**Week 10 In-Class Exercises (files)**

**Note:** Please download the file “Week 10 In-Class Exercises Starting Code (v1.0).ipynb”.

**Q1: Numbers in Files [\*]**

**Part (a)**

Define a function called sum\_up(). The function takes in a string called input\_file\_name as its parameter, which is the name of a text file. Each line of the text file is a number. The function ***returns*** the sum of all the numbers inside the text file.

We’ve provided you with two text files, q1-1.txt and q1-2.txt. The test cases are also provided to you in the Jupyter notebook.

**Part (b)**

Define a function called create\_number\_file(). The function takes in a file name (called output\_file\_name) and a positive integer (called n) as its parameters. The function writes to a text file named output\_file\_name all the even numbers between 0 and n (both inclusive).

For example, if output\_file\_name is 'q1-output.txt' and n is 10, then the generated q1-output.txt should look like the following:

0  
2  
4  
6  
8  
10

**Q2: Books**

You are given three files (“books-1.txt”, “books-2.txt”, “books-3.txt”) that contain books, where each book has a title, an author and a price. The three pieces of information are contained in three columns separated by tabs. For some of the books, the price information is missing, i.e., the third column in the file for that book is an empty string (but there is still a tab between the second column and the third column in this case).

Open one of the files to understand the structure of the files.

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

Define a function called get\_books\_cheaper\_than() that takes in a file name (called input\_file\_name), a number (called price\_limit) and another file name (called output\_file\_name) as its parameter. The function writes to output\_file\_name those books from the given file whose prices are lower than price\_limit. For those books in the input file whose price information is missing, they should not be included in the output file.

We’ve provided two test cases for you in the Jupyter notebook.

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

Define a function called check\_same\_author() that takes in a file name (called file\_name) as its parameter. The function returns True if all books from the given file are written by the same author and False otherwise. You can assume that the file contains at least one book. For books whose price information is missing, they should still be checked.

We’ve provided three test cases for you in the Jupyter notebook.

**Q3: Phone Book**

You are given a file called “phone\_book.txt” that contains the names and phone numbers of a list of people. The name of a person and his/her phone number are separated by the character ‘|’.

Phone numbers are in one of the following three formats:

* An 8-digit number, which is a Singapore-based number. E.g., 98765432
* A phone number starting with a country code indicated by ‘+’. There is a single space after the country code. Examples of phone numbers in this format include the following:
  + +65 87654321
  + +60 234567890
* A phone number starting with a country code indicated by ‘+’ and inside a pair of parentheses. Examples of phone numbers in this format include the following:
  + (+65)87654321
  + (+60)320000000

Open the file to understand the structure of the file.

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

Read the file line by line and display only those lines where the phone numbers are Singapore-based numbers (i.e., either the number doesn’t have a country code or the country code is +65).

Your code should produce the following output. (Use '\t' to separate the name and the phone number to produce the output below.)

George Leung 98987676

Michelle Lee (+65)67894321

Eric Wong +65 91234567

Michelle Lee (+65)88776655

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

Because some people have more than one phone numbers, the file “phone\_book.txt” have some lines that contain the same name but different phone numbers. Create a new file called “phone\_book\_reorganized.txt” that groups phone numbers of the same person together, as shown below:

George Leung  
98987676  
  
Eric Wong  
+60 412345678  
+65 91234567  
  
Michelle Lee  
(+65)67894321  
(+65)88776655

Note that after each person’s phone numbers are displayed, there is a blank line.

**Q4: News**

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

You are given a text file called “news.txt”. Each line of the file is the headline of a news article. Write a program that prompts the user for a keyword or a keyphrase and then displays those headlines that match the keyword (or keyphrase). (Here matching means the keyword or keyphrase is a substring of the headline.) Matching is case-insensitive.

For example, if the keyword is “travel”, then those headlines containing the substring “travel” or “TRAVEL” (or “Travel”, “tRAvEL”, etc.) are all considered matching headlines.

The matching headlines are displayed together with a sequence number.

Two sample runs of the program are as follows:

* Sample Run #1:

Enter a keyword or a keyphrase:**travel**

1. COVID: 14-day travel history, 10-day SHN for incoming travellers

2. Singapore, South Korea to launch Vaccinated Travel Lanes from 15 November

* Sample Run #2:

Enter a keyword or a keyphrase :**covid cases**

1. Record 2,478 new COVID cases in Singapore, 2 unvaccinated deaths

2. Singapore's total COVID cases exceed 100,000 with 2,356 new infections, 4 more deaths

3. Large number of COVID cases putting 'significant strain' on healthcare system: MOH

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

Let us now improve the program above. Your program should keep asking the user whether he/she wants to search for matching headlines again. If so, the program prompts for a keyword or keyphrase and then displays the matching headlines. Also, when the matching headlines are displayed, the program will show how many matching headlines are found and will assign sequential numbers to the matching headlines when displaying them.

Hints:

* Store the headlines into a list in the very beginning. This list can be re-used throughout the program. I.e., you only need to read the file once in the beginning of the program.
* Introduce functions to help you simplify your code. For example, define a function that takes in a list of headlines and a keyword/keyphrase as its parameters. The function returns a list of matching headlines.

A sample run of the program is shown below:

Do you want to search our news database? [Y|N] :Y

Please enter a keyword or a keyphrase :malaysia

There is no matching headline!

Do you want to search again? [Y|N] :Y

Please enter a keyword or a keyphrase :jail

There are 1 matching headlines:

1. NSF who left home twice while on MC jailed 18 weeks

Do you want to search again? [Y|N] :Y

Please enter a keyword or a keyphrase :covid cases

There are 3 matching headlines:

1. Record 2,478 new COVID cases in Singapore, 2 unvaccinated deaths

2. Singapore's total COVID cases exceed 100,000 with 2,356 new infections, 4 more deaths

3. Large number of COVID cases putting 'significant strain' on healthcare system: MOH

Do you want to search again? [Y|N] :N

Good-bye!