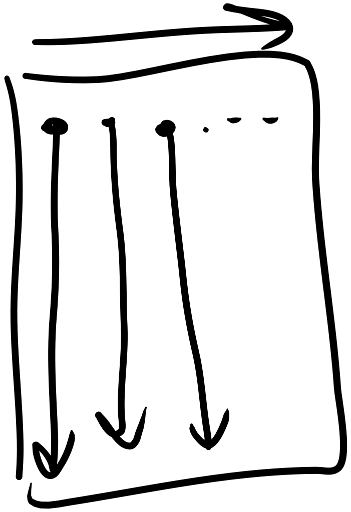
```
for y in range( pic.size[1] ):
    for x in range( pic.size[0] ):
        pic.putpixel((x,y), ( 100, 100, 100 ) )
```

In what order does the above code visit the pixels?

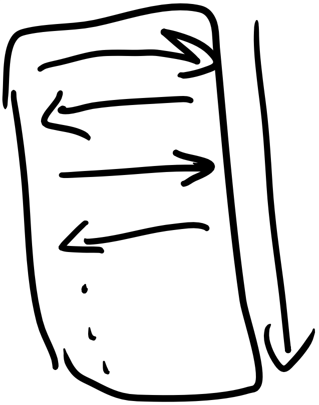
A.



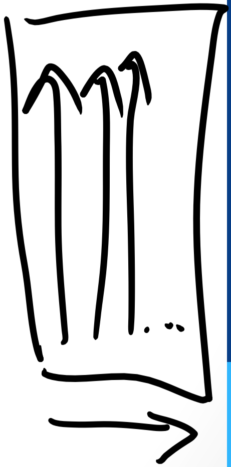
B



C



D



Order matters…

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

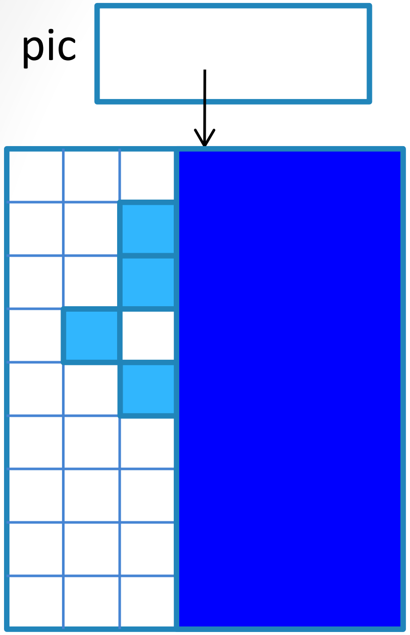
```
for x in range( pic.size[0] ):
   for y in range( pic.size[1] ):
     pic.putpixel((x,y), ( 100, 100, 100 ) )
```

Do the two pieces of code above do the same thing?

A. Yes

B. No

Writing nested for loops



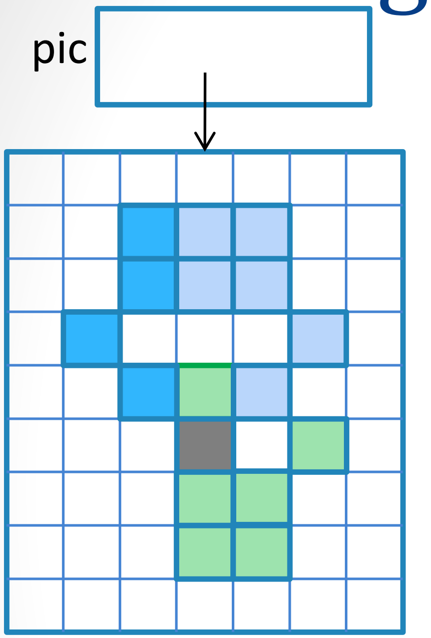
What should be the range of x?

A. pic.size[0]/2, pic.size[0]
B. pic.size[0]/2
C. pic.size[0]
D. Something else

Fill in the code below to make the right half of the picture *pure blue*

```
for x in range(______________________________):
   for y in range(______________________________):
      pic.putpixel((x, y),(0,0,255))
```

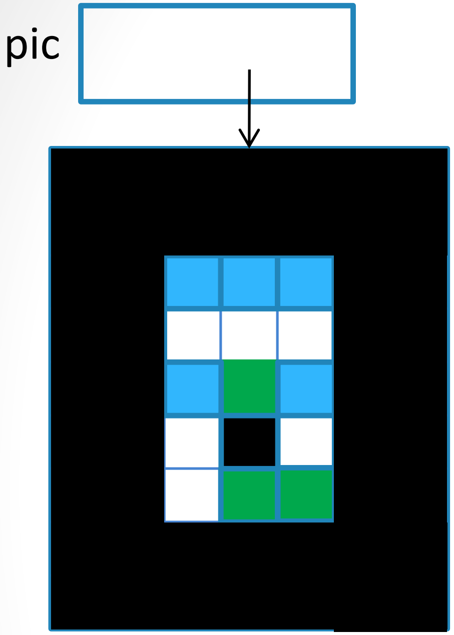
Writing nested for loops



Fill in the code below to make the right half of the picture *lighter* (without changing its color).

```
for x in range(pic.size[0]/2, pic.size[0]):
   for y in range(pic.size[1]):
      (red,green,blue) = pic.getpixel((x,y))
      newRed =    __________
      newGreen =  __________
      newBlue =   __________
      pic.putpixel((x, y),(newRed,newGreen,newBlue))
```

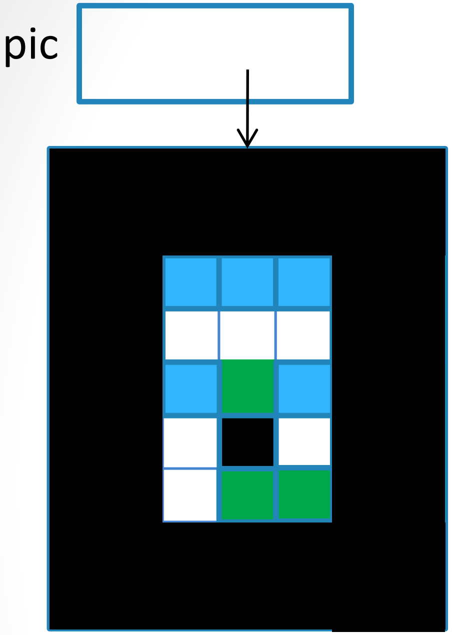
Using if-statements



Fill in the code below to create a black border 2 pixels wide around the border of a picture. Assume the picture is at least 2 pixels wide and tall.

```
for x in range(pic.size[0]):
   for y in range(pic.size[1]):

      if ( ________________________________

            _________________________________):
         pic.putpixel((x, y),(0,0,0))
```

Using if-statements



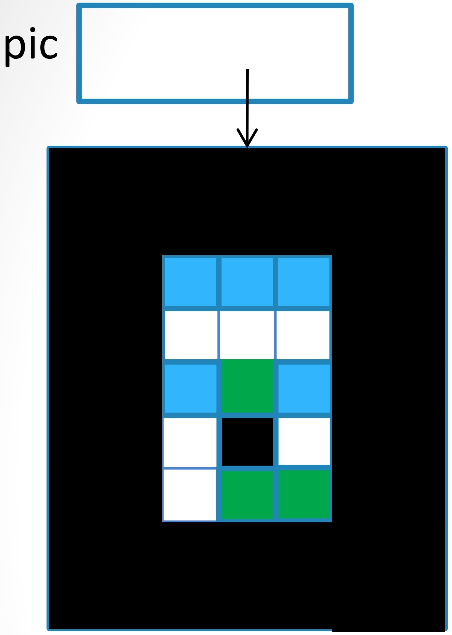
Fill in the code below to create a black border 2 pixels wide around the border of a picture. Assume the picture is at least 2 pixels wide and tall.

Is the if conditional correct?

A. Yes
B. No

```
for x in range(pic.size[0]):
    for y in range(pic.size[1]):

        if ( y <2 or y> (pic.size[1]-2) ):
           pic.putpixel((x, y),(0,0,0))
```

Using if-statements



Fill in the code below to create a black border 2 pixels wide around the border of a picture. Assume the picture is at least 2 pixels wide and tall.

Is the if conditional correct?

A. Yes
B. No

```
for x in range(pic.size[0]):
   for y in range(pic.size[1]):

      if ( y <2 or y> (pic.size[1]-2) or x<2 or
           x>(pic.size[0]-2) ):
         pic.putpixel((x, y),(0,0,0))
```

What have we done with pictures so far?

A. Modified pixel colors to constant values?

B. Modified pixel colors based on the pixel coordinates?

C. Modified pixels by copying color value across pixels?

Geometric Transformations

The key to (almost) all of the image manipulation problems in lab is to copy the color value across pixels in an image. The key is figuring out *which pixels* to copy and *where to copy them to.*

Here is the generic template that you will use for almost all of these problems:

```
for x in range(_____________________________):
   for y in range(_____________________________):
      fromX =
      fromY =
      (newRed,newGreen,newBlue) = pic.getpixel((fromX,fromY))
      pic.putpixel((x, y),(newRed,newGreen,newBlue))
```

Geometric Transformations

The key to (almost) all of the image manipulation problems in lab is to copy the color value across pixels in an image. The key is figuring out *which pixels* to copy and *where to copy them to.*

Here is the generic template that you will use for almost all of these problems:

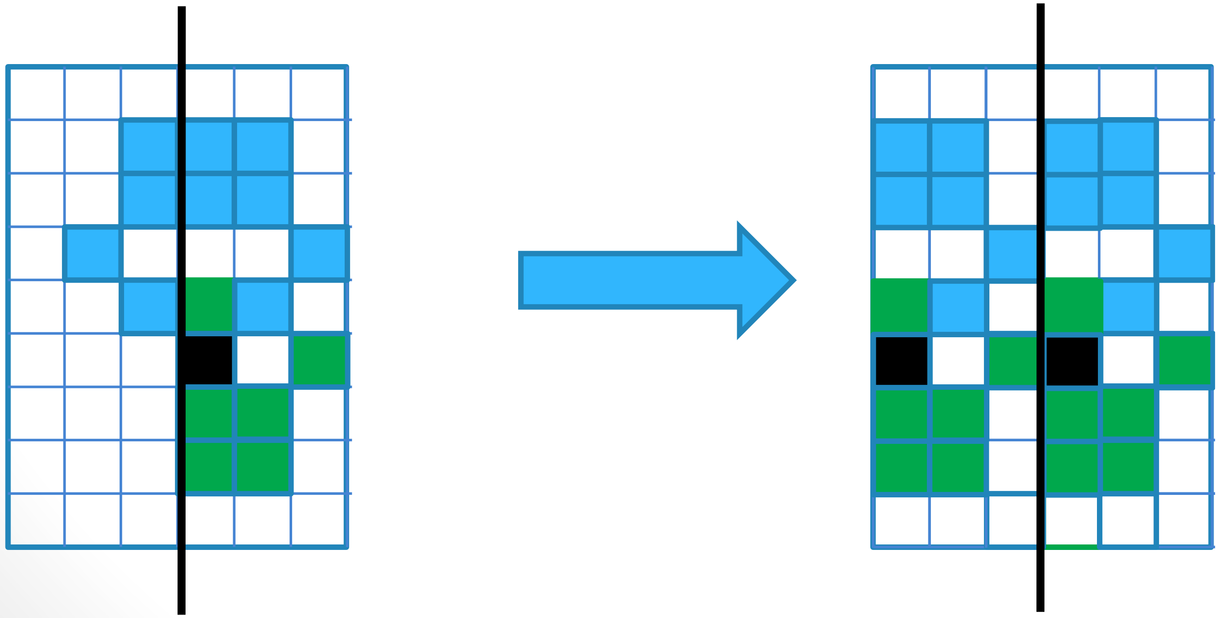
```
for x in range(____________________________):
   for y in range(____________________________):
      fromX =
      fromY =
      (newRed,newGreen,newBlue) = pic.getpixel((fromX,fromY))

      pic.putpixel((x, y),(newRed,newGreen,newBlue))
```

Notice we are copying the color of a pixel over from one location over to another! What is the source pixel? What is the destination pixel?

Copying right to left

Write a function that copies the right half of a picture to the left half

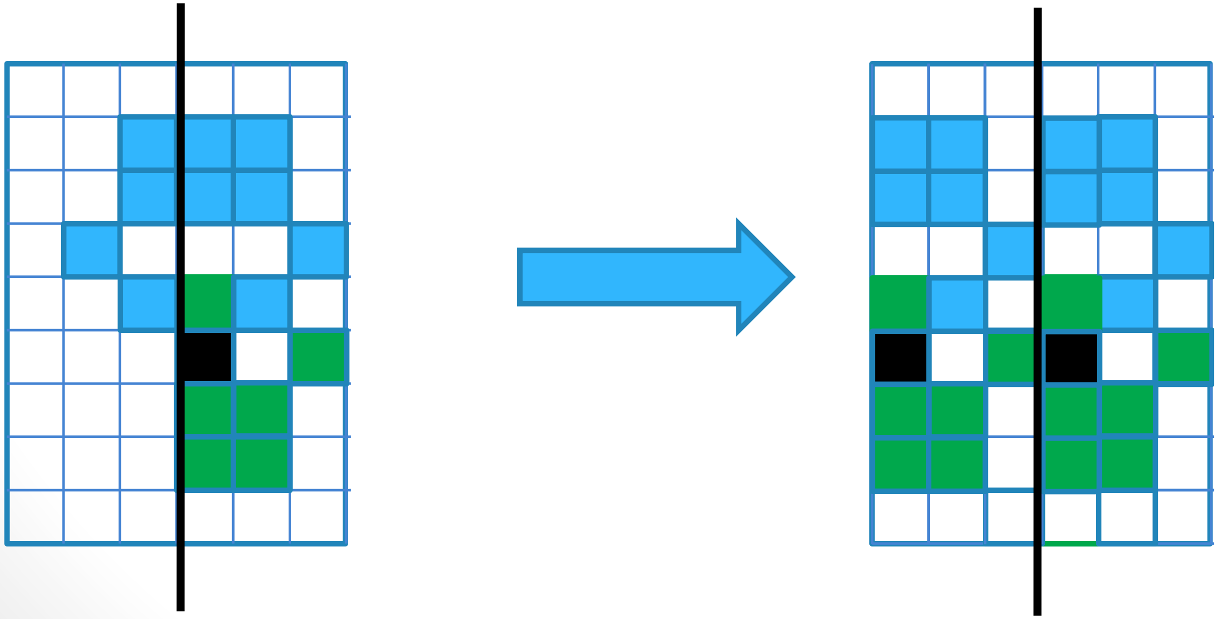
1. Figure out the bounds of your for-loop. x and y will be you loop control variables, x and y are also the coordinates of the destination pixel.
2. Figure out how to represent x_from and y_from (the coordinates of the source pixel) in terms of x and y
3. Fill in the template



Copying right to left

Write a function that copies the right half of a picture to the left half

1. Figure out the bounds of your for-loop. x and y will be you loop control variables, x and y are also the coordinates of the destination pixel.

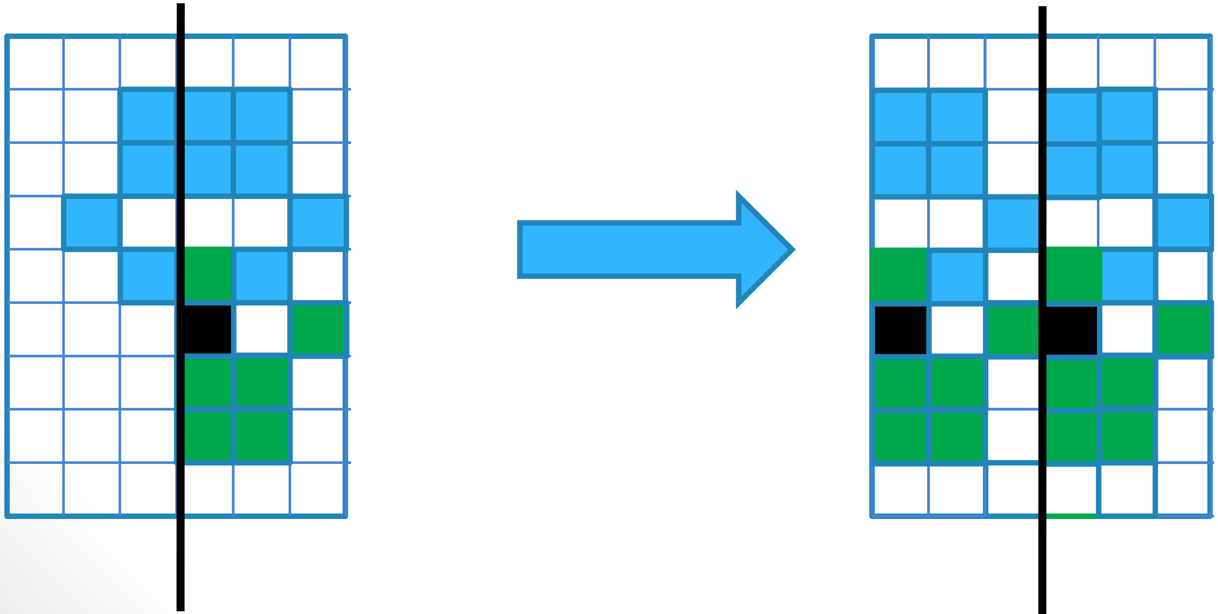


Copying right to left

Write a function that copies the right half of a picture to the left half

1. Figure out the bounds of your for-loop. x and y will be you loop control variables, x and y are also the coordinates of the destination pixel.

```
for x in range(pic.size[0]/2 ):
   for y in range(pic.size[1]):
      fromX =
      fromY =
      (newRed,newGreen,newBlue) = pic.getpixel((fromX,fromY))
      pic.putpixel((x, y),(newRed,newGreen,newBlue))
```

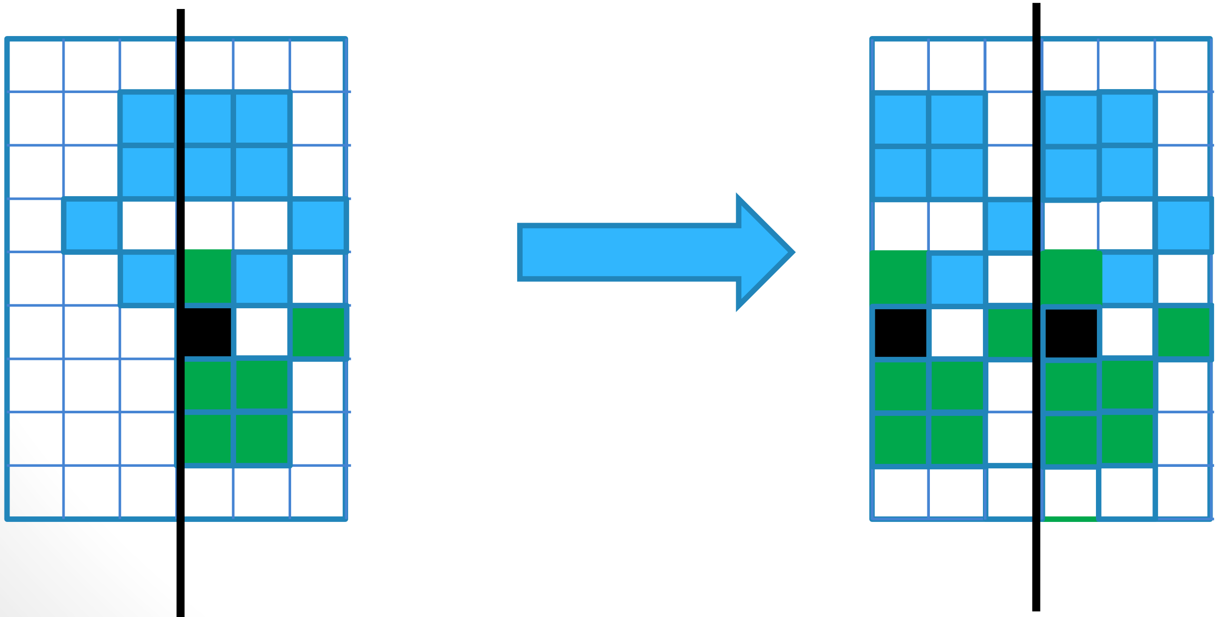


Copying right to left

Write a function that copies the right half of a picture to the left half

2. How does y_from relate to y?
How does x_from relate to x?

```
for x in range(pic.size[0]/2 ):
    for y in range(pic.size[1]):
        fromX =
        fromY =
        (newRed,newGreen,newBlue) = pic.getpixel((fromX,fromY))
        pic.putpixel((x, y),(newRed,newGreen,newBlue))
```

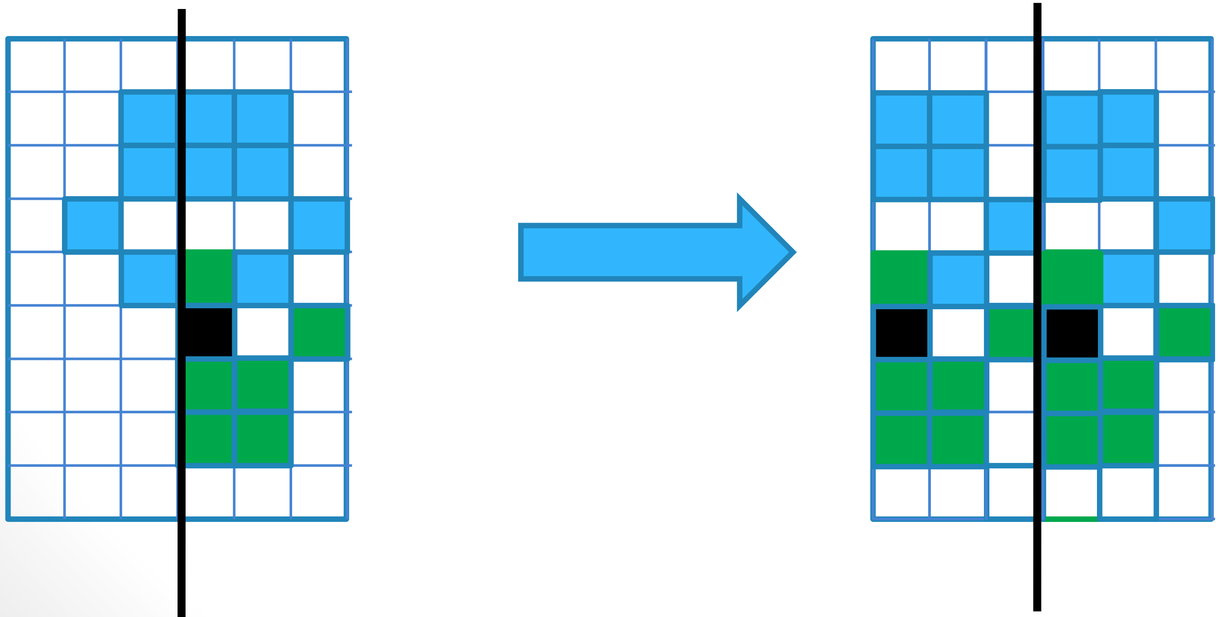


Copying right to left

Write a function that copies the right half of a picture to the left half

2. How does y_from relate to y?
   How does x_from relate to x?

```
for x in range(pic.size[0]/2 ):
   for y in range(pic.size[1]):
      fromX = x+ pic.size[0]/2
      fromY = y
      (newRed,newGreen,newBlue) = pic.getpixel((fromX,fromY))
      pic.putpixel((x, y),(newRed,newGreen,newBlue))
```



Transformation: mystery1

What does the following geometric transformation do?

```
def mystery1(pic, N):
 for x in range(pic.size[0]):
  for y in range(pic.size[1]-N):
   fromX = x
   fromY = y+N
   (newRed,newGreen,newBlue) = pic.getpixel((fromX,fromY))
   pic.putpixel((x, y),(newRed,newGreen,newBlue))
```

A. Shift the whole image up by N pixels
B. Shift the whole image down by N pixels
C. None of the above.

Transformation: mystery2

What does the following geometric transformation do?

```
def mystery2(pic, N):
  for x in range(pic.size[0]):
    for y in range(N, pic.size[1]):
      fromX = x
      fromY = y-N
      (newRed,newGreen,newBlue) = pic.getpixel((fromX,fromY))
      pic.putpixel((x, y),(newRed,newGreen,newBlue))
```

A. Shift the whole image up by N pixels
B. Shift the whole image down by N pixels
C. None of the above.

Transformation: shift down

How do we shift the entire image down by N pixels?

```
def shiftDown(pic,N):
  for x in range(____________________________):
    for y in range(____________________________):
      fromX =
      fromY =
      (newRed,newGreen,newBlue) = pic.getpixel((fromX,fromY))
      pic.putpixel((x, y),(newRed,newGreen,newBlue))
```

Transformation: shift down

Key: don’t overwrite pixels we later need to copy

```
def shiftDown(pic, N):
  for x in range(pic.size[0]):
    for y in range(pic.size[1]-1, N, -1):
      fromX = x
      fromY = y-N
      (newRed,newGreen,newBlue) = pic.getpixel((fromX,fromY))
      pic.putpixel((x, y),(newRed,newGreen,newBlue))
```

Be careful about using the template, you still need to reason about the correctness of your solution