

Introduction to Python Persistence / Serialization

Christopher Barker

UW Continuing Education / Isilon

September 05, 2012

Table of Contents

- 1 Review/Questions
- 2 Serialization / Persistence
- 3 Python Specific Formats
- 4 Interchange Formats
- 5 DataBases
- 6 Object-Relational Mappers
- 7 Object databases

Review of Previous Class

Lightning talks today: Chris and John

- doctests
- unittests
- profiling

Homework review

Homework notes:

Anyone add some doctests or unittests to his project?

Anyone time or profile their project?

Did you find some bottlenecks??

Serialization

I'm focusing on methods available in the Python standard library

Serialization is the process of putting your potentially complex (and nested) python data structures into a linear (serial) form .. i.e. a string of bytes

The serial form can be saved to a file, pushed over the wire, etc.

Persistence

Persistence is saving your python data structure(s) to disk – so they will persist once the python process is finished.

Any serial form can provide persistence (by dumping/loading it to/from a file), but not all persistence mechanisms are serial (i.e RDBMS)

<http://wiki.python.org/moin/PersistenceTools>

Python Literals

Putting plain old python literals in your file

Pickle

Pickle is a binary format for python objects
You can essentially dump any python object to disk
(or string, or socket, or...

Example:

```
import pickle
```

```
data1 = {'a': [1, 2.0, 3, 4+6j],  
        'b': ('string', u'Unicode string'),  
        'c': None}
```

```
pickle.dump(open('data.pkl', 'wb'), output)
```

cPickle is faster than pickle, but can't be
customized – you usually want cPickle

Shelve

A shelf is a persistent, dictionary-like object

The difference with dbm databases is that the values (not the keys!) in a shelf can be essentially arbitrary Python objects

A shelf is essentially a "dbm" database with pickles as the values

NOTE: will not reflect changes in mutable objects without re-writing them to the db.
(or use writeback=True)

<http://docs.python.org/library/shelve.html>

Shelve

shelve example:

```
import shelve
```

```
d = shelve.open(filename)
```

```
d[key] = data    # store data at key
```

```
data = d[key]    # retrieve a COPY of data at key
```

```
del d[key]       # delete data stored at key
```

```
flag = d.has_key(key)    # true if the key exists
```

```
# as d was opened WITHOUT writeback=True, beware:
```

```
d['xx'] = range(4)    # this works as expected, but...
```

```
d['xx'].append(5)     # *this doesn't!* -- d['xx'] is STILL 1
```

JSON

JSON (JavaScript Object Notation) is a subset of JavaScript syntax used as a lightweight data interchange format.

Python module has an interface similar to pickle

Can handle the standard Python data types

Specializable encoding/decoding for other types –
but I wouldn't do that!

<http://www.json.org/>

<http://docs.python.org/library/json.html>

XML

XML is a standardized version of SGML, designed for use as a data storage/interchange format.

NOTE: HTML is also SGML, and modern versions conform to the XML standard.

XML looks like:

....

XML in the python std lib

`xml.dom:`

`xml.sax:`

`xml.parsers.expat:`

`xml.etree:`

INI

INI files

(the old Windows config files)

```
[Section1]
```

```
int = 15
```

```
bool = true
```

```
float = 3.1415
```

```
[Section2]
```

```
int = 32
```

```
...
```

Good for configuration data, etc.

ConfigParser

Writing ini files:

```
import ConfigParser
config = ConfigParser.ConfigParser()

config.add_section('Section1')
config.set('Section1', 'int', '15')
config.set('Section1', 'bool', 'true')
config.set('Section1', 'float', '3.1415')

# Writing our configuration file to 'example.cfg'
config.write( open('example.cfg', 'wb') )
```

Note: all keys and values are strings

ConfigParser

Reading ini files:

```
>>> config = ConfigParser.ConfigParser()
>>> config.read('example.cfg')
>>> config.sections()
['Section1', 'Section2']

>>> config.get('Section1', 'float')
'3.1415'

>>> config.items('Section1')
[('int', '15'), ('bool', 'true'), ('float', '3.1415')]
```

<http://docs.python.org/library/configparser.html>

CSV

CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases.

No real standard – the Python csv package more or less follows MS Excel standard
(with other "dialects" available)

Can use delimiters other than commas...
(I like tabs better)

Most useful for simple tabular data

CSV module

Reading CSV files:

```
>>> import csv
>>> spamReader = csv.reader( open('eggs.csv', 'rb') )
>>> for row in spamReader:
...     print ', '.join(row)
Spam, Spam, Spam, Spam, Spam, Baked Beans
Spam, Lovely Spam, Wonderful Spam
```

csv module takes care of string quoting, etc. for you

<http://docs.python.org/library/csv.html>

CSV module

Writing CSV files:

```
>>> import csv
>>> spamWriter = csv.writer(open('eggs.csv', 'wb'),
                             quoting=csv.QUOTE_MINIMAL)
>>> spamWriter.writerow(['Spam'] * 5 + ['Baked Beans'])
>>> spamWriter.writerow(['Spam', 'Lovely Spam', 'Wonderful'])
```

csv module takes care of string quoting, etc for you

<http://docs.python.org/library/csv.html>

anydbm

anydbm is a generic interface to variants of the DBM database (clones of the bsd dbm system)
Suitable for storing data that fits well into a python dict with strings as both keys and values
Note: anydbm will use the dbm system that works on your system – this may be different on different systems – so the db files may NOT be compatible!
whichdb will try to figure it out, but it's not guaranteed

<http://docs.python.org/library/anydbm.html>

anydbm module

Writing data:

```
#creating a dbm file:  
anydbm.open(filename, 'n')
```

flag options are:

- 'r' Open existing database for reading only (default)
- 'w' Open existing database for reading and writing
- 'c' Open database for reading and writing, creating it if it doesn't exist
- 'n' Always create a new, empty database, open for reading and writing

<http://docs.python.org/library/anydbm.html>

anydbm module

dbm provides dict-like interface:

```
db = dbm.open("dbm", "c")
```

```
db["first"] = "bruce"
```

```
db["second"] = "micheal"
```

```
db["third"] = "fred"
```

```
db["second"] = "john" #overwrite
```

```
db.close()
```

```
# read it:
```

```
db = dbm.open("dbm", "r")
```

```
for key in db.keys():
```

```
    print key, db[key]
```

sqlite

SQLite is a C library that provides a lightweight disk-based single-file database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language

SQLite is very broadly used as as an embedded databases for storing application-specific data etc. (Picassa, Apple tools, on iOS/Android, small spatial databases, etc....

Firefox plug-in:

<https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/>

DB-API

The DB-API spec (PEP 249) is a specification for interaction between Python and Relational Databases. Support by a large number of third-party Database drivers:

- MySQL
- PostgreSQL
- Oracle
- MSSQL (?)
-

<http://www.python.org/dev/peps/pep-0249>

Object-Relation Mappers ORMs

Systems for mapping PYton objects to tables in a RDBMS

Saves you writing that glue code

usually deals with mapping to various back-end RDBMSs

- SQL Alchemy

- <http://www.sqlalchemy.org/>

Django ORM

<https://docs.djangoproject.com/en/dev/topics/db/>

Object Databases

Python objects:

- ZODB (<http://www.zodb.org/>)
- Durus (<https://www.mems-exchange.org/software/DurusWorks/>)

NoSQL

Map-Reduce, etc.... What to say here???

Document-Oriented Storage:

- MongoDB (BSON interface, JSON documents)
- CouchDB (Apache):
 - JSON documents
 - Javascript querying (MapReduce)
 - HTTP API

Review/Questions
Serialization / Persistence
Python Specific Formats
Interchange Formats
DataBases
Object-Relational Mappers
Object databases

LAB



Lightning Talk

Lightning Talk:

Chris

Lightning Talk

Lightning Talk:

Peter

First Topic

A topic

some code example

Review/Questions
Serialization / Persistence
Python Specific Formats
Interchange Formats
DataBases
Object-Relational Mappers
Object databases

LAB



Review/Questions
Serialization / Persistence
Python Specific Formats
Interchange Formats
DataBases
Object-Relational Mappers
Object databases

Wrap up



Homework

Send me a copy of your project: due next Wed.

Keep learning about and using Python