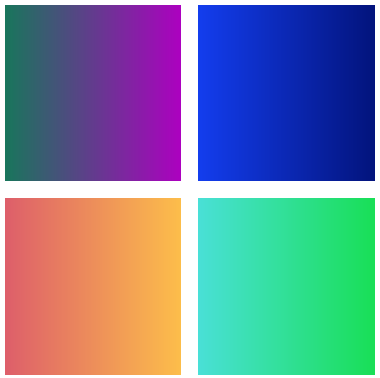
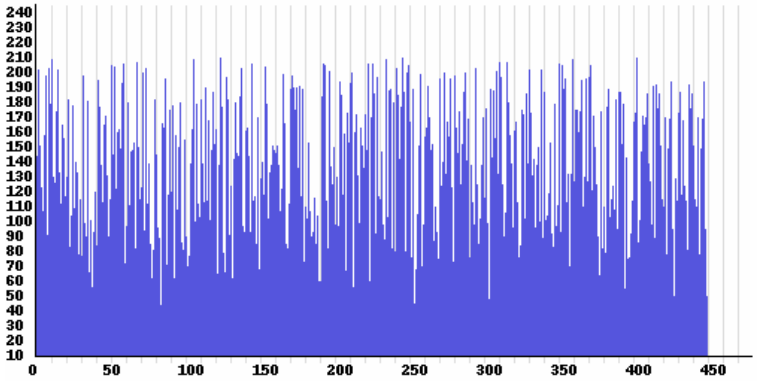




Random and Dynamic Images using Python CGI

Steven Kryskalla
September 14 2005
skryskalla@gmail.com
http://lost-theory.org

Overview

- Intro
- Simple Python CGI
- Random Image Script
- Python's Graphics Libraries (PIL)
- Dynamic Image Script
- Slightly more advanced Python CGI
- A small surprise…

Intro

- CGI Programming with Python
- Python Graphics Libraries (PIL)

- Goals:
  - Create a random image script
  - Create a dynamic image script
  - Learn about Python's features as a web scripting language
  - Learn a little about Python's graphics capabilities

Simple Python CGI

- Like any other language
- Generate content from scripts

- Python's built in CGI test server:

```
python -c "import CGIHTTPServer;CGIHTTPServer.test()"
```

- First goal: write a random image script
- First step: redirecting a static image to the browser
  - Duplicate the behavior of a normal server

Simple Python CGI

- Set the Content-type
- Pipe image to browser: one

```
if __name__ == "__main__":
    print "Content-type: image/png\n"
    print file(r"c:\python\random_img\1.png", "rb").read()
```

Simple Python CGI

- What if we are serving multiple file types?
- Lookup the file's extension to get the content-type


```
ext2conttype = {"jpg": "image/jpeg",
                "jpeg": "image/jpeg",
                "png": "image/png",
                "gif": "image/gif"}

def content_type(filename):
        return ext2conttype[filename[filename.rfind(".")+1:].lower()]

if __name__ == "__main__":
        imgpath = r"c:\python\random_img\3.jpg"
        print "Content-type: %s\n" % (content_type(imgpath))
        print file(imgpath, "rb").read()
```

Random Image Script

- We can serve up an image file
- We can find the content type of any image

- Pick a random image
  - `os.listdir()`
  - Filter out the non-images
  - `random.choice()` on the remaining images

Random Image Script

```
from os import listdir
from random import choice
ext2conttype = {"jpg": "image/jpeg",
                "jpeg": "image/jpeg",
                "png": "image/png",
                "gif": "image/gif"}

def content_type(filename):
        return ext2conttype[filename[filename.rfind(".")+1:].lower()]

def isimage(filename):
        """true if filename's extension is in content-type lookup"""
        filename = filename.lower()
        return filename[filename.rfind(".")+1:] in ext2conttype

def random_file(dir):
        """returns the filename of a randomly chosen image in dir"""
        images = [f for f in listdir(dir) if isimage(f)]
        return choice(images)

if __name__ == "__main__":
        dir = "c:\\python\\random_img\\"
        r = random_file(dir)
        print "Content-type: %s\n" % (content_type(r))
        print file(dir+r, "rb").read()
```

Random Image Script

- Not that useful…
- Improvements, etc.
  - Serve multiple directories of images
  - Randomly serve other kinds of files
  - Use as a starting point for a dynamic image script

Python's Graphics Libraries

- Many to choose from:
  - PIL
  - Agg
  - Gnuplot
  - ImageMagick
  - …
- PIL
- Is there more interest in this?

PIL

- www.pythonware.com/products/pil
- Docs available on website

- First get acquainted with basic PIL functionality

PIL

- `Image`
- Image objects handle input, output, and information about PIL images
- Opening an image:
  - `Image.open(filename)`
  - `Image.open("test.png")`
- Create a new image:
  - `Image.new(mode, size, color)`
  - `Image.new("RGB", (300,300), fill="#000")`
- Save an image
  - `Img.save(filename, format)`
  - `Img.save("out.png", "PNG")`

PIL

- `ImageDraw`
- Create a draw object:
  - `draw = ImageDraw.Draw(img)`
- Draw object provides simple 2d graphics
  - `draw.line(xy, options)`
  - `draw.line((0,0,300,300), fill="#00F")`
  - `draw.text(xy, str, options)`
  - `draw.text((50,50), "Hello World", fill=(50,50,150))`

PIL

- Create a dynamic image script
- No CGI yet
- Creates a random gradient

PIL

```
import Image,ImageDraw
from random import randint as rint

def randgradient():
    img = Image.new("RGB", (300,300), "#FFFFFF")
    draw = ImageDraw.Draw(img)

    r,g,b = rint(0,255), rint(0,255), rint(0,255)
    dr = (rint(0,255) - r)/300.
    dg = (rint(0,255) - g)/300.
    db = (rint(0,255) - b)/300.
    for i in range(300):
        r,g,b = r+dr, g+dg, b+db
        draw.line((i,0,i,300), fill=(int(r),int(g),int(b)))
        img.save("img.png", "PNG")

if __name__ == "__main__":
     randgradient()
```

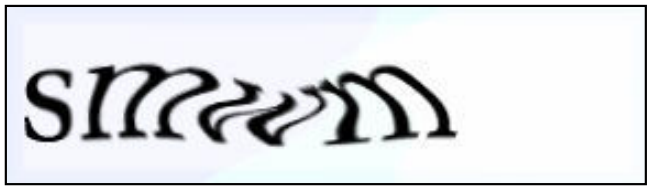
Dynamic Image Script

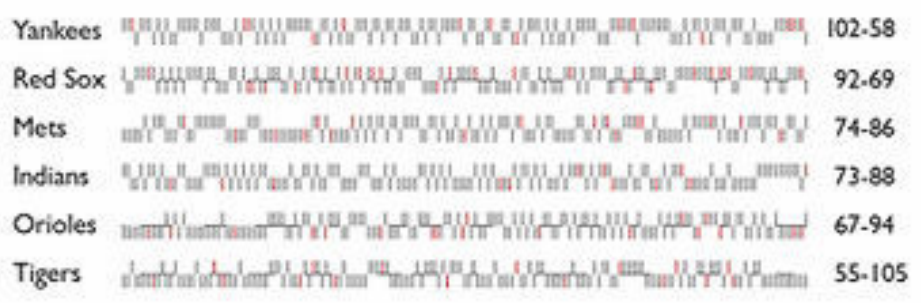
- Now add CGI
- Don’t write to a file
- Use `cStringIO` for a file-like object

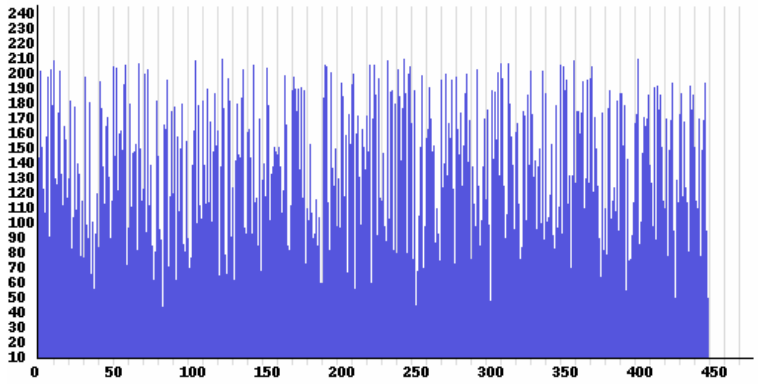
```
  import Image,ImageDraw
+ import cStringIO
  from random import randint as rint

  def randgradient():
      img = Image.new("RGB", (300,300), "#FFFFFF")
      draw = ImageDraw.Draw(img)
      r,g,b = rint(0,255), rint(0,255), rint(0,255)
      dr = (rint(0,255) - r)/300.
      dg = (rint(0,255) - g)/300.
      db = (rint(0,255) - b)/300.
      for i in range(300):
          r,g,b = r+dr, g+dg, b+db
          draw.line((i,0,i,300), fill=(int(r),int(g),int(b)))

+     f = cStringIO.StringIO()
+     img.save(f, "PNG")

      print "Content-type: image/png\n"
+     f.seek(0)
      print f.read()

  if __name__ == "__main__":
      randgradient()
```

Slightly More Advanced CGI

- More useful applications of dynamic images:
  - CAPTCHAs
  - Resize images for web photo gallery
  - Sparklines
  - Dynamic graphs of log files







Slightly More Advanced CGI

- Dynamically graph log files
- Accept arguments through the query string (from forms, user input, etc.)
- `cgi` module
  - `cgi.FieldStorage()`

Slightly More Advanced CGI

- Read values out of the FieldStorage
- If a value is not present we can return an error message or use a default value

```
if __name__ == "__main__":
    form = cgi.FieldStorage()
    if "filename" in form:
        if "color" in form:
            graph(form["filename"].value, form["color"].value)
        else:
            graph(form["filename"].value, "55d")
    else:
        print "Content-type: text/html\n"
        print """<html><body>No input file given</body></html>"""
```

Slightly More Advanced CGI

- The script is now usable with forms
- Simple form:

```
<html>
<head><title>Graph Log File</title></head>
<body>

<br>
<form action="/cgi-bin/dynimg3.py">
   <input maxlength="100" size="55" name="filename" value="">
   <select name="color">
      <option value="FF3333">Red
      <option value="33FF33">Green
      <option value="3333FF">Blue
      <option value="33AAAA">Teal
   </select>
   <input type="submit" value="Graph!">
</form>

</body>
</html>
```

Conclusion

- Python's features make it a great CGI language
  - Web frameworks extend its usefulness even more
- Solid image & graphics libraries allow for quick writing of useful visual output

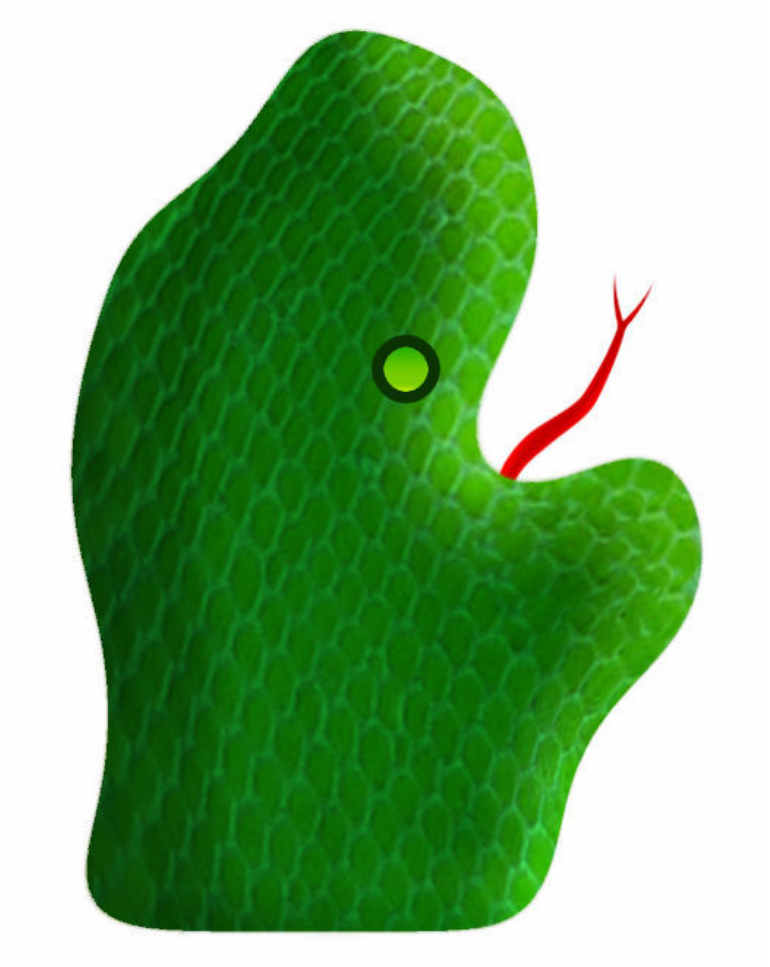
Questions?

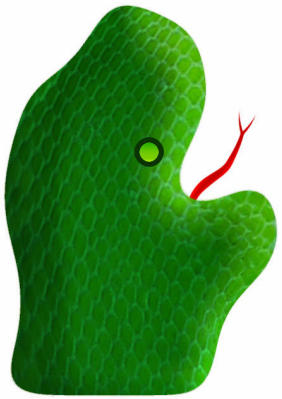
Thank you :]

Now..

A small surprise..

A logo





Michipug

Michigan Python Usergroup