Python Shelve Module

Home (/) / Modules (/modules) / Shelve

Overview:

One of the primitive forms of a database is a collection of values where each value is identified by a key, stored in disk and physical memory - called a DBM database or in a nutshell, a persistent dictionary.

The shelve module of Python Standard Library:

The shelve module of Python standard library is about providing the following functionalities to a Python program:

- A dictionary of pickled Python objects identified by keys
- A dictionary of pickled Python objects identified by keys that can be persisted to a file.
- Provide one or more forms of popular DBM implementations. A DBM database in its primitive form is a dictionary that can be persisted to a file.

Types of shelves:

- The `shelve` module of Python Standard Library provides multiple variants of shelves of pickles:
  - The `DbfilenameShelf` abstracts a dbm database and supports writing the keys vs pickles to a file.
  - The class `Shelf` offers a python dictionary of keys vs pickles.
  - The classes support caching from their actual storage: either disk or memory.

Example - Create a DbfilenameShelf object and write values:

```
import shelve
```

```
# Create a Shelf
shelf = shelve.open("/Valli/PythonProgs/TestShelve", "n");

# Check the type of the shelf
print("Type of the shelf object created:");
print(type(shelf));

# Add an integer and a double as values
shelf["a"] = 1;
shelf["b"] = 2.1;

# Add a python tuple
shelf["c"] = (2, 4, 6, 8);

# Add a python list
shelf["d"] = [1, 3, 5, 7];

print("Number of items present in the shelf:");
print(len(shelf));

# Close the shelf
shelf.close();
```

Output:

```
Type of the shelf object created:
<class 'shelve.DbfilenameShelf'>
```

```
Number of items present in the shelf:
4
```

Example - Open an existing DbfilenameShelf object and read the values:

```
# A python example program that reads values from a Shelf -
# a DbfilenameShelf

import shelve

dbFileName  = "/Valli/PythonProgs/TestShelve";
shelf       = shelve.open(dbFileName);

# Read all the items from the shelf
for item in shelf:
    print("%s:%s:%s"%(item, shelf[item], type(shelf[item])));

# Close the shelf
shelf.close();
```

Output:

```
b:2.1:<class 'float'>
d:[1, 3, 5, 7]:<class 'list'>
a:1:<class 'int'>
c:(2, 4, 6, 8):<class 'tuple'>
```

Shelf:

- The `Shelf` class is a dictionary. The `Shelf` class subclasses the `collections.abc.MutableMapping` abstract base class. In Python,

associative containers implement the interfaces put forth by the `mapping` or `MutableMapping` abstract base classes. The storage for a `Shelf` object is provided through the dictionary object it was initialized with during its creation. Remember, a dictionary uses another dictionary for its storage.
- A `Shelf` object stores pickled Python objects against string keys. When a lookup is performed the `Shelf` method(s) perform unpickling of the Python objects and return the value.

Example:

```
# Example Python program to create an in-memory shelf

import shelve
import pickle

# Class definition
class Record:

    # Initialiser
    def __init__(self, id, name, contents):
        self.id = id;
        self.name = name;
        self.contents = contents;

    def __repr__(self):
        return "Record>>id:%d, Name:%s, Contents:%s"%(self.ic

r1 = Record(1, "rec1", "Hello World");
r2 = Record(2, "rec2", "Hello Universe");

d1 = {};
shelf = shelve.Shelf(d1);

# Python internally pickles the values
shelf["1"] = r1;
shelf["2"] = r2;

# Python internally unpickles the values
print(shelf["1"]);
print(shelf["2"]);
```

Output:

```
Record>>id:1, Name:rec1, Contents:Hello World
Record>>id:1, Name:rec1, Contents:Hello World

Record>>id:2, Name:rec2, Contents:Hello Universe
```