Some Useful File Formats

Luís Pedro Coelho

Programming for Scientists

March 19, 2009

University of Pittsburgh

Carnegie Mellon

XML: Xtensible Markup Language

XML is a

- meta-format
- on which you define formats.

XML Says

You define a hierarchical structure of tags,
with optional attributes, and text.

A file format is only complete when you specify what the tags mean!

XML Example

```
<recipe name="bread" prep_time="5m" cook_time="3h">
  <title>Basic bread</title>
  <ingredient amount="8dL">Flour</ingredient>
  <ingredient amount="10g">Yeast</ingredient>
  <ingredient amount="4dl">Water</ingredient>
  <ingredient amount="1tsp">Salt</ingredient>
  <instructions>
    <step>Mix all ingredients together.</step>
    <step>Knead thoroughly.</step>
    <step>Cover with a cloth, and wait 1h.</step>
    <step>Knead again.</step>
    <step>Place in a bread baking tin.</step>
    <step>Bake in the oven at 180C for 30 min.</step>
  </instructions>
</recipe>
```

(Wikipedia)

XML

XML Pros & Cons

- Appropriate for heavy-duty formats
- Based on standards
- Parseable in any programming language
- extensible, mix-'n'-match
- XML has a lot attached to it.
  It would take us more than one lecture on how to parse that simple example!

Comma-Separated Values

- Simple idea
- Messy details

Great for communicating with a spread-sheet application.

CSV Example

```
luispedro,Luís Pedro Coelho,lpc@cmu.edu
rita,Rita Reis,rita@somewhere.com
```

Messy Issues

- Everything we talked about last time comes into play again (file enconding issues, line-endings,...)
- What if you want to have a comma inside a field?
- What if you want to have a newline inside a field?
- ...

CSV: Comma Separated Files

```
import csv
for name,user,email in csv.reader(file('emails.csv')):
    print name,user,email
```

Python Pickle

```
import pickle

obj = ...

pickle.dump(obj,file('output.pp','w'))

obj2 = pickle.load(file('output.pp'))
```

Gzip

```
import pickle
from gzip import GzipFile

obj = ...

pickle.dump(obj,GzipFile('output.pp.gz','w'))
obj2 = pickle.load(GzipFile('output.pp'))
```

Numpy Files (npy)

```
import numpy as np

A = np.arange(100).reshape((10,10))
np.save('output.npy',A)

B = np.load('output.npy')
```

JSON: JavaScript Object Notation

- Strings
- Lists
- String → Object Dictionaries

```
import simplejson

object = ['One','Two','Three']
print simplejson.dumps(object)
```

prints [”One”, ”Two”, ”Three”]

JSON II

```
import simplejson
complex = (range(3),
           {'Name': 'Luis',
           'Emails' : ['lpc@cmu.edu',
                       'lcoelho@andrew.cmu.edu']})
print simplejson.dumps(complex)
```

prints

```
[[0, 1, 2],
    {"Name": "Luis",
    "Emails": ["lpc@cmu.edu", "lcoelho@andrew.cmu.edu"]}]
```

Project Configuration

Options

- Which model to use for generation? (and parameters)
- Which model to use for image? (and parameters)
- Which method for detection? (and parameters)
- Which method for tracking? (and parameters)
- Which method for visualisation? (and parameters)
- Which statistics?

INI File

```
[Generation]
method=brownian
std=1.2
[Image-Generation]
method=single
shot-noise=2.0
[Detection]
method=otsu
filter-smaller=2
[Tracking]
method=hungarian
[Visualization]
method=colors
[Statistics]
speed=True
velocity=True
```

Desired Characteristics

- Read-able
- Edit-able
- Parse-able
- Standard

Parsing INI Files

```
import ConfigParser
config = ConfigParser.ConfigParser()
config.readfp(file('particles.ini'))
generation_method = config.get('Generation','method')
generation_options = dict(config.items('Generation'))

print generation_options
```

Particles

```
import ConfigParser
import optparse
parser = optparse.OptionParser()
parser.add_option('--config-file',
                  action='store',
                  dest='configfile',
                  default='particles.ini')
options,args = parser.parse_args()
config = ConfigParser.ConfigParser()
config.readfp(file(options.configfile))
```

Summary

- For your own stuff: Python pickles
- (Or, if it's only numpy arrays: npy)
- For simple communication/editing: INI files or JSON
- For heavy-duty publication: standard XML-based format