COMP1005AB – Fall 2021
Assignment # 3
Due on Saturday, November 13th by 11 pm.

Submission Guidelines:

Submit a single zip file called A3.zip on Brightspace.

Did you zip it correctly? You must always create the zip files using the built-in, default, archiving program available with your operating system. If we cannot easily open your zip file and extract the python files from them, we cannot grade your assignment. Other file formats, such as rar, 7zip, etc., will not be accepted.

Windows: Highlight (select with ctrl-click) all of your files for submission. Right-click and select "Send to" and then "Compressed (zipped) folder". Change the name of the new folder "A3.zip".

MacOS: Highlight (select with shift-click) all of your files for submission in Finder. Right-click on one of the files and select "compress N items…" where N is the number of files you have selected. Rename the "Archive.zip" file "A3.zip".

Linux: use the zip program.

Did you submit the correct files? You are responsible for ensuring that you have submitted the correct file(s), and the submission is completed successfully. After submitting your A3.zip file to Brightspace, be sure that you download it and then unzip it to make sure that you submitted the correct files. This also checks that your zip file is not corrupted and can be unzipped.

You can submit as many times as you wish and Brightspace will save your latest submission. I would suggest that you submit as soon as you have one question done and keep re-submitting each time you add another problem (or partial problem).

We will not accept excuses like "I accidentally submitted the wrong files" or "I didn't know that the zip file was corrupt" or "My Internet was down" or "My computer was not working" after the due date.

If you are having issues with submission, contact the instructor before the due date.

Invalid submissions and penalties

- Submissions with an incorrect file name or format will receive a 10% penalty, with no exceptions. You are responsible for following the guidelines outlined in this assignment and the course outline.
- Add your name and id number as comments on top of each py file. If this info is not present, a 10% penalty will be applied.
- Submissions that crash (i.e., terminate with an error) on execution may receive a mark of zero (0).
- Each function that does not have a docstring will receive -1 mark.

Late Policy

The assignment is due on Saturday, November 13th, 2021 by 11:00 PM (Ottawa time). Late submissions are allowed until Sunday, November 14th, 2021, 11:00 PM (Ottawa time), but a penalty will be applied.

Please note that Brightspace will only keep your LAST submission. If you submit on-time and then submit the exact same files after the due date, it will be treated as a LATE assignment.

Here is a summary of the penalties based on the submission time.

Last submission Day/Time	Penalty applied
Saturday, November 13th, 2021 (11:00 PM Ottawa time)	No penalty
Sunday, November 14th, 2021 (12:00 PM noon Ottawa time)	10%
Sunday, November 14th, 2021 (11:00 PM Ottawa time)	20%
Monday, November 15th or later	100%

Regret Clause

If you think you have committed an academic violation (plagiarism), you have 12 hours after the final allowable submission time (11 AM on Monday, November 15th, for this assignment) to invoke this regret clause.

If you do, you will receive ZERO for the assignment (or portions of it that you invoke the clause for) but will not receive any further sanctions from the Dean's office. Your assignment will still be used when we are checking for plagiarism, but you would not face any further sanctions if it was decided that you had committed an academic violation.

For example, if you collaborated too closely with another student and this is discovered, you would not be further sanctioned. The other student, if they did not also invoke this clause, may still be further sanctioned by the Dean's office.

This is not a group project, so collaboration is not allowed on this assignment.

Self Evaluation Quiz Criteria

SE Quiz 3 is a follow up quiz for Assignment-3. Once you submit this assignment, you can take the quiz on Brightspace.

You will be asked some leading questions to help you asses the problem-solving skills that you might have achieved by doing this assignment, such as:

- Do you feel more comfortable with programming terminologies?
- Did you draw a flow-chart or write a pseudocode to help understand an algorithm?
- Did you try to break your problem into smaller sub-problems and solve each sub-problem separately?
- Did you use examples to help understand a problem?
- Could you explain the problems in your own words to someone else in this class?
- Did you find similar or related statements that we've talked about in lecture or tutorial, and did you try to understand how the problems were similar and how they were different?
- Did you ask a question on discord that required you to communicate what you understood and what was still confusing?
- Did you answer a peer's question on discord?
- Do you feel more comfortable using Python data structures like lists and dictionaries?
- Do you feel more comfortable using Functions, scopes, local/global variables in Python?
- Did you trace your code/algorithm and make sure it follows correctly and produce correct results?
- Did you try to identify the stages of the programming life cycle while you were solving a problem for this assignment?
- Do you feel more comfortable with identifying the errors in your code and handling those properly?
- For the given assignment, which letter grade would you give yourself (A+, …, F)?
- In 2-3 sentences, justify your letter grade selection.

Note: only the last two questions are mandatory for this quiz.

Topics: Functions, Modules, File I/O, Data Structures: Lists and Dictionaries

First part – the Python module: in the first part of this assignment, you will write a Python module to provide functionality for reading, analyzing, and displaying data stored on some external text (csv) files. The text files that we will be using in this assignment contain a simplified version of several Covid-19 test related data that have been reported in different Public Health Units (PHUs) in Ontario, Canada. These text files are available in a zipped file under the Assignment-3 section in Brightspace. Download the zipped file from Brightspace in a single folder. You should find a Python module called **a3.py** and the text files if you unzip the file. You are expected to add (and test) several functions to **a3.py** that must follow the instructions described in Section 2 of this specification and produce the correct results.

Second part – the tester file: in the second part of this assignment, you will write a Python program that will import your a3 module and use the functionality of this module to analyze Covid-19 test data.

Please be sure to read through the entire assignment before you begin to design your algorithm and start to code it. You are required to present your algorithm if you seek any help from the TAs or the instructor.

You must follow the naming conventions and all function signatures precisely. This means, your functions must

- have the names exactly as they are specified in this specification
- accept the right types of input parameters; you may rename the parameters, but they should appear in the correct sequence
- have brief docstrings
- return the correct values

In addition, you must

- use the main() function correctly to call other functions
- not call a function or use the print() function outside the main function
- not use any global variable
- add your name and student id at the top of every py file

Therefore, you should have print statements only where it is explicitly mentioned in the assignment. If you have additional print statements (for testing), make sure they are all converted to comments before you submit the assignment.

1. Text File Format

Each of the Covid-19 test data files included in the folder is a csv file (comma separated value file; it can be opened with a spreadsheet or a text editor) and has the following structure:

- Row -1 provides the headers for each column (the *header* row).
- Each subsequent row of the file stores a single Covid-19 test record.
- Every value on each row is separated by a comma (‘,’).
- Each row has ten (10) columns (*starting at index 0 and ending at index 9*).
- These columns store the following data in the given sequence.

0. Case_Reported_Date: the date (yyyy-mm-dd) when this case was first reported
1. Age_Group: the client’s age belongs to this group
2. Client_Gender: the client’s gender
3. Reporting_PHU_ID: the official id (a 4-digit number) of the PHU
4. Reporting_PHU: name of the PHU
5. Reporting_PHU_City: the city where this PHU is located
6. Reporting_PHU_Postal_Code: the postal code of the PHU
7. Reporting_PHU_Website: PHU’s website
8. Reporting_PHU_Latitude: PHU’s location
9. Reporting_PHU_Longitude: PHU’s location

You may assume that these files are properly formatted, i.e., that the 0’th column will always have the Case_Reported_Date, the 1’st column will always have the Age_Group, and so on.

2. A3.py Module [40 marks]

This section describes the required functions that you must add to your module. You are encouraged to create additional helper functions where necessary. You must test these functions using different text files inside the **`main()`** function. The final version of your module must only contain functions and a single call to the **`main()`** function (i.e., there should not be any additional function calls or print statements outside the **`main()`** function).

(a) `display_2Dlist()` that takes a 2D-list as input parameter and prints all elements of that list in the following format. If the list is empty, print “`No data in list`”.

```
>>> alist = [[1,2],[3,4],[5],[6,7]]
>>> display_2Dlist(alist)
[1,2]
[3,4]
[5]
[6,7]
```

(b) `display_dict()` that takes a dictionary as the input parameter and prints all keys and values of the dictionary in a sorted sequence of keys. If the dictionary is empty, print "**`Dictionary is empty`**".

```
>>> adict = {2:6, 7:1, 4:3, 5:"zero"}
>>> display_dict(adict)
2: 6
4: 3
5: zero
7: 1
```

(c) `return_list()` that takes the name of a csv file as the input parameter and returns a 2D list. I will call this 2D list `database` in the rest of this specification, but you may rename it. Each element of `database` must contain data from each row of the csv file as follows.

```
[   [Case_Reported_Date1,   Age_Group1,   Client_Gender1,   Reporting_PHU_ID1],
[Case_Reported_Date2, Age_Group2, Client_Gender2, Reporting_PHU_ID2], … ]
```

Note: It should not include the header row from the file. You will need to remove the 's' from the values of the `Age_Group` field, you can still store them as strings. You must store the `Reporting_PHU_IDs` as integers.

See sample output below.

```
>>> database = return_list("data10.csv")
>>> display_2Dlist(database)
['2021-05-19', '40', 'MALE', 2241]
['2021-05-22', '40', 'MALE', 2265]
['2021-05-19', '60', 'FEMALE', 2244]
['2021-05-20', '60', 'MALE', 4913]
['2021-05-19', '70', 'FEMALE', 2251]
['2021-05-19', '<20', 'FEMALE', 2244]
['2021-05-19', '<20', 'GENDER DIVERSE', 3895]
['2021-05-20', '<20', 'MALE', 2237]
['2021-05-19', '<20', 'MALE', 2268]
['2021-05-19', '20', 'MALE', 2270]
['2021-05-21', '<20', 'FEMALE', 2253]
['2021-05-19', '30', 'UNSPECIFIED', 3895]
```

(d) `return_dict()` that takes the name of a csv file as the input parameter and returns a dictionary, where each key is the Reporting_PHU_ID and the value is a list containing the following data (I will call this dictionary `dict,` you may rename it).

```
{Reporting_PHU_ID1:  [Reporting_PHU1,  Reporting_PHU_City1,
Reporting_PHU_Postal_Code1,  Reporting_PHU_Website1,  Reporting_PHU_Latitude1,
Reporting_PHU_Longitude1 ],
Reporting_PHU_ID2:  [Reporting_PHU2,  Reporting_PHU_City2,
Reporting_PHU_Postal_Code2,  Reporting_PHU_Website2,  Reporting_PHU_Latitude2,
Reporting_PHU_Longitude2 ], ...  }
```

See a sample output below.

```
>>> dict = return_dict(database)
>>> dict = return_dict("data10.csv")
>>> display_dict(dict)
2237:  ['Hamilton  Public  Health  Services',  'Hamilton',  'L8P  4S6',
'www.hamilton.ca/publichealth', '43.3', '-79.9']
2241: ['Kingston Frontenac and Lennox & Addington Public Health', 'Kingston', 'K7M
1V5', 'www.kflaph.ca', '44.2', '-76.5']
2244: ['Middlesex-London Health Unit', 'London', 'N6A 5L7', 'www.healthunit.com',
'43.0', '-81.3']
2251: ['Ottawa Public Health', 'Ottawa', 'K2G 6J8', 'www.ottawapublichealth.ca',
'45.3', '-75.8']
2253:  ['Peel  Public  Health',  'Mississauga',  'L5W  1N4',
'www.peelregion.ca/health/', '43.6', '-79.7']
2265:  ['"Region  of  Waterloo',  '  Public  Health"',  'Waterloo',  'N2J  4V3',
'www.regionofwaterloo.ca', '43.5']
2268: ['Windsor-Essex County Health Unit', 'Windsor', 'N9A 4J8', 'www.wechu.org',
'42.3', '-83.0']
2270:  ['York  Region  Public  Health  Services',  'Newmarket',  'L3Y  6Z1',
'www.york.ca/wps/portal/yorkhome/health/', '44.0', '-79.5']
3895: ['Toronto Public Health', 'Toronto', 'M5B 1W2', 'www.toronto.ca/community-
people/health-wellness-care/', '43.7', '-79.4']
4913:  ['Southwestern  Public  Health',  'St.  Thomas',  'N5P  1G9',
'www.swpublichealth.ca', '42.8', '-81.2']
```

(e) `get_total_cases()` takes the a 2D-list (similar to `database`) and an integer x from this set {0, 1, 2} as input parameters. Here, 0 represents **Case_Reported_Date**, 1 represents **Age_Group** and 2 represents **Client_Gender** (these are the fields on the *header* row, the integer value represents the index of each of these fields on that row). This function computes the total number of reported cases for each instance of x in the text file, and it stores this information in a dictionary in this form `{an_instance_of_x : total_case}`. Finally, it returns the dictionary and the total number of all reported cases saved in this dictionary.

(Suppose we want to know the total number of cases reported on each date, so use x = 0.)

```
>>> result, total_cases = get_total_cases(database, 0)
>>> display_dict(result)
2021-05-19: 8
2021-05-20: 2
2021-05-21: 1
2021-05-22: 1
>>> print(total_cases)
```

```
12
```

The sample output for x = 1 (Age_Group)

```
>>> result, total_cases = get_total_cases(database, 1)
>>> display_dict(result)
20: 1
30: 1
40: 2
60: 2
70: 1
<20: 5
>>> print(total_cases)
12
```

The sample output for x = 2 (Client_Gender)

```
>>> result, total_cases = get_total_cases(database, 2)
>>> display_dict(result)
FEMALE: 4
GENDER DIVERSE: 1
MALE: 6
UNSPECIFIED: 1
>>> print(total_cases)
12
```

(f) `display_PHU_summary()` takes three input parameters, a 2D-list (similar to `database`), a dictionary (similar to `dict`), and an integer for the PHU_ID. Your function must print the following information, the id, name, city, website, and the total number of cases reported at this PHU. If the given PHU_ID is invalid, it should print a message and exit the function.

```
>>> display_PHU_summary(database, dict, 2241)
PHU id: 2241
PHU name: Kingston Frontenac and Lennox & Addington Public Health
PHU city: Kingston, K7M 1V5
PHU website: www.kflaph.ca
Total cases: 1

>>> display_PHU_summary(database, dict, 3895)
PHU id: 3895
PHU name: Toronto Public Health
PHU city: Toronto, M5B 1W2
PHU website: www.toronto.ca/community-people/health-wellness-care/
Total cases: 2

>>> display_PHU_summary(database, dict, 3800)
The PHU Id is invalid
```

(g) `get_cases_by_PHU_and_age()` takes a 2D-list (similar to `database`) and a dictionary (similar to `dict`) as input parameters. This function must return a dictionary that contains information in the following format.

```
{
PHU_ID1: [ [Age_Group1, total number of cases reported for Age_Group1],
[Age_Group2, total number of cases reported for Age_Group2], …].

PHU_ID2: [ [Age_Group1, total number of cases reported for Age_Group1],
[Age_Group2, total number of cases reported for Age_Group2], …].

… }
```

If a PHU does not have any cases recorded for a particular age group, then data for this age group should not be included in the output. See the sample output for an example.

```
>>> result = get_cases_by_PHU_and_age(database, dict)
>>> display_dict(result)
2237: [['<20', 1]]
2241: [['40', 1]]
2244: [['60', 1], ['<20', 1]]
2251: [['70', 1]]
2253: [['<20', 1]]
2265: [['40', 1]]
2268: [['<20', 1]]
2270: [['20', 1]]
3895: [['<20', 1], ['30', 1]]
4913: [['60', 1]]
```

(h) `get_topx_hotspots()` that takes the 2D-list (similar to `database`), a dictionary (similar to `dict`) and an integer **`x`** as input parameters. This function computes the total number of cases reported in each PHU, and returns a dictionary containing the top **x** hotspots (x number of PHUs with the largest number of reported cases). Each **key:value** pair in this resulting dictionary represents a Reporting_PHU_ID as the key and the value is the total numbers of cases reported in this PHU.

The result does not need to be sorted, the PHUs of the resulting dictionary can appear in any order as long as the top-x PHUs have been identified correctly. If the value of x is 0 or less, or more than the total number of unique IDs, your function should print a message and return an empty dictionary. See sample outputs below.

```
>>> hotspots = get_topx_hotspots(database, dict, -1)
>>> display_dict(hotspots)
The value of x is too low or too high.
Dictionary is empty
```

```
>>> hotspots = get_topx_hotspots(database, dict, 3)
>>> display_dict(hotspots)
2241: ['Kingston Frontenac and Lennox & Addington Public Health', 'Total cases:
1']
2244: ['Middlesex-London Health Unit', 'Total cases: 2']
3895: ['Toronto Public Health', 'Total cases: 2']


>>> hotspots = get_topx_hotspots(database, dict, 5)
>>> display_dict(hotspots)
2241: ['Kingston Frontenac and Lennox & Addington Public Health', 'Total cases:
1']
2244: ['Middlesex-London Health Unit', 'Total cases: 2']
2265: ['"Region of Waterloo', 'Total cases: 1']
3895: ['Toronto Public Health', 'Total cases: 2']
4913: ['Southwestern Public Health', 'Total cases: 1']
```

(i) `main():` a main() function has been supplied with this module to test your functions. All sample outputs provided in this specification use data10.csv file. You may add your test cases to this function using different text files. The main guard has also been added to ensure that the `main()` function does not get executed if your program is imported as a module in some other program.

(j) **Bonus [5 marks].** What other interesting results we might want to extract from this dataset? If you have an idea then add a function named `my_function_rocks()` to your module that would serve this purpose. You can specify any input parameters and the returned values for this function. Be sure to add a detailed description of what your function does in its docstrings. Your function must modify the existing data and/or use operations in a way that we did not cover in any other functions (i.e., this cannot be a slightly different version of an existing function). And this function should not perform any tasks that can be accomplished by a built-in python method. This function can make use of any other function(s) of your module.

3. a3_tester.py [20 marks]

Write a python file named **a3_tester.py** that provides the following functionality.
a) imports a3 module
b) uses a main() function as the starting point of your program and to control the flow of your program, calls the main() function with a main-guard
c) shows the following menu to the user and repeatedly asks for a valid choice (an integer from 1-8) until the user chooses to quit.

```
1. Load file
2. Display database
3. Display dictionary
4. Show total cases by dates (0), age groups (1), genders (2)
5. Display data for a PHU
6. Show total cases in PHUs by age groups
7. Show total cases for top-x hotspots
8. Quit the program
```

The descriptions of what each option will do are the following.

1. Load file (takes a valid datafile name, calls two functions return_list() and return_dict() with the required arguments and stores the returned values)
2. Display database (prints the database using display_list2D() function with the required arguments)
3. Display dictionary (prints the dictionary using display_dict() function with the required arguments)
4. Show total cases by dates (0), age groups (1), genders (2) (calls get_total_cases() function with the required arguments)
5. Show data for a PHU (takes an integer value from the user to call display_PHU_summary() function with the required arguments)
6. Show the cases by age groups for all PHUs (calls get_cases_by_PHU_and_age() function with the required arguments)
7. Show cases for top-x hotspots (takes an integer value from the user and call get_topx_hotspots() function with the required arguments)
8. Quit the program (terminates the program when the user selects this option)

All invalid user inputs must be identified, and the prompt should be repeated. See a sample output at the end of this specification.

Some considerations

- Your output should be neatly formatted (close to the sample).
- You cannot assume that the user enters good input. That is, they may enter strings or numbers, and integer values outside the acceptable range. All these cases must be handled by your program by printing appropriate error messages and repeating the menu until a proper value is entered.
- You are encouraged to create additional helper functions where necessary.

Invalid submissions and penalties

- Submissions with an incorrect file name or format will receive a 10% penalty, with no exceptions. You are responsible for following the guidelines outlined in this assignment and the course outline.
- Add your name and id number as comments on top of each py file. If this info is not present, a 10% penalty will be applied.
- Submissions that crash (i.e., terminate with an error) on execution may receive a mark of zero (0).
- Each function that does not have a docstring will receive -1 mark.

A3.zip Submission Recap

You must submit the following python scripts with A3.zip

1. a3.py – the completed version of this module
2. a3_tester.py
3. all text files that are supplied with this assignment specification

Sample output of the tester file.

```
1.      Load file
2.      Display database
3.      Display dictionary
4.      Show total cases by dates (0), age groups (1), genders (2)
5.      Display data for a PHU
6.      Show total cases in PHUs by age groups
7.      Show total cases for top-x hotspots
8.      Quit the program

Please select a number between 1 - 8. >> 1
Enter a file name: data25.csv
Data loaded successfully.

1.      Load file
2.      Display database
3.      Display dictionary
4.      Show total cases by dates (0), age groups (1), genders (2)
5.      Display data for a PHU
6.      Show total cases in PHUs by age groups
7.      Show total cases for top-x hotspots
8.      Quit the program

Please select a number between 1 - 8. >> 2
['2021-05-19', '40', 'MALE', 2241]
['2021-05-19', '40', 'MALE', 2265]
['2021-05-19', '40', 'MALE', 3895]
['2021-05-19', '50', 'FEMALE', 2236]
['2021-05-19', '50', 'FEMALE', 2251]
['2021-05-19', '50', 'FEMALE', 3895]
['2021-05-19', '50', 'MALE', 2236]
['2021-05-19', '50', 'MALE', 2244]
['2021-05-19', '50', 'MALE', 2244]
['2021-05-19', '50', 'MALE', 2268]
['2021-05-19', '60', 'FEMALE', 2244]
['2021-05-19', '60', 'MALE', 4913]
['2021-05-19', '70', 'FEMALE', 2251]
['2021-05-19', '70', 'MALE', 4913]
['2021-05-19', '<20', 'FEMALE', 2244]
['2021-05-19', '<20', 'FEMALE', 2265]
['2021-05-19', '<20', 'FEMALE', 2253]
['2021-05-19', '<20', 'GENDER DIVERSE', 3895]
['2021-05-19', '<20', 'MALE', 2237]
['2021-05-19', '<20', 'MALE', 2268]
['2021-05-19', '20', 'MALE', 2270]
['2021-05-19', '<20', 'FEMALE', 2253]
['2021-05-19', '<20', 'MALE', 2253]
['2021-05-19', '30', 'UNSPECIFIED', 3895]
['2021-05-19', '60', 'FEMALE', 2253]
```

```
1.      Load file
2.      Display database
3.      Display dictionary
4.      Show total cases by dates (0), age groups (1), genders (2)
5.      Display data for a PHU
6.      Show total cases in PHUs by age groups
7.      Show total cases for top-x hotspots
8.      Quit the program

Please select a number between 1 - 8. >> 3
2236: ['Halton Region Health Department', 'Oakville', 'L6M 3Ll', 'www.halton.ca/For-
Residents/Public-Health/', '43.4', '-79.7']
2237: ['Hamilton Public Health Services', 'Hamilton', 'L8P 4S6',
'www.hamilton.ca/publichealth', '43.3', '-79.9']
2241: ['Kingston Frontenac and Lennox & Addington Public Health', 'Kingston', 'K7M
1V5', 'www.kflaph.ca', '44.2', '-76.5']
2244: ['Middlesex-London Health Unit', 'London', 'N6A 5L7', 'www.healthunit.com',
'43.0', '-81.3']
2251: ['Ottawa Public Health', 'Ottawa', 'K2G 6J8', 'www.ottawapublichealth.ca',
'45.3', '-75.8']
2253: ['Peel Public Health', 'Mississauga', 'L5W 1N4', 'www.peelregion.ca/health/',
'43.6', '-79.7']
2265: ['"Region of Waterloo', ' Public Health"', 'Waterloo', 'N2J 4V3',
'www.regionofwaterloo.ca', '43.5']
2268: ['Windsor-Essex County Health Unit', 'Windsor', 'N9A 4J8', 'www.wechu.org',
'42.3', '-83.0']
2270: ['York Region Public Health Services', 'Newmarket', 'L3Y 6Z1',
'www.york.ca/wps/portal/yorkhome/health/', '44.0', '-79.5']
3895: ['Toronto Public Health', 'Toronto', 'M5B 1W2', 'www.toronto.ca/community-
people/health-wellness-care/', '43.7', '-79.4']
4913: ['Southwestern Public Health', 'St. Thomas', 'N5P 1G9',
'www.swpublichealth.ca', '42.8', '-81.2']


1.      Load file
2.      Display database
3.      Display dictionary
4.      Show total cases by dates (0), age groups (1), genders (2)
5.      Display data for a PHU
6.      Show total cases in PHUs by age groups
7.      Show total cases for top-x hotspots
8.      Quit the program

Please select a number between 1 - 8. >> 4
Enter an index: 1
Number of cases for each group:
20: 1
30: 1
40: 3
50: 7
```

```
60: 3
70: 2
<20: 8
Total cases: 25


1.      Load file
2.      Display database
3.      Display dictionary
4.      Show total cases by dates (0), age groups (1), genders (2)
5.      Display data for a PHU
6.      Show total cases in PHUs by age groups
7.      Show total cases for top-x hotspots
8.      Quit the program

Please select a number between 1 - 8. >> 5
Enter a PHU Id: 3895
PHU id: 3895
PHU name: Toronto Public Health
PHU city: Toronto, M5B 1W2
PHU website: www.toronto.ca/community-people/health-wellness-care/
Total cases: 4


1.      Load file
2.      Display database
3.      Display dictionary
4.      Show total cases by dates (0), age groups (1), genders (2)
5.      Display data for a PHU
6.      Show total cases in PHUs by age groups
7.      Show total cases for top-x hotspots
8.      Quit the program

Please select a number between 1 - 8. >> 6
2236: [['50', 2]]
2237: [['<20', 1]]
2241: [['40', 1]]
2244: [['50', 2], ['60', 1], ['<20', 1]]
2251: [['50', 1], ['70', 1]]
2253: [['<20', 3], ['60', 1]]
2265: [['40', 1], ['<20', 1]]
2268: [['50', 1], ['<20', 1]]
2270: [['20', 1]]
3895: [['40', 1], ['50', 1], ['<20', 1], ['30', 1]]
4913: [['60', 1], ['70', 1]]


1.      Load file
2.      Display database
3.      Display dictionary
4.      Show total cases by dates (0), age groups (1), genders (2)
```

```
5.      Display data for a PHU
6.      Show total cases in PHUs by age groups
7.      Show total cases for top-x hotspots
8.      Quit the program

Please select a number between 1 - 8. >> 7
Enter the number of hotspots: 4
2244: ['Middlesex-London Health Unit', 'Total cases: 4']
2253: ['Peel Public Health', 'Total cases: 4']
2265: ['"Region of Waterloo', 'Total cases: 2']
3895: ['Toronto Public Health', 'Total cases: 4']


1.      Load file
2.      Display database
3.      Display dictionary
4.      Show total cases by dates (0), age groups (1), genders (2)
5.      Display data for a PHU
6.      Show total cases in PHUs by age groups
7.      Show total cases for top-x hotspots
8.      Quit the program

Please select a number between 1 - 8. >> 8
Thank you for using this program.
```