Lab 8

Lab 8 - Hash Tables

Attached Files: bdaylist.txt (991 B)

CS 172 - Lab 8

Hashing and Hash Tables

In this lab, you will implement your own open-hashing hash table an implement a hashable object type, Birthday, that can be used as a key into the hash table.

Pair Programming

This lab uses Pair Programming. You must work in a pair with another student. If the class has an odd number of students, the Teaching Assistants may create a group of three students. During the course of the lab, each member of the pair must take each of the following positions.

- **Driver:** writes and tests code
- **Observer:** watches the driver, reviews code for errors and suggests way to improve program.

You will only receive full credit for the lab if you work as both a **Driver** and **Observer** during the course of the lab.

Part 1: The Birthday Class

The key for our hash table will be Birthday objects. Create a class called Birthday that has the following:

- Three attributes for birth day, birth month, birth year
- A constructor method that accepts the day, month, and year as parameters.
- A __str__ method to provide a string representation of the Birthday object.
- A __hash__ method that returns an integer as the sum of of the day, month, and year, mod 12. So if day=1, month=11, year = 1990, then this method would return (1+11+1990)%12, which is 10.
- An __eq__ method to test if two Birthday objects have the same attribute value.

Place your implementation in a module called **Birthday.py** and create a __main__ section where you test instantiating Birthday objects, printing them, and calling the __hash__ method.

Part 2: Main Application

Switch roles!

Create a **lab8.py** usage file. In this file you will:

1. Create an empty hash table
2. Read in a list of birthdays from the supplied **bdaylist.txt** file.
3. For each birthday, create a `Birthday` object, and add the tuple `(Birthday,i)` to the appropriate list in the hash table, where `i` is the line number from the input file.
4. Output the total length of the list at each of the hash locations.

Here's a few things to help you get started:

- Since our Birthday object's hash function hashes to integers in the range [0,12), we need to create a hash table with 12 empty lists in it.   Here's some code to do this:

```
hashtable = []

for i in range(12):

    hashtable.append([])
```

- Recall how easy reading all the lines from a file is in Python!
    - Open the file, via the `open` function.
    - Read in the lines via the `readlines()` method to a list
    - Iterate over this list
- If an object implements the `__hash__` method (as you were asked to do for the `Birthday` class), then you can call the `hash(obj)` function on the object to get the hash location (it basically calls `obj.__hash__()`)

Sample Output

```
Hash location 0 has 10 elements in it
Hash location 1 has 12 elements in it
Hash location 2 has 8 elements in it
Hash location 3 has 10 elements in it
Hash location 4 has 5 elements in it
Hash location 5 has 7 elements in it
Hash location 6 has 7 elements in it
Hash location 7 has 11 elements in it
Hash location 8 has 4 elements in it
Hash location 9 has 8 elements in it
Hash location 10 has 8 elements in it
Hash location 11 has 10 elements in it
```

Scoring

The score for the assignment is determined as follows.

- 10 points - Attendance
- 10 points- Acted as Driver
- 10 points- Acted as Observer
- 35 points - Part 1: Designed, implemented, and tested Birthday class
- 35 points - Part 2: The main application.