CS228: Human Computer Interaction

Deliverable 8

Description

In this deliverable, you will create a database. This database will:

1. store information about people who use your system;
2. it will be loaded every time your system starts up;
3. it will be updated as the current user interacts with your system; and
4. it will be re-saved when your program ends.

This deliverable will be stored in a particular Python data structure known as a dictionary. If you are unfamiliar with this data structure, it is recommended that you work through this short tutorial before attempting this deliverable.

Instructions

1. Let's create a new python program called `Dict.py` which you can use to build up this database. At the end of this deliverable, you will have to incorporate this database into the code you developed in the previous deliverable. In `Dict.py`, let's create an empty dictionary called `database`:

   (a) `database = {}`

   (b) `print database`

2. Now, let's save this database to a file every time this program ends. Place these lines

   (a) `import pickle`

   (b) ...

   (c) `pickle.dump(database,open('userData/database.p','wb'))`

   at the start and end of your program, respectively. This will save the empty database to this file. (Make sure you have a subdirectory called `userData` in the same directory where `Dict.py` is running.)

3. Now, instead of creating an empty database every time this program starts up, you will read the database in from the file. Replace line 1(a) with

   (a) `database = pickle.load(open('userData/database.p','rb'))`

When you run this program, you should still see an empty database printed out: we haven't stored anything in the database yet.

4. When the program starts up, we will now ask for the user's name. If this user is not present in the database, we will add her. If she has used this program before and is already in the database, we will not add her. This will result in a database that has one entry for each user that has used your system at least once. These lines

```
(a) userName = raw_input('Please enter your name:  ')
(b) if userName in database:
(c)       print 'welcome back ' + userName + '.'
(d) else:
(e)       database[userName] = {}
(f)       print 'welcome ' + userName + '.'
```

   should be added after line 3(a). These lines collect the user's name (a) and check to see if the user is stored in the database (b). If she is, welcome her back (c). If not (d), add an entry to the dictionary (e). This entry has as its key the name of the user. The value of this key is another empty dictionary, as indicated by the empty curly brackets. Later, you will fill this dictionary, associated with this user, with information about this user's performance as they use your system. Once this user has been added, we welcome her to the system (f).

5. Let's write a small, separate program called `Reset.py` that will reset the database. It will do this by saving an empty dictionary to the file `database.p`:

```
(a) import pickle
(b) dictionary = {}
(c) pickle.dump(dictionary,open('userData/database.p','wb'))
```

6. Let's make sure that `Dict.py` is working correctly. If you run it, you should something like the following behavior:

```
(a) python Dict.py
(b) Please enter your user name:  josh
(c) welcome josh.
(d) {'josh':  {}}
(e)
(f) python Dict.py
(g) Please enter your user name:  josh
(h) welcome back josh.
(i) {'josh':  {}}
```

(j)

(k) `python Dict.py`

(l) `Please enter your user name:  josh2`

(m) `welcome josh2.`

(n) `{'josh':  {}, 'josh2':  {}}`

Here, user `josh` uses the program for the first time (a-d). This same user then uses the program a second time (f-i). Then, a new user, `josh2`, uses the system for the first time (k-n). When this third interaction terminates, we can see that the database now contains two empty dictionaries, one for each of the two users (n).

Hint: As you debug your program, you might need to run `Reset.py` from time to time to empty your database.

7. Modify your code now so that, for each user, you record how many times they have logged into the system. (No need for passwords at this stage.) This will require you to do the following:

   (a) If the user is new, store a key in each user's empty dictionary called `'logins'`. The value of that key should initially be set to 1.

   (b) If the user is returning to the system, you should increment the value of the `'logins'` key by one.

   If you do this correctly, you should see the following behavior:

   (a) `python Dict.py`

   (b) `Please enter your user name:  josh`

   (c) `welcome josh.`

   (d) `{'josh':  {'logins':  1}}`

   (e)

   (f) `python Dict.py`

   (g) `Please enter your user name:  josh`

   (h) `welcome back josh.`

   (i) `{'josh':  {'logins':  2}}`

   (j)

   (k) `python Dict.py`

   (l) `Please enter your user name:  josh2`

   (m) `welcome josh2.`

   (n) `{'josh':  {'logins':  2}, 'josh2':  {'logins':  1}}`

   Hint: Compare lines (d), (i), and (n) above with lines 6(d), 6(i), and 6(n).

8. Now, gradually incorporate this code into the code you developed in the previous deliverable. Make sure that all of this code is called before you start Matplotlib's interactive mode (`matplotlib.interactive(True)`). Once you start interactive mode, you cannot collect text from the user.

9. Now, think about what performance metrics you want to record while the user interacts with the system. In order to do so, you will have to add a key and value pair to a user's dictionary when they perform some action for the first time. For example, they attempt to sign the ASL digit '3' for the first time:

   (a) `userRecord = database[userName]`

   (b) `userRecord['digit3attempted'] = 1`

   If they have attempted that action before, you need to increment the key and value pair associated with that action. For example, ASL digit '3' is presented to them again, and they attempt to sign it:

   (a) `userRecord['digit3attempted'] = userRecord['digit3attempted'] + 1`

10. As a first step, let's do the following: present the user with one of the 10 ASL digits, chosen at random. Regardless of whether the user successfully signs it or not, record the fact that they have been presented with this digit after a set number (your choice) of iterations through your main drawing loop have occurred. After you have recorded this information, choose a new digit at random and repeat.

11. Draw something to the screen to indicate to the user how many times they have been presented with each of the 10 ASL digits. This can be text or a drawing. It can be in the same panel where they see the virtual hand or in a separate panel.

12. Rather than saving out the database only when the program terminates, save out the database to the file whenever this transition between one digit and another occurs.

13. Now, create a short video that shows:

    (a) A new user logging into the system for the first time.

    (b) Random digits presented for them to sign.

    (c) Text or a drawing should show up that indicates how many times they've been presented with a digit.

    (d) Show the user stopping the program.

    (e) Show them logging back in.

    (f) Show them presented with randomly-chosen digits again.

    (g) The presentation counts for the 10 digits should be same at the start of this second session as they were at the end of the first session.

14. Upload your video to YouTube and submit to BlackBoard.