Multimedia Programming

Images

Pictures are much older than text

Just as easy to work with...

...given the right libraries

Explore the Python Imaging Library (PIL)

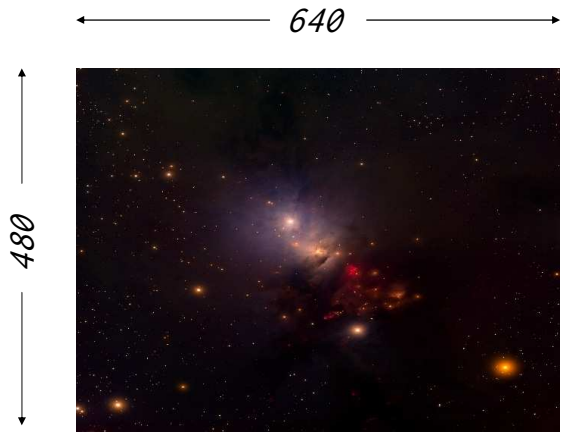
Other languages have similar tools

Start by loading the image into memory

```
>>> from PIL import Image
>>> pic = Image.open('ngc1333-noao.jpg')
```

Examine image properties

```
>>> pic.format
'JPEG'
>>> pic.size
(640, 480)
```

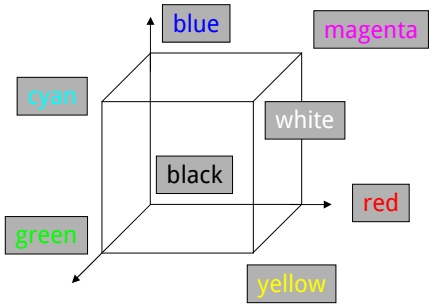


Colors represented by red-green-blue (RGB) triples

Black is (0, 0, 0)

White is (255, 255, 255) or (0xFF, 0xFF, 0xFF)

A *color cube*

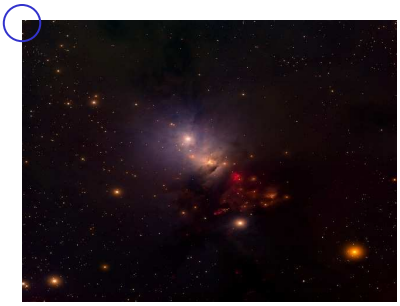


Pixel coordinates are (x, y) tuples

(0, 0) is the upper left corner of the image

– Because that's how old CRT monitors drew things

```
>>> pic.getpixel((0, 0)) # upper left corner
(17, 12, 18)
```



Find the brightest pixel

Find the brightest pixel ← What does "brightest" actually mean?

Find the brightest pixel

```
>>> xsize, ysize = pic.size
>>> bx, by, max_val = 0, 0, 0
>>> for x in range(xsize):
...   for y in range(ysize):
...     r, g, b = pic.getpixel((x, y))
...     if r + g + b > max_val:
...       bx, by, total = x, y, r + g + b
... print (bx, by), total
...
```


Find the brightest pixel

```
>>> xsize, ysize = pic.size
>>> bx, by, max_val = 0, 0, 0
>>> for x in range(xsize):
...   for y in range(ysize):
...     r, g, b = pic.getpixel((x, y))
...     if r + g + b > max_val:
...       bx, by, total = x, y, r + g + b
... print (bx, by), total
...
```

```
(59, 345) 758
```

How fast is that?

How fast is that?

```
def brightest(picture):
  ...as above...
  return (bx, by), total
```

How fast is that?

```
def brightest(picture):
  ...as above...
  return (bx, by), total

from time import time

def elapsed(func, picture):
   start = time()
   result = func(picture)
   return time() - start, result
```


How fast is that?

```
def brightest(picture):
  ...as above...
  return (bx, by), total

from time import time

def elapsed(func, picture):
   start = time()
   result = func(picture)
   return time() - start, result
```

0.63 seconds

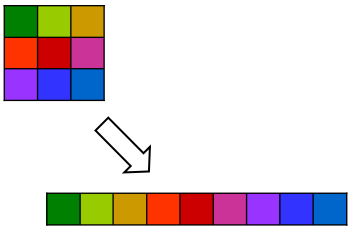
Ignore coordinates

Ignore coordinates

```
def faster(picture):
  max_val = 0
  for (r, g, b) in picture.getdata():
    if r + g + b > max_val:
      max_val = r + g + b
  return max_val
```

Ignore coordinates

```
def faster(picture):
  max_val = 0
  for (r, g, b) in picture.getdata():
    if r + g + b > max_val:
      max_val = r + g + b
  return max_val
```

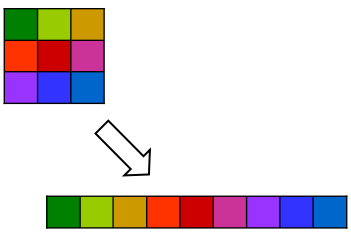


Pixels ordered row by row

Ignore coordinates

0.07 seconds

```
def faster(picture):
  max_val = 0
  for (r, g, b) in picture.getdata():
    if r + g + b > max_val:
      max_val = r + g + b
  return max_val
```

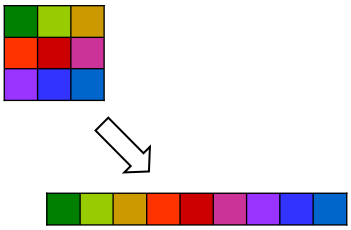


Pixels ordered row by row

Ignore coordinates

```
def faster(picture):
  max_val = 0
  for (r, g, b) in picture.getdata():
    if r + g + b > max_val:
      max_val = r + g + b
  return max_val
```

0.07 seconds



Pixels ordered row by row

Exercise: return (x, y) coordinate of brightest pixel

A useful compromise

A useful compromise

```
def inbetween(picture):
  xsize, ysize = picture.size
  temp = picture.load()
  bx, by, max_val = 0, 0, 0
  for x in range(xsize):
    for y in range(ysize):
      r, g, b = temp[x, y]
      if r + g + b > max_val:
        bx, by, total = x, y, r + g + b
  return (bx, by), total
```


A useful compromise

0.13 seconds

```
def inbetween(picture):
  xsize, ysize = picture.size
  temp = picture.load()
  bx, by, max_val = 0, 0, 0
  for x in range(xsize):
    for y in range(ysize):
      r, g, b = temp[x, y]
      if r + g + b > max_val:
        bx, by, total = x, y, r + g + b
  return (bx, by), total
```

Find stars

Find stars

Convert to black and white

Find stars

Convert to black and white ← Easier to see black on white than vice versa

Find stars

Convert to black and white

```
def monochrome(picture, threshold):
  black = (  0,   0,   0)
  white = (255, 255, 255)
  xsize, ysize = picture.size
  temp = picture.load()
  for x in range(xsize):
    for y in range(ysize):
      r, g, b = temp[x, y]
      if r + g + b >= threshold: temp[x, y] = black
      else:                      temp[x, y] = white
```

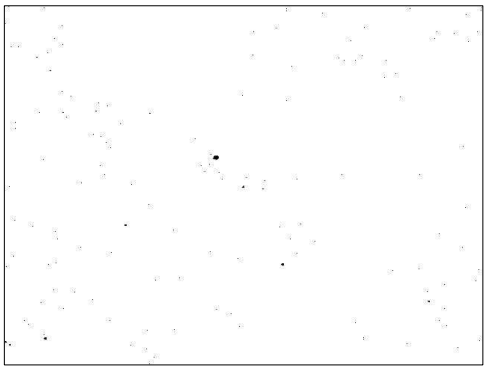
```
if __name__ == '__main__':
  pic = Image.open(sys.argv[1])
  monochrome(pic, 200 + 200 + 200)
  pic.save(sys.argv[2])
```



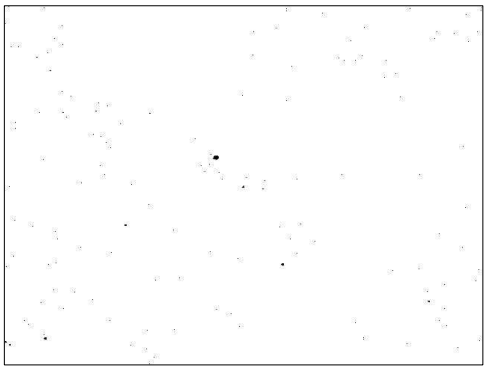
```
if __name__ == '__main__':
  pic = Image.open(sys.argv[1])
  monochrome(pic, 200 + 200 + 200)
  pic.save(sys.argv[2])
```

Not the same as (200, 200, 200)

```
if __name__ == '__main__':
  pic = Image.open(sys.argv[1])
  monochrome(pic, 200 + 200 + 200)
  pic.save(sys.argv[2])
```




```
if __name__ == '__main__':
  pic = Image.open(sys.argv[1])
  monochrome(pic, 200 + 200 + 200)
  pic.save(sys.argv[2])
```

← Now we can start counting

created by

Greg Wilson

November 2010