Review

Mutable data

```
def swap(L2, i1, i2):
   temp = L2[i1]
   L2[i1] = L2[i2]
   L2[i2] = temp

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

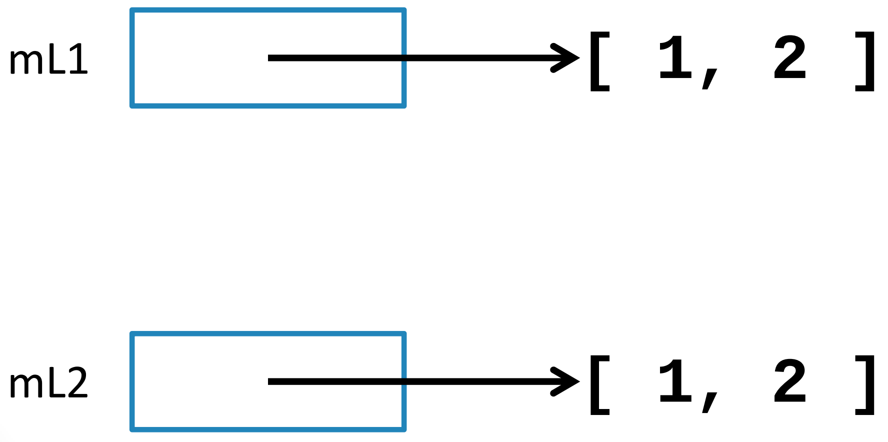
Immutable data

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

x = 5
y = 10
(x,y) = swap(x, y)
print(x, y)
```

Reassignment vs. Data Mutation

```
mL1 = [1,2]
mL1 = [3,4]


print(mL1)
```

```
mL2 = [1,2]
mL2[0] = 3
mL2[1] = 4

print(mL2)
```



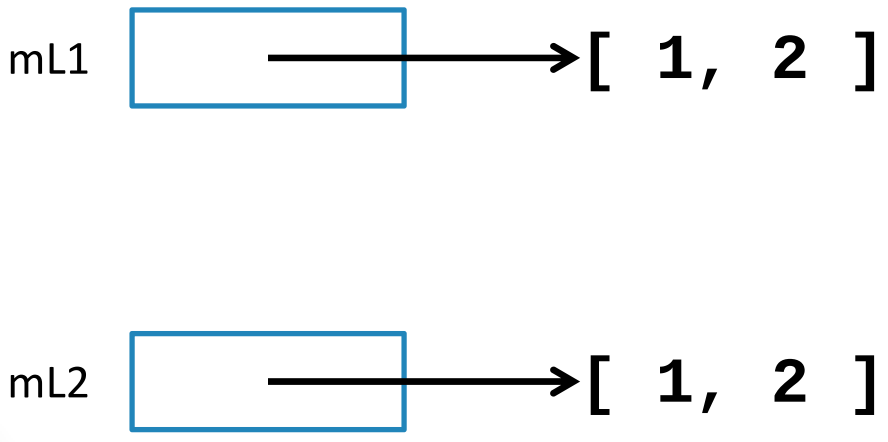
What do you think about this code?

A. It prints different results
B. It prints the same results

Reassignment vs. Data Mutation

```
mL1 = [1,2]
L1 = mL1
mL1 = [3,4]

print(mL1)
print(L1)
```

```
mL2 = [1,2]
L2 = mL2
mL2[0] = 3
mL2[1] = 4

print(mL2)
print(L2)
```



What do you think about this code?

A. It prints different results

B. It prints the same results

Reassignment vs. Data Mutation

```
def process(x):
    x = [3,4]
    x[1] = 5
    return x


L1 = [1,2]
process(L1)
print(L1)
```

What does this print?

A. [1,2]
B. [3,4]
C. [3,5]
D. [1,5]
E. Something else

Reassignment vs. Data Mutation

```
def process(x):
    x[1] = 5
    x = [3,4]
    x[0] = 8


L1 = [1,2]
process(L1)
print(L1)
```

What does this print?

A. [1,2]
B. [8,4]
C. [8,5]
D. [1,5]
E. Something else

Reassignment vs. Data Mutation

```
def inc1(x):
    x = x + 1

def inc2(x):
    x[0] = x[0] + 1

L1 = [1,2]
inc1(L1[0])
inc2(L1)
print(L1)
```

What does this print?

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

Reassignment vs. Data Mutation

```
def inc1(x):
    x = x + 1

def inc2(x):
    x[0] = x[0] + 1

L1 = [1,2]
L2 = [3,6]
L1[1] = L2
print(L1)
inc2(L1)
inc2(L2)
print(L1[1][0])
```

What does this print?

A. 2
B. 3
C. 5
D. 6
E. Something else

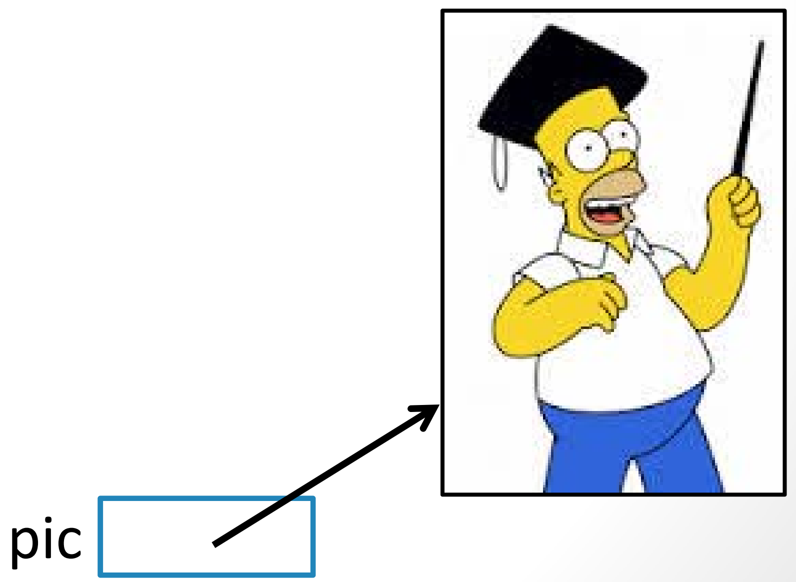
```
def modify(im):
   for x in range(im.size[0]):
     im.putpixel( (x,im.size[1]//2), (0,0,0) )
   return im


pic = Image.open(‘homer.jpg’)
pic2 = modify(pic)
pic.show()
```

What happens?

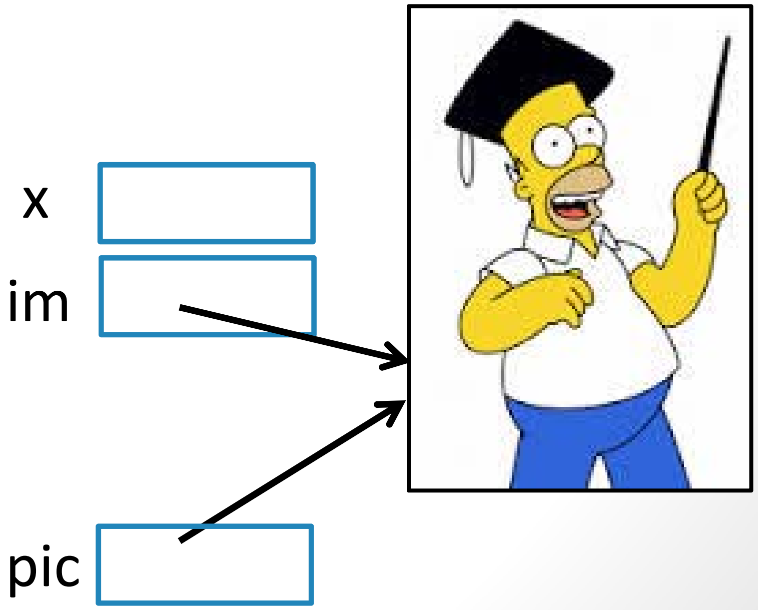
A. You get an error
B. An empty image is shown
C. The original image is shown
D. The modified image is shown
E. Something else

test11

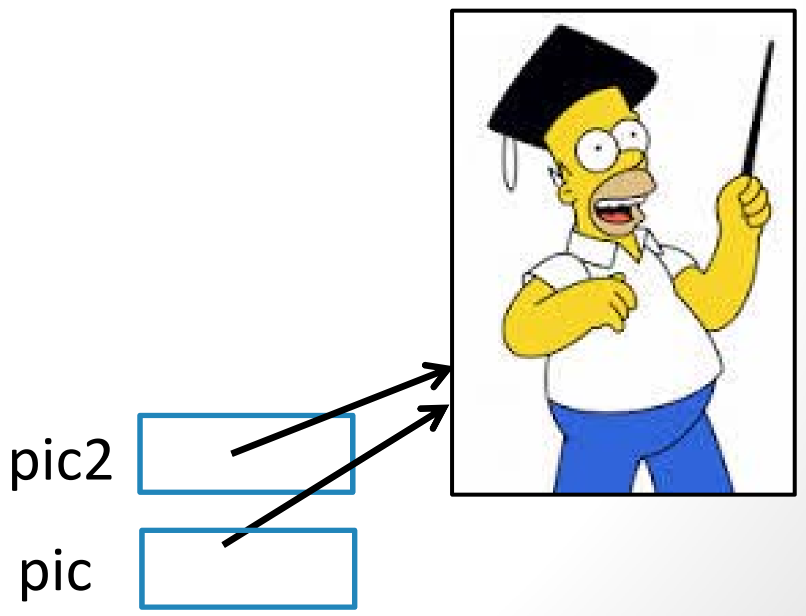
```
def modify(im):
   for x in range(im.size[0]):
     im.putpixel( (x,im.size[1]//2), (0,0,0) )
   return im


pic = Image.open(‘homer.jpg’)
pic2 = modify(pic)
pic.show()
```



```
def modify(im):
   for x in range(im.size[0]):
      im.putpixel( (x,im.size[1]//2), (0,0,0) )
   return im


pic = Image.open('homer.jpg')
pic2 = modify(pic)
pic.show()
```



```
def modify(im):
   for x in range(im.size[0]):
     im.putpixel( (x,im.size[1]//2), (0,0,0) )
   return im


pic = Image.open(‘homer.jpg’)
pic2 = modify(pic)
pic.show()
```



```
def modify(im):
   for x in range(im.size[0]):
     im.putpixel( (x,im.size[1]//2), (0,0,0) )


pic = Image.open('homer.jpg')
pic2 = pic
pic = Image.open('homer.jpg')
modify(pic2)
pic2 = pic
pic2.show()
```

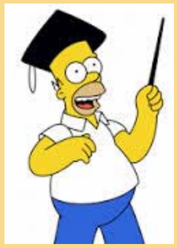
What happens now?

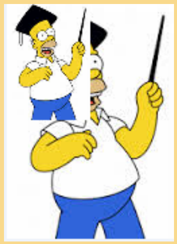
A. You get an error
B. An empty image is shown
C. The original image is shown
D. The modified image is shown
E. Something else

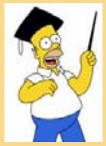
test12

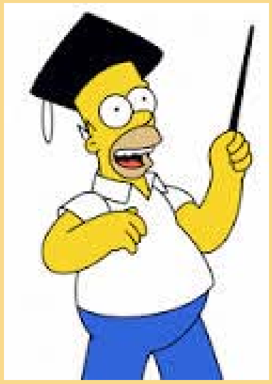
```
from PIL import Image

def resize(im):
    for x in range(im.size[0]//2):
        for y in range(im.size[1]//2):
            (r,g,b) = im.getpixel( (x*2,y*2) )
            im.putpixel( (x,y) , (r,g,b) )

pic1 = Image.open(“homer.jpeg")
pic1.show()
resize(pic1)
pic1.show()
```

The first image that is shown is What is the second?

A. 

B. 

C. 

D. 

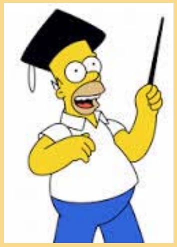
E. Something else

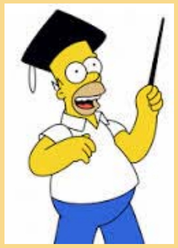
image_resize02

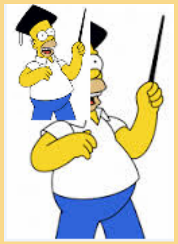
```
from PIL import Image

def resize(im):
    for x in range(im.size[0]):
        for y in range(im.size[1]):
            (r,g,b) = im.getpixel( (x//2,y//2) )
            im.putpixel( (x,y) , (r,g,b) )

pic1 = Image.open(“homer.jpeg")
pic1.show()
resize(pic1)
pic1.show()
```

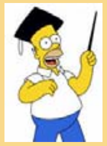
image_resize02

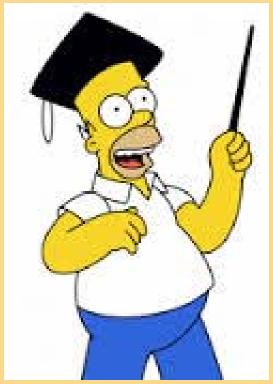
The first image that is shown is What is the second?



A. 

B. 

C. 

D. 

E. Something else

Images and Recursion

```
# The resize function takes an image as an argument
# and returns a new image of half the size
def resize(im):
        …
        return im2
```

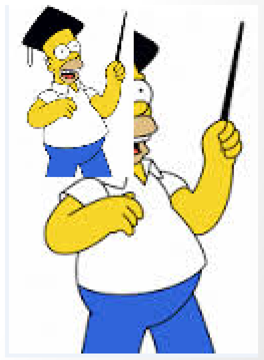
```
# The insert function takes three arguments. The first two
# arguments are images. The second image is inserted into the
# first one (which is therefore modified)
# The location where the insertion happens is given by loc,
# which specifies the quadrant (0 .. 3, counterclockwise)
def insert(im1,im2,loc):
        …
```

image-insert0a

```
# The resize function takes an image as an argument and returns a new image
# of half the size
def resize(im):
        …
        return im2
```

```
# The insert function takes three arguments. The first two arguments are
# images. The second image is inserted into the first one (which is therefore
# modified). The location where the insertion happens is given by loc (quadrant)
def insert(im1,im2,loc):
        …
```

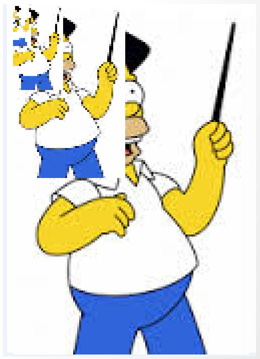
```
def create(im):
    # Complete this function


pic = Image.open("homerprof.jpg")
create(pic)
pic.show()
```



```
# The resize function takes an image as an argument and returns a new image
# of half the size
def resize(im):
        ...
        return im2
```

```
# The insert function takes three arguments. The first two arguments are
# images. The second image is inserted into the first one (which is therefore
# modified). The location where the insertion happens is given by loc (quadrant)
def insert(im1,im2,loc):
        ...
```

```
def create(im, levels):
   # Complete this function


pic = Image.open("homerprof.jpg")
create(pic,5)
pic.show()
```

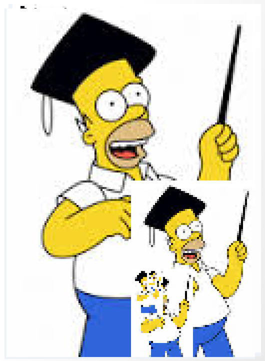


image_recursivelinear1

```
# The resize function takes an image as an argument and returns a new image
# of half the size
def resize(im):
        …
        return im2
```

```
# The insert function takes three arguments. The first two arguments are
# images. The second image is inserted into the first one (which is therefore
# modified). The location where the insertion happens is given by loc (quadrant)
def insert(im1,im2,loc):
        …
```

```
def create(im, levels):
   # Complete this function


pic = Image.open("homerprof.jpg")
create(pic,6)
pic.show()
```



image_recursivespiral1