Chapter 13 — Persistent Data

Introduction

Stand alone programs that do not save and use saved data are interesting, but most programs need to store and remember them. This chapter will introduce the UNIX DB and the Shelve Libraries to create stores for data. These are different from files, introduced in a previous chapter, in that

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Blah de blah.
- Baz and Barf.

Prerequisites

This Chapter requires... Support this work at The UNIX DB http://syw2l.org

The UNIX DB module has a confusing name because of its origin, but it is available on virtually all Python implementations. The UNIX DB (the dbm module) acts very much like a dictionary, except that key value pairs you set are stored on the hard drive for later.

The actual method and operating system library used to store the data varies between implementations. On UNIX and LINUX systems the GNU gdbm or the UNIX ndb libraries are used. On Windows systems a hashed file or "dumb" file is used to store the key value pairs. The files stored on the systems may not be readable on other systems, but using them to store values is the same.

The dbm module has a limitation that all of the keys and values must be collections of bytes. You may use strings, but the values when they are returned will be collections of bytes and need to be encoded back to their original format (if they include Unicode characters); This also means that you will need to encode/decode numbers into collections of bytes or to strings.



Using dbm

This sample program uses a dbm object to store a value that the user enters. When the user runs the program again they can either change the value, or retrieve the old value. This program demonstrates that if a Unicode character outside the range of 0-127 is entered, the result will be returned as a collection of bytes. Like in file-system IO, it is strongly suggested that you use the with statement to manage the context of the module.

To use the dbm module, you first must import dbm into your program. This will load the best library for key value storage you have available to you.

Now all you need to do is "open it". The open method dbm.open () usually requires two values, file name and mode. The file name may be a simple name or the path to a file and does not need to include a file extension. The mode may be: 'r' -read only, 'w' -reading and writing, 'c' — create a new database if it does not exist for reading and writing, and 'n' - always create a new database. The 'r' mode is default (if you do not specify a mode).

If you are using the dbm module in a with context manager, it will be closed for you automatically when you leave the suite of code. If you are accessing dbm outside a context manager, you will need to close your dbm object using the .close() method. NIC WORK

Γ IEase support this w	<u>UIN al</u>
dbm	Module
The dbm module will create a persistent key-value storage of byte arrays for your program to use	
REF	

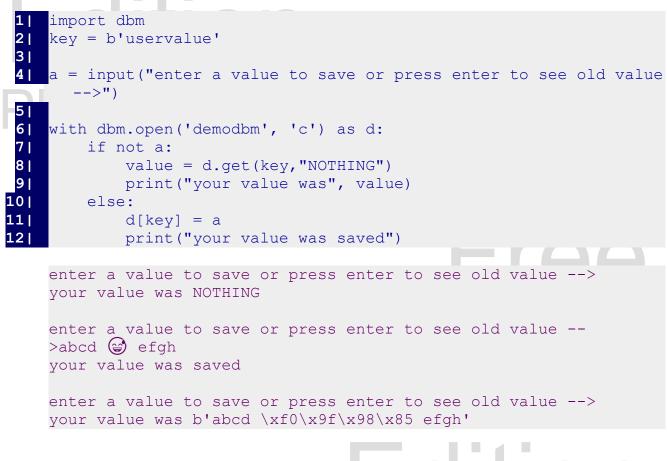
dbm.oj	pen(file_name, mode)	Method of dbm
-	UNIX database with the file name specified and return an object that can to access it.	
Mode	Description	
"r"	Open the database for "read-only" access. This is the default action of mode is not specified.	UN
"w"	Open the database for "write-only". Use this mode is you are populating a database with large quantities of data and you are not going to be reading from it.	00
"с"	Create a database if it does not exist and then open it in read and write	



An Introduction to STEM Programming with Python — 2019-09-03a Chapter 13 — Persistent Data

	mode.	
"n"	Delete the old database create a new one. Open it in read and write mode.	
REF		

dbm_object.close()	Method of dbm
Closes a dbm database. If a database is not closed, changes may not be committed to the disk. It is recommended that you use a with context manager, and allow it to automatically close your database.	
REF	



Using Shelve for Persistent Values



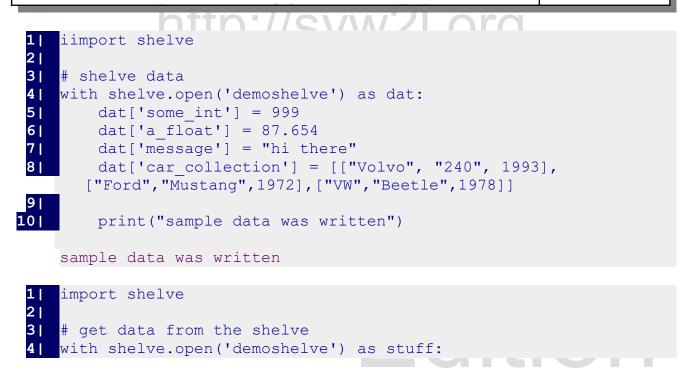
The shelve library actually combines the pickle library and the dbm library to create an easy way to store virtually anything. With shelve you do not have to convert the data to a string or a collection of bytes. The pickle library is insecure and may expose your system to malicious code. It is recommended that you do not use shelve data files from untrustworthy sources. As with dbm it is strongly recommended that you execute the .close() method or execute it as a context manager using the with statement.

The shelve module combines the dbm module with the pickle module to create a key-value persistent store that will return values of various types, and nor just byte arrays.	
Data files, created by shelve, from unknown sources are not secure and should not be used. REF	

 shelve.open(file_name)
 Method of shelve

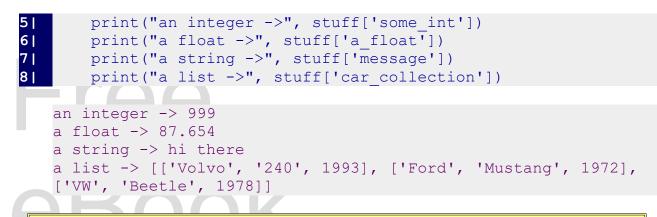
 Closes a shelve for reading and writing. Create a new one of it does not exist.
 Image: Closes a shelpe for reading and writing.

 REF
 Image: Closes a shelpe for reading and writing.
 Image: Closes a shelpe for reading and writing.





An Introduction to STEM Programming with Python — 2019-09-03a Chapter 13 — Persistent Data



NOTE: It is recommended that when accessing values from either a dbm or shelve database that you trap for errors; check to see if the key exists (using the in operator); or use the .get(key, default) method with a default value.

Sample Program — Number Guess with History

This sample program plays the classic number guessing game where after each guess you are given a clue; higher or lower. It uses the shelve module to store how many times the game has been played and the fewest number of guesses that were made to win. It creates a local file called "numberguess.dat" to store this persistent information.

```
11
    import random
 21
    import shelve
 31
 4 |
    # save constants into variales
 5|
    DATABASENAME = "numberguess"
 6|
    PLAYED = "played times"
 7|
    PLAYED DEFAULT = 0
 8 |
    FEWEST = "fewest quesses"
 9|
    FEWEST DEFAULT = 100
10|
11|
    print("number guessing game")
12|
    with shelve.open(DATABASENAME) as db:
13|
        played times = db.get(PLAYED, PLAYED DEFAULT)
14|
        fewest guesses = db.get(FEWEST, FEWEST DEFAULT)
15|
16|
        print('this game has been played', played times, 'times')
17|
        print("can you beat", fewest guesses, 'guesses?')
18|
191
    #
```



An Introduction to STEM Programming with Python — 2019-09-03a Chapter 13 — Persistent Data

```
20 n = random.randint(1,100)
    print("i am thinking about a number from 1 to 100.")
21|
22|
    print("can you guess it?")
231
    print("you will get a clue after each guess.")
24 |
251
    guesses = 0
261
    while True:
27|
        try:
281
             guess = int(input("enter your guess (or -1 to quit)?"))
291
             if quess = -1:
30|
                 print('good bye')
311
                 break
32|
             if (quess \geq 1 and quess \leq 100):
331
                 quesses = quesses + 1
34|
351
                 if quess == n:
361
                     print('you got it in', guesses, 'guesses.')
371
                     # reopen shelve and update
381
                     with shelve.open(DATABASENAME) as db:
39|
                         db[PLAYED] = db.get(PLAYED, PLAYED DEFAULT)
       + 1
401
                         if guesses < db.get(FEWEST, FEWEST DEFAULT):
41|
                              db[FEWEST] = quesses
42|
                     break
43|
                 elif quess < n:
44|
                     print('you need to guess higher')
451
                 else:
461
                     print('you need to quess lower')
47|
             else:
48|
                 print("your guess should be between 1 and 100")
491
         except:
50|
             print("you need to enter an integer.")
51|
    print('thanks for playing')
```

number guessing game this game has been played 4 times can you beat 6 guesses? i am thinking about a number from 1 to 100. can you guess it? you will get a clue after each guess. enter your guess (or -1 to quit)?59 you need to guess higher enter your guess (or -1 to quit)?77



