14.6. `plistlib` — Generate and parse Mac OS X `.plist` files¶

Changed in version 2.6: This module was previously only available in the Mac-specific library, it is now available for all platforms.

This module provides an interface for reading and writing the “property list” XML files used mainly by Mac OS X.

The property list (`.plist`) file format is a simple XML pickle supporting basic object types, like dictionaries, lists, numbers and strings. Usually the top level object is a dictionary.

Values can be strings, integers, floats, booleans, tuples, lists, dictionaries (but only with string keys), `Data` or `datetime.datetime` objects. String values (including dictionary keys) may be unicode strings – they will be written out as UTF-8.

The `<data>` plist type is supported through the `Data` class. This is a thin wrapper around a Python string. Use `Data` if your strings contain control characters.

See also

PList manual page
Apple’s documentation of the file format.

This module defines the following functions:

`plistlib.readPlist`(*pathOrFile*)¶

Read a plist file. *pathOrFile* may either be a file name or a (readable) file object. Return the unpacked root object (which usually is a dictionary).

The XML data is parsed using the Expat parser from `xml.parsers.expat` – see its documentation for possible exceptions on ill-formed XML. Unknown elements will simply be ignored by the plist parser.

`plistlib.writePlist`(*rootObject*, *pathOrFile*)¶

Write *rootObject* to a plist file. *pathOrFile* may either be a file name or a (writable) file object.

A `TypeError` will be raised if the object is of an unsupported type or a container that contains objects of unsupported types.

`plistlib.readPlistFromString`(*data*)¶
Read a plist from a string. Return the root object.

`plistlib.writePlistToString`(*rootObject*)¶
Return *rootObject* as a plist-formatted string.

`plistlib.readPlistFromResource`(*path*[, *restype='plst'*[, *resid=0*]])¶

Read a plist from the resource with type *restype* from the resource fork of *path*. Availability: Mac OS X.

Note

In Python 3.x, this function has been removed.

`plistlib.writePlistToResource`(*rootObject*, *path*[, *restype='plst'*[, *resid=0*]])¶

Write *rootObject* as a resource with type *restype* to the resource fork of *path*. Availability: Mac OS X.

Note

In Python 3.x, this function has been removed.

The following class is available:

class `plistlib.Data`(*data*)¶

Return a “data” wrapper object around the string *data*. This is used in functions converting from/to plists to represent the `<data>` type available in plists.

It has one attribute, `data`, that can be used to retrieve the Python string stored in it.

14.6.1. Examples¶

Generating a plist:

```
pl = dict(
   aString="Doodah",
   aList=["A", "B", 12, 32.1, [1, 2, 3]],
   aFloat = 0.1,
   anInt = 728,
   aDict=dict(
       anotherString="<hello & hi there!>",
       aUnicodeValue=u'M\xe4ssig, Ma\xdf',
       aTrueValue=True,
       aFalseValue=False,
   ),
   someData = Data("<binary gunk>"),
   someMoreData = Data("<lots of binary gunk>" * 10),
   aDate = datetime.datetime.fromtimestamp(time.mktime(time.gmtime())),
)
# unicode keys are possible, but a little awkward to use:
pl[u'\xc5benraa'] = "That was a unicode key."
writePlist(pl, fileName)
```

Parsing a plist:

```
pl = readPlist(pathOrFile)
print pl["aKey"]
```

Table Of Contents

14.6. `plistlib` — Generate and parse Mac OS X `.plist` files

- 14.6.1. Examples

Previous topic

14.5. `xdrlib` — Encode and decode XDR data

Next topic

15. Cryptographic Services

This Page

- Show Source

Navigation

- index
- modules |
- next |
- previous |
- Python v2.6.4 documentation »
- The Python Standard Library »
- 14. File Formats »

Last updated on Feb 26, 2010. Created using Sphinx 0.6.3.