Python modules for accessing .dbf (xBase) files

Project page
Download
Browse CVS

Description

dbfpy is a python-only module for reading and writing DBF-files. It was created by Jeff Kunce and then modified by Hans Fiby.

dbfpy module code has been placed in public domain.

What is dbfpy?

dbfpy can read and write simple DBF-files. The DBF-format was developed about 30 years ago and was used by a number of simple database applications (dBase, Foxpro, Clipper, ...). The basic datatypes numbers, short text, and dates are available. Many different extensions have been used; **dbfpy** can read and write only simple DBF-files.

What is it for?

- The most important use is reading old DBF-files created by other programs.
- DBF-files can be imported by most tabular data handling programs like Excel, Access, etc. If you want to generate tabular data for import into these programs from a python-script **dbfpy** would be an option.
- Some rather recent programs still use dbf for data storage. The GIS-application ArcView is one of them.

Do not use dbfpy ...

... if you need to store data for internal use in a python program. **dbfpy** does not provide any of the features of a modern database application. Use a database system like PostgreSQL, Oracle or MySQL for everyday data storage. Take SQLite if you want to go lightweight.

Example

dbfpy version 2.0.0

```
import datetime
from mx import DateTime
from dbfpy import dbf

## create empty DBF, set fields

db = dbf.Dbf("test.dbf", new=True)
db.addField(
    ("NAME", "C", 15),
    ("SURNAME", "C", 25),
    ("INITIALS", "C", 10),
    ("BIRTHDATE", "D"),
)
print db
print

## fill DBF with some records

for name, surname, initials, birthdate in (
    ("John", "Miller", "JM", (1980, 1, 2)),
    ("Andy", "Larkin", "AL", datetime.date(1981, 2, 3)),
    ("Bill", "Clinth", "", DateTime.Date(1982, 3, 4)),
    ("Bobb", "McNail", "", "19830405"),
):
    rec = db.newRecord()
    rec["NAME"] = name
    rec["SURNAME"] = surname
    rec["INITIALS"] = initials
    rec["BIRTHDATE"] = birthdate
    rec.store()
db.close()

## read DBF and print records

db = dbf.Dbf("test.dbf")
for rec in db:
    print rec
print

## change record

rec = db[2]
rec["INITIALS"] = "BC"
rec.store()
del rec

print db[2]


# vim: set et sts=4 sw=4 :
```

dbfpy version 1.0.0

demo1() is a simple python function that lists the content of the DBF-file `county.dbf`.

```
from dbf import *

def demo1():
    dbf1 = Dbf()
    dbf1.openFile('county.dbf', readOnly=1)

    dbf1.reportOn()
    print 'sample records:'
    for i1 in range(min(3,len(dbf1))):
        rec = dbf1[i1]
        for fldName in dbf1.fieldNames():
            print '%s:\t %s'%(fldName, rec[fldName])
        print
    dbf1.close()
```

demo2() creates a file `tst.dbf` and appends some records to it.

```
from dbf import *

def demo2():
    dbfn=dbf_new()

    dbfn.add_field("name",'C',80)
    dbfn.add_field("price",'N',10,2)
    dbfn.add_field("date",'D',8)
    dbfn.write("tst.dbf")
    # test new dbf
    print "*** created tst.dbf: ***"
    dbft = Dbf()
    dbft.openFile('tst.dbf', readOnly=0)
    dbft.reportOn()

    # add a record
    rec=DbfRecord(dbft)
    rec['name']='something'
    rec['price']=10.5
    rec['date']=(2000,1,12)
    rec.store()
    # add another record
    rec=DbfRecord(dbft)
    rec['name']='foo and bar'

    rec['price']=12234
    rec['date']=(1992,7,15)
    rec.store()
    dbft.close()
```