PyYAML, flickrapi and Tkinter

In a Desktop Image Display App

Doug Latornell <djl@douglatornell.ca>

VanPyZ :: 1 December 2009

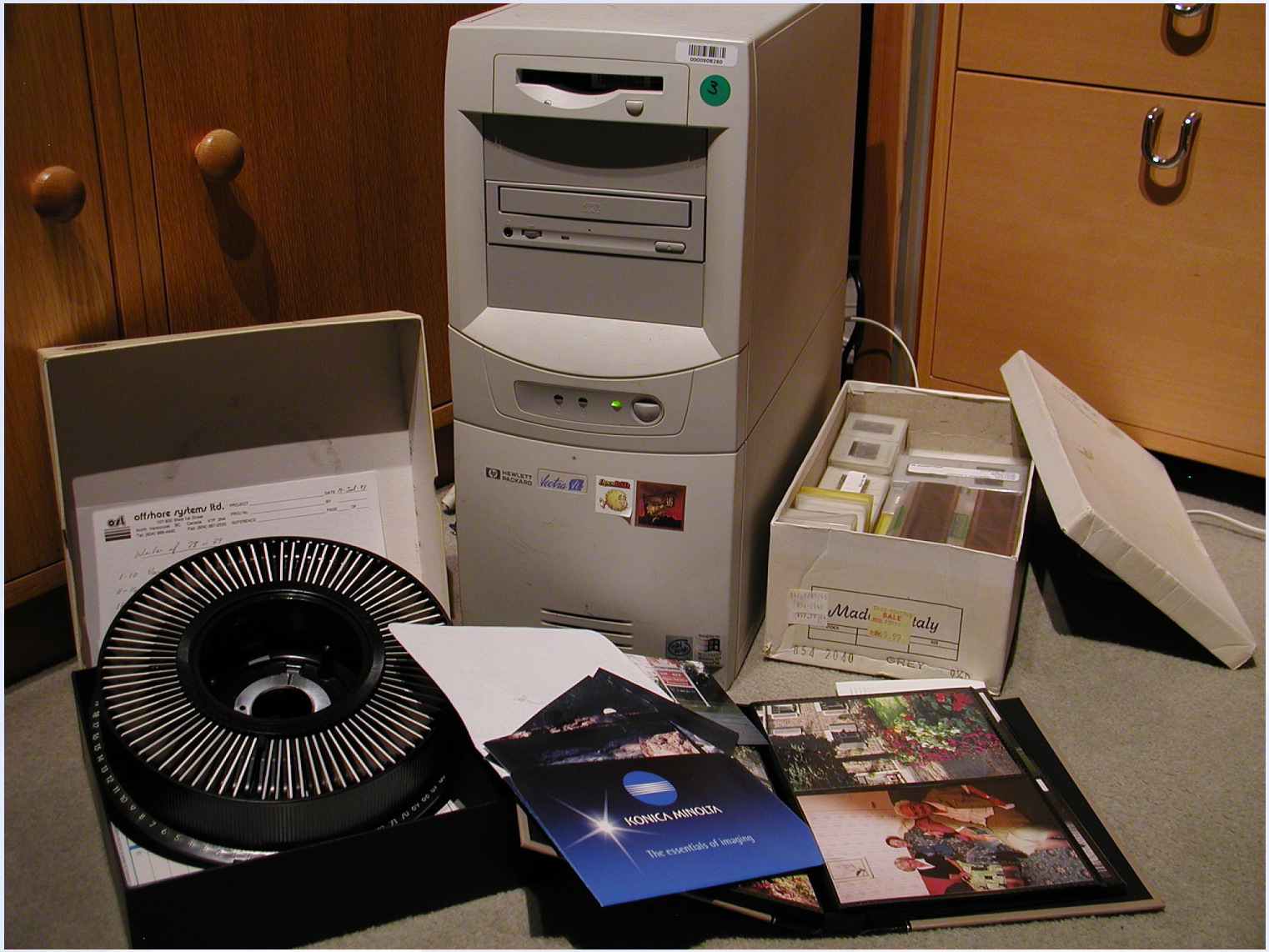
2 Cool Things

- `YAML` as an alternative to `INI` for configuration files
- Access to Flickr from Python

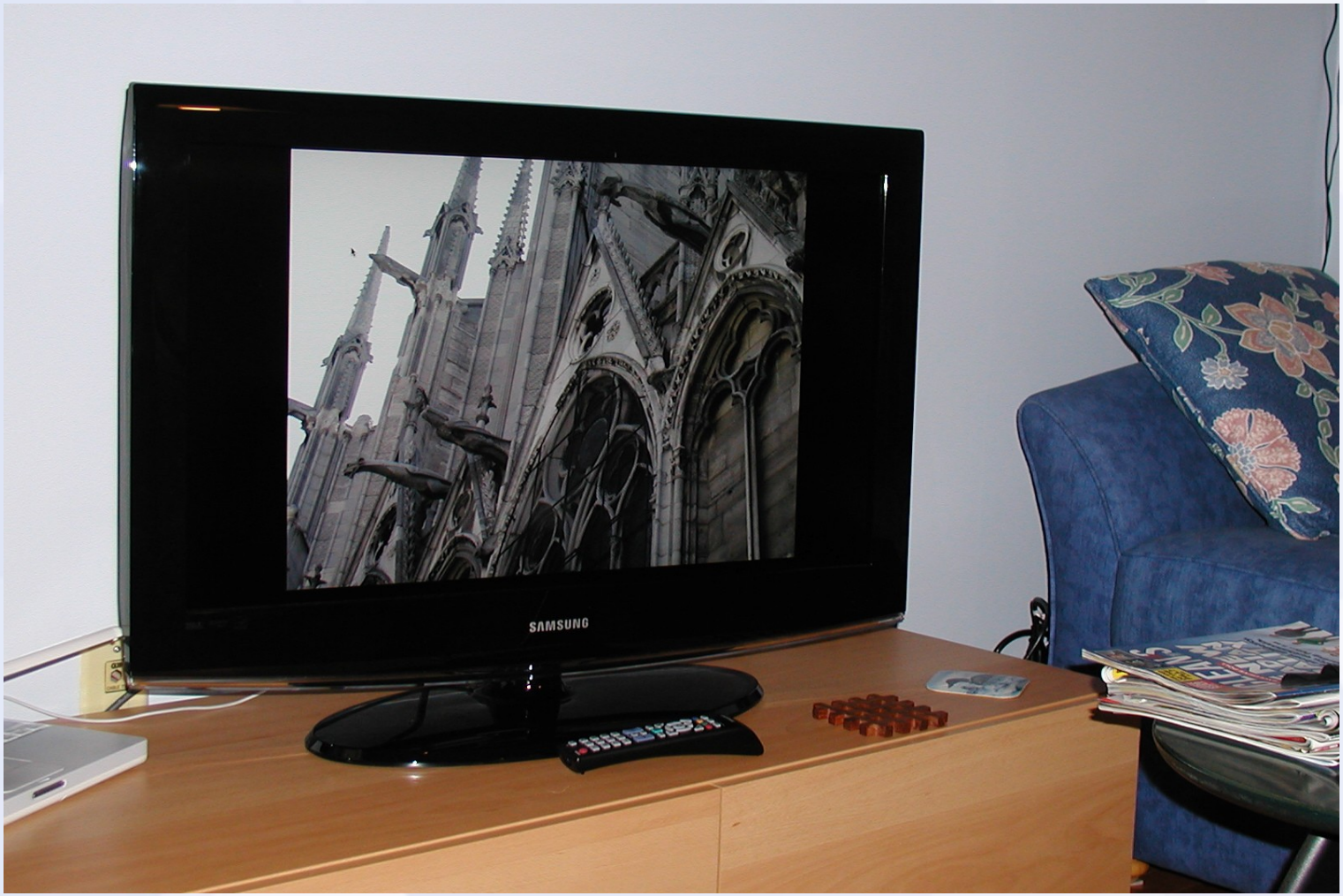
...And 1 Useful One

- `Tkinter`
  - GUI interface toolkit in Python stdlib
  - Cross platform

Dude, Where are the Pictures?



The Big Picture



But How?

- Sequential

  boring...

- Random

  annoyingly... random...

LivingPics

- Display images from local or network connected storage, or Flickr
- Control the randomness and repetitiousness of the display
- Something to do while recovering from 2 broken wrists!

INI Config

```
[Timing]
noshowhrs = 4.0
displaysecs = 10

[Exclusion List]
exclusionlist = ./exclusion_list.txt

[Image Sources]
defaultsource = /Users/doug/Pictures/iPhoto Library/Originals
imagesources = /Users/doug/Pictures/iPhoto Library/Originals, ""
        /Users/doug/Pictures/iPhoto Library/Modified/2006/Big
Island of Hawaii, ""
        /Users/doug/Pictures/iPhoto Library/Originals/2005, ""
        http://www.flickr.com/photos/sada_images/, "Our flickr
Stream"

[Image Selection]
probofjump = 0.05
rangeincr = 20
rangedecr = -10
```

YAML

- YAML Ain't Markup Language
- “a human friendly data serialization standard for all programming languages”

YAML Config

```
exclude_file: exclusions.yaml

img_selection:
  prob_of_jump: 0.05
  range_decr: -10
  range_incr: 20

img_srcs:
  current_src: /Users/doug/Pictures/iPhoto Library/Originals

  stored_srcs:
  - nickname: null
    src: /Users/doug/Pictures/iPhoto Library/Originals
  - nickname: Our flickr Stream
    src: http://www.flickr.com/photos/sada_images/
  - nickname: Spectacular Landscapes
    src: http://www.flickr.com/groups/spectacular_landscapes/

timing:
  display_secs: 10
  no_show_hrs: 4.0
```

So What?

- ~120 lines of ConfigParser code replaced by 20 lines of code using PyYAML
  - For the price of an external dependency
- ~50 lines of list subclass code for the exclusion list replaced by 20 lines of code using PyYAML

LivingPics ConfigMgr

```
class ConfigMgr(object):
    def __init__(self, config_file):
        """Create a ConfigMgr instance and populate its attributes
        from the specified config_file.
        """
        self.config_file = config_file
        with open(config_file, 'r') as fp:
            config = yaml.load(fp)
        for attr in 'timing exclude_file img_selection img_srcs'.split():
            self.__setattr__(attr, config[attr])

    def write(self):
        """Write the current configuration to the config file.
        """
        config = dict()
        for attr in 'timing exclude_file img_selection img_srcs'.split():
            config[attr] = self.__getattribute__(attr)
        with open(self.config_file, 'w') as fp:
            yaml.dump(config, stream=fp, default_flow_style=False)
```

How?

- Everything in an INI file is interpreted as a string, whereas YAML understands types

YAML Config

```
exclude_file: exclusions.yaml

img_selection:
  prob_of_jump: 0.05
  range_decr: -10
  range_incr: 20

img_srcs:
  current_src: /Users/doug/Pictures/iPhoto Library/Originals

  stored_srcs:
  - nickname: null
    src: /Users/doug/Pictures/iPhoto Library/Originals
  - nickname: Our flickr Stream
    src: http://www.flickr.com/photos/sada_images/
  - nickname: Spectacular Landscapes
    src: http://www.flickr.com/groups/spectacular_landscapes/

timing:
  display_secs: 10
  no_show_hrs: 4.0
```

Key-Value pairs parse into dicts

List of dicts

LivingPics ConfigMgr

```
class ConfigMgr(object):
    def __init__(self, config_file):
        """Create a ConfigMgr instance and populate
        from the specified config_file.
        """
        self.config_file = config_file
        with open(config_file, 'r') as fp:
            config = yaml.load(fp)
        for attr in 'timing exclude_file img_selection img_srcs'.split():
            self.__setattr__(attr, config[attr])

    def write(self):
        """Write the current configuration to the config file.
        """
        config = dict()
        for attr in 'timing exclude_file img_selection img_srcs'.split():
            config[attr] = self.__getattribute__(attr)
        with open(self.config_file, 'w') as fp:
            yaml.dump(config, stream=fp, default_flow_style=False)
```

Promote the top level dict keys to instance attributes

YAML and JSON

- JSON is a subset of YAML
- Encoder / decoder included in standard library as of Python 2.6
  - Grab simplejson from the Cheeseshop for versions back to 2.4
- But JSON syntax is stricter
  - you have to quote strings, include {}, etc.

PyYAML

- http://pypi.python.org/pypi/PyYAML/
- Python Magazine Dec-2008 article by Jesse Noller
  - http://jessenoller.com/2009/04/13/yaml-aint-markup-language-completely-different/

Getting the Image List

- From a file system
  - Use `os.walk()`
- From a Flickr photostream or group
  - Use `flickrapi`

http://www.flickr.com/services/api/

Using the Flickr API

- Get an API key for your app
  - Flickr gives you a key and a shared secret
- Get a frob
- Give the app permission to access your photos
- Convert the frob to a token
- Make authenticated calls to the API
- Or, use `flickrapi` to do the heavy lifting

Flickr Authentication

```
# Authenticate against Flickr for images from there
self.API_key = '...'
API_secret = '...'
self.flickr = FlickrAPI(self.API_key, API_secret)
try:
    token, frob = self.flickr.get_token_part_one()
    if not token:
        raw_input('Press ENTER after you authorize '
                  'LivingPics on Flickr')
    self.flickr.get_token_part_two((token, frob))
except IOError:
    print 'Warning: Unable to connect to Flickr'
```

After That, It's Easy!

- The `FickrAPI()` instance that you created and authenticated has a clever `__getattr__()` method that maps any method call you want to an API request to Flickr, and returns the results as an `ElementTree` object

flickapi Examples

```
def build_list_from_photostream(self, img_src):
    """Build the image list from a Flickr photostream.
    """
    resp = self.flickr.urls_lookupUser(api_key=self.API_key, url=img_src)
    user_id = resp.find('user').attrib['id']
    self.build_list_from_flickr(self.flickr_photos_search, user_id)


def build_list_from_group(self, img_src):
    """Build the image list from a Flickr pool
    """
    resp = self.flickr.urls_lookupGroup(api_key=self.API_key,
                                        url=img_src)
    group_id = resp.find('group').attrib['id']
    self.build_list_from_flickr(self.flickr_groups_pools_getPhotos,
                                group_id)
```

Replace the dots with underscores

flickrapi

- http://pypi.python.org/pypi/flickrapi
- http://stuvel.eu/projects/flickrapi
- http://www.flickr.com/services/api/

Tkinter

- http://docs.python.org/library/tk.html
- Cross platform
- Gets the job done...

Tkinter

- Widgets
- Geometry managers
  - Packer
  - Grid
- Event loop

Questions?

Thanks for listening!