Multimedia Programming

Image Operations

Most operations built into the library

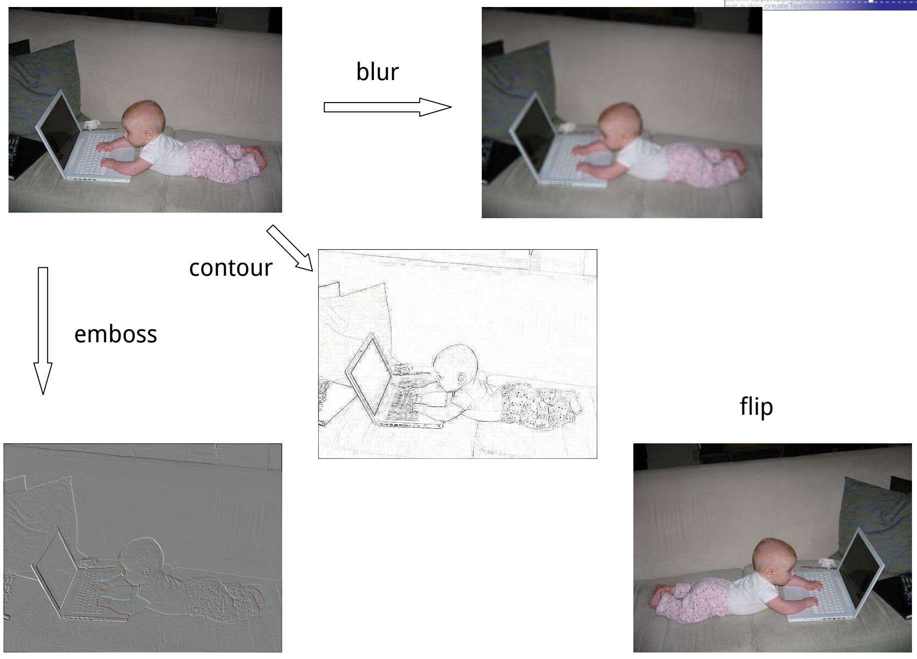
Most operations built into the library

```
import sys
from PIL import Image, ImageFilter

filename = sys.argv[1]
p = Image.open(filename)

p.filter(ImageFilter.BLUR).save('blur-' + filename)
p.filter(ImageFilter.CONTOUR).save('contour-' + filename)
p.filter(ImageFilter.EMBOSS).save('emboss-' + filename)

p.transpose(Image.FLIP_LEFT_RIGHT).save('flip-' + filename)
```



Basic image editing is mostly about coordinates

Basic image editing is mostly about coordinates

```
original = Image.open(sys.argv[1])
x_size, y_size = original.size
x_half, y_half = x_size / 2, y_size / 2
half = original.resize((x_half, y_half))
x_double, y_double = x_size * 2, y_size * 2
double = Image.new('RGB', (x_double, y_double))
for x in range(4):
  for y in range(4):
    box = (x * x_half, y * y_half,
           (x+1) * x_half, (y+1) * y_half)
    double.paste(half, box)
double.save(sys.argv[2])
```

Basic image editing is mostly about coordinates

```
original = Image.open(sys.argv[1])
x_size, y_size = original.size
x_half, y_half = x_size / 2, y_size / 2
half = original.resize((x_half, y_half))
x_double, y_double = x_size * 2, y_size * 2
double = Image.new('RGB', (x_double, y_double))
for x in range(4):
  for y in range(4):
    box = (x * x_half, y * y_half,
           (x+1) * x_half, (y+1) * y_half)
    double.paste(half, box)
double.save(sys.argv[2])
```

Basic image editing is mostly about coordinates

```
original = Image.open(sys.argv[1])
x_size, y_size = original.size
x_half, y_half = x_size / 2, y_size / 2
half = original.resize((x_half, y_half))
x_double, y_double = x_size * 2, y_size * 2
double = Image.new('RGB', (x_double, y_double))
for x in range(4):
  for y in range(4):
    box = (x * x_half, y * y_half,
           (x+1) * x_half, (y+1) * y_half)
    double.paste(half, box)
double.save(sys.argv[2])
```

Basic image editing is mostly about coordinates

```
original = Image.open(sys.argv[1])
x_size, y_size = original.size
x_half, y_half = x_size / 2, y_size / 2
half = original.resize((x_half, y_half))
x_double, y_double = x_size * 2, y_size * 2
double = Image.new('RGB', (x_double, y_double))
for x in range(4):
  for y in range(4):
    box = (x * x_half, y * y_half,
           (x+1) * x_half, (y+1) * y_half)
    double.paste(half, box)
double.save(sys.argv[2])
```

Basic image editing is mostly about coordinates

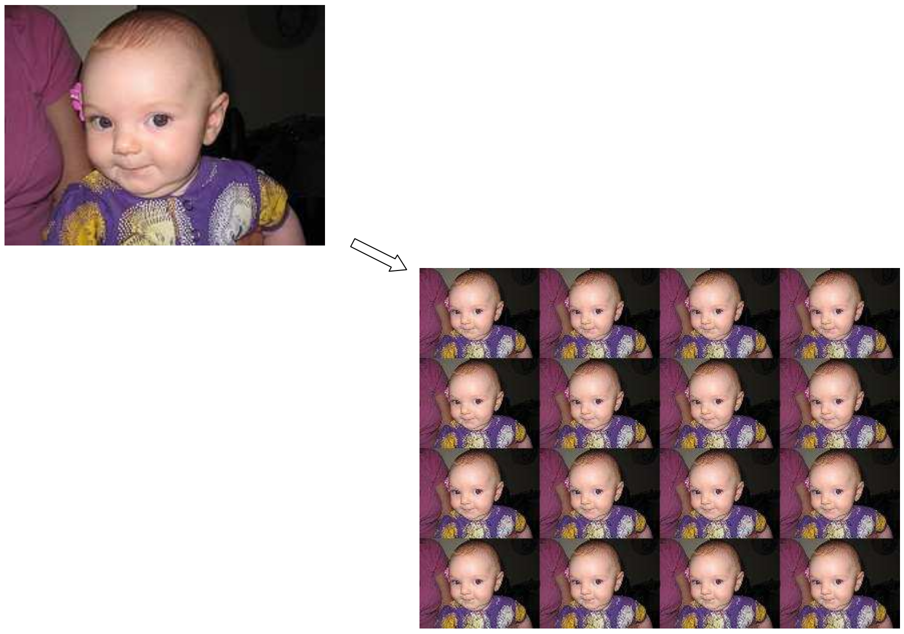
```
original = Image.open(sys.argv[1])
x_size, y_size = original.size
x_half, y_half = x_size / 2, y_size / 2
half = original.resize((x_half, y_half))
x_double, y_double = x_size * 2, y_size * 2
double = Image.new('RGB', (x_double, y_double))
for x in range(4):
  for y in range(4):
    box = (x * x_half, y * y_half,
           (x+1) * x_half, (y+1) * y_half)
    double.paste(half, box)
double.save(sys.argv[2])
```

Basic image editing is mostly about coordinates

```
original = Image.open(sys.argv[1])
x_size, y_size = original.size
x_half, y_half = x_size / 2, y_size / 2
half = original.resize((x_half, y_half))
x_double, y_double = x_size * 2, y_size * 2
double = Image.new('RGB', (x_double, y_double))
for x in range(4):
  for y in range(4):
    box = (x * x_half, y * y_half,
           (x+1) * x_half, (y+1) * y_half)
    double.paste(half, box)
double.save(sys.argv[2])
```

Basic image editing is mostly about coordinates

```
original = Image.open(sys.argv[1])
x_size, y_size = original.size
x_half, y_half = x_size / 2, y_size / 2
half = original.resize((x_half, y_half))
x_double, y_double = x_size * 2, y_size * 2
double = Image.new('RGB', (x_double, y_double))
for x in range(4):
  for y in range(4):
    box = (x * x_half, y * y_half,
           (x+1) * x_half, (y+1) * y_half)
    double.paste(half, box)
double.save(sys.argv[2])
```



Draw on images

```
import sys
from PIL import Image, ImageDraw
BORDER = 10
GRAY = (128, 128, 128)
pic = Image.open(sys.argv[1])
xsize, ysize = pic.size
draw = ImageDraw.Draw(pic)
draw.rectangle((0, 0, xsize, BORDER), fill=GRAY)
draw.rectangle((0, 0, BORDER, ysize), fill=GRAY)
draw.rectangle((0, ysize-BORDER, xsize, ysize), fill=GRAY)
draw.rectangle((xsize-BORDER, 0, xsize, ysize), fill=GRAY)
pic.save('border-' + sys.argv[1])
```

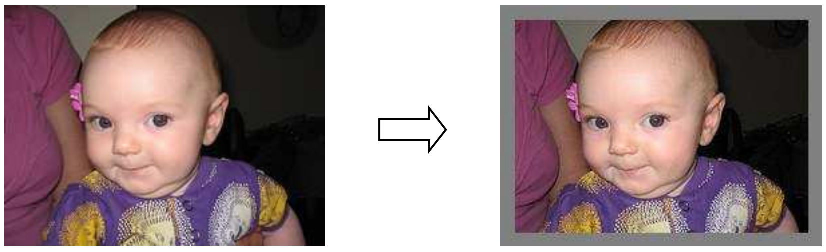
Draw on images

```
import sys
from PIL import Image, ImageDraw
BORDER = 10
GRAY = (128, 128, 128)
pic = Image.open(sys.argv[1])
xsize, ysize = pic.size
draw = ImageDraw.Draw(pic)
draw.rectangle((0, 0, xsize, BORDER), fill=GRAY)
draw.rectangle((0, 0, BORDER, ysize), fill=GRAY)
draw.rectangle((0, ysize-BORDER, xsize, ysize), fill=GRAY)
draw.rectangle((xsize-BORDER, 0, xsize, ysize), fill=GRAY)
pic.save('border-' + sys.argv[1])
```

Draw on images

```
import sys
from PIL import Image, ImageDraw
BORDER = 10
GRAY = (128, 128, 128)
pic = Image.open(sys.argv[1])
xsize, ysize = pic.size
draw = ImageDraw.Draw(pic)
draw.rectangle((0, 0, xsize, BORDER), fill=GRAY)
draw.rectangle((0, 0, BORDER, ysize), fill=GRAY)
draw.rectangle((0, ysize-BORDER, xsize, ysize), fill=GRAY)
draw.rectangle((xsize-BORDER, 0, xsize, ysize), fill=GRAY)
pic.save('border-' + sys.argv[1])
```

Draw on images



Exercise: put frame around entire image

Work with *color bands*

```
import sys
from PIL import Image

filename = sys.argv[1]
pic = Image.open(filename)
pic.load()
bands = pic.split()
for (i, name) in enumerate('rgb'):
  bands[i].save(filename.replace('.', '-%s.' % name))
```

Work with *color bands*

```
import sys
from PIL import Image

filename = sys.argv[1]
pic = Image.open(filename)
pic.load()
bands = pic.split()
for (i, name) in enumerate('rgb'):
  bands[i].save(filename.replace('.', '-%s.' % name))
```

(red[], green[], blue[])

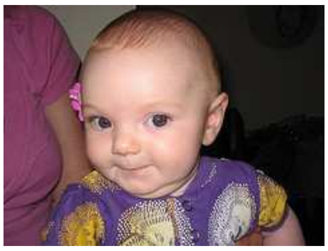
Work with *color bands*

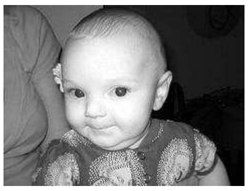
```
import sys
from PIL import Image

filename = sys.argv[1]
pic = Image.open(filename)
pic.load()
bands = pic.split()
for (i, name) in enumerate('rgb'):
  bands[i].save(filename.replace('.', '-%s.' % name))
```

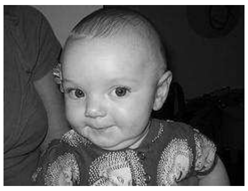
workaround

Work with *color bands*

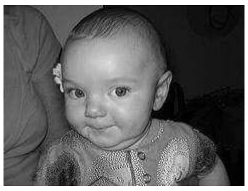




red



green



blue

Use *point functions* to manipulate pixel values

```
R, G, B = 0, 1, 2
SCALE = 0.5

def decrease(x): return x * SCALE

pic = Image.open(sys.argv[1])
pic.load()
bands = pic.split()
bands = (bands[R].point(decrease), bands[G], bands[B])
more_red = Image.merge('RGB', bands)
more_red.save('bluegreen-' + sys.argv[1])
```

Work with *color bands* and *point functions*

```
R, G, B = 0, 1, 2
SCALE = 0.5

def decrease(x): return x * SCALE

pic = Image.open(sys.argv[1])
pic.load()
bands = pic.split()
bands = (bands[R].point(decrease), bands[G], bands[B])
more_red = Image.merge('RGB', bands)
more_red.save('bluegreen-' + sys.argv[1])
```

Work with *color bands* and *point functions*

```
R, G, B = 0, 1, 2
SCALE = 0.5

def decrease(x): return x * SCALE

pic = Image.open(sys.argv[1])
pic.load()
bands = pic.split()
bands = (bands[R].point(decrease), bands[G], bands[B])
more_red = Image.merge('RGB', bands)
more_red.save('bluegreen-' + sys.argv[1])
```

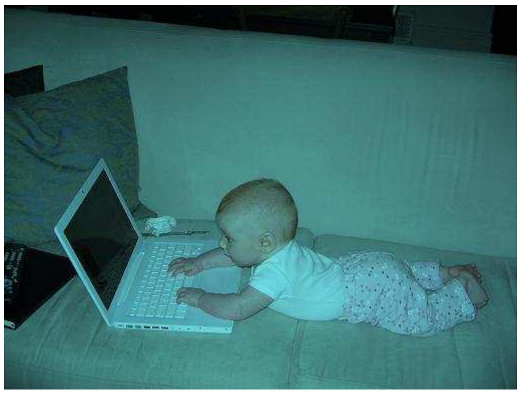
Work with *color bands* and *point functions*

```
R, G, B = 0, 1, 2
SCALE = 0.5

def decrease(x): return x * SCALE

pic = Image.open(sys.argv[1])
pic.load()
bands = pic.split()
bands = (bands[R].point(decrease), bands[G], bands[B])
less_red = Image.merge('RGB', bands)
less_red.save('bluegreen-' + sys.argv[1])
```

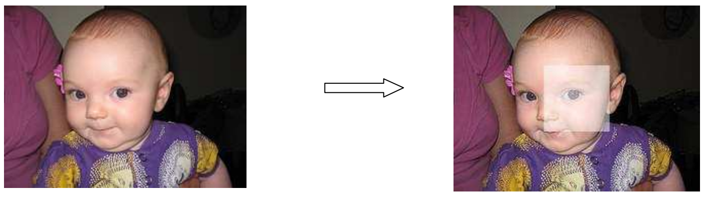
Less red makes the image look more blue/green



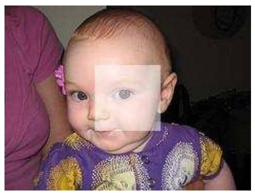
Less red makes the image look more blue/green

What happens if you increase blue and green instead?

Highlight a region in an image

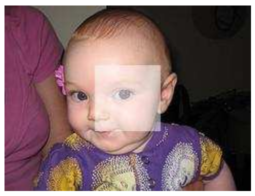


Highlight a region in an image



Option 1: recolor the pixels

Highlight a region in an image

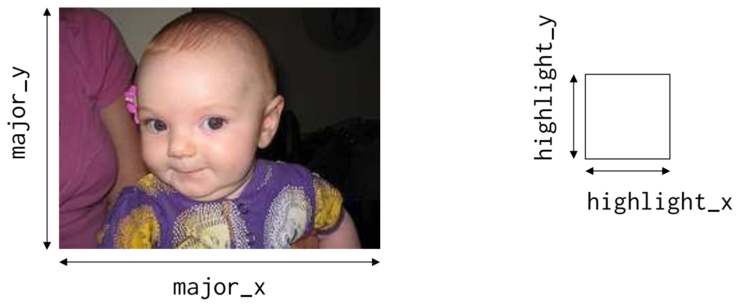


Option 1: recolor the pixels

Option 2: *blend* with a square of desired size

– New pixel = (left pixel + right pixel) / 2

Figure out the coordinates



Low x = (major_x / 2) – (highlight_x / 2)

= (major_x – highlight_x) / 2

High x = (major_x / 2) + (highlight_x / 2)

= (major_x + highlight_x) / 2

```
BLEND = 0.5

major_name, highlight_name = sys.argv[1:3]
major, major_x, major_y = get(major_name)
highlight, highlight_x, highlight_y = get(highlight_name)

box = ((major_x - hl_x) / 2, (major_y - hl_y) / 2,
       (major_x + hl_x) / 2, (major_y + hl_y) / 2)

middle = major.crop(box)
middle = Image.blend(middle, highlight, BLEND)
major.paste(middle, box)
major.save('higlight-' + major_name)
```

```
BLEND = 0.5

major_name, highlight_name = sys.argv[1:3]
major, major_x, major_y = get(major_name)
highlight, highlight_x, highlight_y = get(highlight_name)

box = ((major_x - hl_x) / 2, (major_y - hl_y) / 2,
       (major_x + hl_x) / 2, (major_y + hl_y) / 2)

middle = major.crop(box)
middle = Image.blend(middle, highlight, BLEND)
major.paste(middle, box)
major.save('higlight-' + major_name)
```

```
BLEND = 0.5

major_name, highlight_name = sys.argv[1:3]
major, major_x, major_y = get(major_name)
highlight, highlight_x, highlight_y = get(highlight_name)

box = ((major_x - hl_x) / 2, (major_y - hl_y) / 2,
       (major_x + hl_x) / 2, (major_y + hl_y) / 2)

middle = major.crop(box)
middle = Image.blend(middle, highlight, BLEND)
major.paste(middle, box)
major.save('higlight-' + major_name)
```

```
BLEND = 0.5

major_name, highlight_name = sys.argv[1:3]
major, major_x, major_y = get(major_name)
highlight, highlight_x, highlight_y = get(highlight_name)

box = ((major_x - hl_x) / 2, (major_y - hl_y) / 2,
       (major_x + hl_x) / 2, (major_y + hl_y) / 2)

middle = major.crop(box)
middle = Image.blend(middle, highlight, BLEND)
major.paste(middle, box)
major.save('higlight-' + major_name)
```

```
BLEND = 0.5

major_name, highlight_name = sys.argv[1:3]
major, major_x, major_y = get(major_name)
highlight, highlight_x, highlight_y = get(highlight_name)

box = ((major_x - hl_x) / 2, (major_y - hl_y) / 2,
       (major_x + hl_x) / 2, (major_y + hl_y) / 2)

middle = major.crop(box)
middle = Image.blend(middle, highlight, BLEND)
major.paste(middle, box)
major.save('higlight-' + major_name)
```

```
BLEND = 0.5

major_name, highlight_name = sys.argv[1:3]
major, major_x, major_y = get(major_name)
highlight, highlight_x, highlight_y = get(highlight_name)

box = ((major_x - hl_x) / 2, (major_y - hl_y) / 2,
       (major_x + hl_x) / 2, (major_y + hl_y) / 2)

middle = major.crop(box)
middle = Image.blend(middle, highlight, BLEND)
major.paste(middle, box)
major.save('higlight-' + major_name)
```

PIL provides basic image processing

PIL provides basic image processing

OpenCV (http://opencv.willowgarage.com) is a complete image processing library

PIL provides basic image processing

OpenCV (http://opencv.willowgarage.com) is a complete image processing library

Have to convert images...

PIL provides basic image processing

OpenCV (http://opencv.willowgarage.com) is a complete image processing library

Have to convert images...

...but it's worth it

created by

Greg Wilson

November 2010