**Week 11 In-Class Exercises (dictionaries)**

**Note:** Please download the file “Week 11 In-Class Exercises Starting Code (v1.0).ipynb”.

## Q1: [ \* ] - Looking up a Value inside a Dictionary

You are given a dictionary called book\_author\_dict that stores the mappings from books to their authors. See the Jupyter notebook for the starting code.

Write a program that allows the user to repeatedly search for the author of a book until the user chooses to stop the program. When the author of a book is found, the name is displayed. When the book is not found in the dictionary, "Not found!" should be displayed.

A sample run of the program can be found below. Text in bold font is user input.

Do you want to search for the author of a book? [Y|N]**Y**

Please enter a book title :**Animal Farm and 1984**

The author of the book is George Orwell

Do you want to continue? [Y|N]**Y**

Please enter a book title :**Da Vinci**

Not found!

Do you want to continue? [Y|N]**Y**

Please enter a book title :**The Da Vinci Code**

The author of the book is Dan Brown

Do you want to continue? [Y|N]**N**

Good-bye!

**Q2:**

**Part (a) [ \* ]**

Define a function called display\_all\_gpas(). The function takes in as its parameter a dictionary that stores the GPAs of a number of students. For example, a dictionary may be like {'George Leung':3.4, 'Eric Wong':3.9, 'Michelle Lee':3.1}.

The function displays all the GPAs in the following format:

Student Name Student GPA

------------ -----------

George Leung 3.4

Eric Wong 3.9

Michelle Lee 3.1

You are given a dictionary in the starting code to test your function. Invoke the display\_all\_gpas() function that you have written. Check that you get the proper output.

**Part (b) [ \*\* ]**

Given a dictionary containing students’ GPAs, write a program that tries to update the GPA of a student. The program first displays all the current GPA information. It then prompts the user for the name of the student whose GPA needs to be updated. If the name exists, the program prompts for the new GPA value and updates it in the dictionary. If the name doesn’t exist, the program displays an error and keeps prompting for the name until it’s an existing name.

**Use the function you have defined in Part (a) to display a GPA dictionary.**

A sample run of the program looks like the following:

Current GPA information:

Student Name Student GPA

------------ -----------

George Leung 3.4

Eric Wong 3.9

Michelle Lee 3.1  
  
Whose GPA do you want to change? **Wong Eric**  
Sorry! This student doesn’t exist.

Whose GPA do you want to change? **Eric Wong**  
What's the new GPA? **3.8**  
Thanks! GPA has changed.   
  
The new GPA information:  
  
Student Name Student GPA  
------------ -----------  
George Leung 3.4  
Eric Wong 3.8  
Michelle Lee 3.1

**Part (c) [ \*\* ]**

Given a dictionary containing students’ GPAs, write a program that tries to insert the GPA of a new student. The program first displays all the current GPA information. It then prompts the user for the name of the new student. If the name doesn’t exist in the dictionary, the program prompts the user for the GPA of this new student and adds the information into the dictionary. If the name already exists, the program displays an error and keeps prompting for another name until the name does not exist in the dictionary.

A sample run is shown below:

Current GPA information:

Student Name Student GPA

------------ -----------

George Leung 3.4

Eric Wong 3.9

Michelle Lee 3.1  
  
Whose GPA do you want to add? **Eric Wong**  
Sorry! This student already exists.

Whose GPA do you want to add? **Michelle Lee**  
Sorry! This student already exists.

Whose GPA do you want to add? **Peter Liu**  
What's the GPA of Peter Liu? **3.7**  
Thanks! GPA has been added.  
  
The new GPA information:  
  
Student Name Student GPA  
------------ -----------  
George Leung 3.4  
Eric Wong 3.9  
Michelle Lee 3.1  
Peter Liu 3.7

**Q3:**

**Part (a) [ \*\* ]**

Define a function called convert\_to\_dict() that takes in a list of tuples as its parameter, where each tuple has two elements. The function then stores the information in the list of tuples into a dictionary by treating the first element of a tuple as a key and the second element of a tuple as a value. The function returns the dictionary created.

For example, suppose the list of tuples passed to the function is [(1, 'apple'), (2, 'banana'), (4, 'durian'), (8, 'orange'), (3, 'peach')], then the function should return the dictionary {1:'apple', 2:'banana', 4:'durian', 8:'orange', 3:'peach'}.

You can assume that in the given list the first elements of tuples are all different.

**Part (b) [ \*\* ]**

Define a function called read\_into\_dict(). The function takes in a file name as its parameter. The file contains multiple lines of data, where each line contains two columns of data separated by a tab. The function reads in the file and treats the first column of each row as a key and the second column of each row as a value, i.e., each row of the file is a key/value pair. The function stores these key/value pairs into a dictionary and returns the dictionary.

You can assume that the file doesn’t contain any two rows that have the same key in the first column.

For example, suppose the file contains the following:

SMU Singapore Management University

NUS National University of Singapore

then the function should return the following dictionary:

{'SMU': 'Singapore Management University', 'NUS': 'National University of Singapore'}

**Part (c) [ \*\* ]**

Define a function called reverse\_dict() that takes in a dictionary as its parameter. This function returns a new dictionary that stores the reversed mappings in the given dictionary. For example, if the given dictionary is {'Monday': 'swimming', 'Tuesday': 'basketball', 'Wednesday': 'tennis', 'Thursday': 'yoga', 'Friday': 'gymnastics'}, then the function returns {'swimming': 'Monday', 'basketball': 'Tuesday', 'tennis': 'Wednesday', 'yoga': 'Thursday', 'gymnastics': 'Friday'}.

You can assume that in the given dictionary, there are no duplicated values.

**Q4:**

You are given a file called "students.txt" where each line of the file contains the following information:

Name Email ID Birthdate Gender

The four columns of information are separated by tabs.

**Part (a) [ \*\* ]**

Write a program that allows a user to quickly find the birth date of a student based on his/her email ID. **(I.e., you should use a dictionary to store the data for fast lookup later.)**

A sample run of the program looks like the following:

Please enter an email ID:**j.wong.2017**  
The birthdate of this student is 03-09-1998  
  
Do you want to continue? [Y|N]:**Y**  
  
Please enter an email ID:**abc**  
This is not a valid email ID.  
  
Do you want to continue? [Y|N]:**Y**  
  
Please enter an email ID:**eric.wong.2017**  
The birthdate of this student is 12-15-1999  
  
Do you want to continue? [Y|N]:**N**  
Good-bye!

**Part (b) [ \*\*\* ]**

Using the same file, now write a modified version of the program above such that the program displays the name, gender and birthdate of a student based on the email ID. **Again, you should use a dictionary to store the data from the given file. Use just one dictionary to store the data.**

Please enter an email ID:**j.wong.2017**  
This student is Joe Wong, male, born on 03-09-1998.  
  
Do you want to continue? [Y|N]:**Y**  
  
Please enter an email ID:**abc**  
This is not a valid email ID.  
  
Do you want to continue? [Y|N]:**Y**  
  
Please enter an email ID:**eric.wong.2017**  
This student is Eric Wong, male, born on 12-15-1999.  
  
Do you want to continue? [Y|N]:**N**  
Good-bye!

**Q5: Contacts [ \*\*\* ]**

In this exercise, you are given a file called "contacts.txt". (Note that the format of this file is the same as the file "phone\_book.txt" you used last week, but the data inside is different.)

Write a program that reads the file and stores the data in a dictionary. The program then prompts the user for a person’s name and displays all the phone numbers of that person. Here retrieving all the phone numbers of a person should be a single dictionary lookup, i.e., you should not need to go through all the phone number records in order to retrieve the phone numbers of a person. In other words, you should have a dictionary that stores the mapping between a person’s name and all his/her phone numbers.

Three sample runs of the program can be found below:

* Sample Run #1:

Enter a person’s name: Peter Liu

Peter Liu has 1 number(s):

(+65)95467564

* Sample Run #2:

Enter a person’s name: Michelle Lee

Michelle Lee has 2 number(s):

(+65)67894321

(+65)88776655

* Sample Run #3:

Enter a person’s name: Michelle Ong  
Michelle Ong cannot be found in our database.