CSCI 509 Python Programming Fall 2019

Final Exam

Overview

This assignment will give you more experience on the use of:
•lists
•dictionaries
•data structures
•functions
•iteration
•data analysis

The goal of this project is to analyze data relevant to the biggest cybersecurity breaches in the past.

Background

Cybersecurity breaches are unfortunately becoming more and more common. These can sometimes be attributed to clever, sophisticated attacks and other times to lax security. However, the result is the same – records are lost such that the confidentiality, integrity, and availability of sensitive information is affected.

Description

This exam focuses on analyzing a publicly available dataset containing information about some infamous breaches that have occurred in the past.

```
open_file(message) -> fp
```

This function taken in a prompt message (string) and returns a file pointer. You likely have acopy from a previous project. It repeatedly prompts for a file until one is successfully opened. It should have a try-except statement. By default (when the user does not provide a filename), this function opens the file **'breachdata.csv'**. Important: remember to open the file with thecorrect encoding as shown below.

```
fp = open(filename, encoding='utf8')
```

`build_dict(reader) -> dictionary`

This function accepts a CSV reader as input and returns the required dictionary. It iterates over the CSV reader and with each iteration, extracts the needed data and remove any extra whitespaces. You need to skip the header line before start reading the data:

```
next(reader,None)
```

The data to be extracted is:

entity - index 0 (string)
records lost - index 2 (int)
year - index 3 (int)
story - index 4 (string)
sector – index 5 (string)
method - index 6 (string)
news sources - index 11 (list of strings)

Valid data checks:
• _If there are multiple news sources at index 11, they will be separated by a comma (‘,’). These should all be stored in a list. If the news sources field is empty ignore that line.
• _Treat all missing numeric values of “records lost” as a 0. You need to remove all occurrences of ‘,’ in the records lost before converting to integer. Hint: use .replace()
• _If any of these pieces of data is missing (other than "records lost"), e.g. the field is empty or spaces, ignore that line of data.
• _If the year is invalid (i.e. not an int), ignore that line.

The overall data structure is a master dictionary. This master dictionary has key ‘entity’ (string), and value as a list. This list contains tuples for each instance of this entity observed in the data file. For example, AOL occurs 3 times in the **breachdata.csv** CSV file, so there are three tuples within the list for AOL. These 3 tuples are color coded below.

```
{‘AOL’:[
({'AOL': (92000000, 2004, 'Jun 2004. A former America Online software
engineer stole 92 million screen names and e-mail addresses and sold them to
spammers who sent out up to 7 billion unsolicited e-mails.', ['CNN'])},
{2004: ('web', 'inside job')}),
({'AOL': (20000000, 2006, 'Aug 2006. Durp. AOL VOLUNTARILY released search
data for roughly 20 million web queries from 658,000 anonymized users of the
service. No one is quite sure why.', ['Tech Crunch'])}, {2006: ('web',
'oops!')}),
({'AOL': (2400000, 2014, "Apr 2014. Users' accounts were compromised to send
out spam messages.", ['NBC News'])}, {2014: ('web', 'hacked')})
]}
```

Within each tuple are two dictionaries. The first dictionary has the key ‘entity’ (string) and its value is a tuple that contains (records lost(int), year(int), story(string), news_source(list of strings)). The second dictionary within this tuple has its key as year(int) and value as a tuple that contains (sector(string), method(string)). Let’s assume that D1 and D2 are the two dictionaries, then the values of the master dictionary are [(D1,D2)]

`top_rec_lost_by_entity(dictionary) -> list`

This function accepts the breach dictionary as created by the build_dict function above and returns a sorted list (in descending order of records lost) of the top 10 entities that lost the most records. To break ties, the returned list should be sorted by entity alphabetically. The returned list will contain 10 tuples, each tuple containing the entity name and total records lost by that entity. For example, in the case of AOL, the tuple will be as follows:

```
('AOL', 114400000)
```

From the AOL entry shown for build_dict you can calculate that AOL lost a total of (92000000+20000000+2400000) 114400000 records.
Hint: use itemgetter from the operator module and the key argument to the sort routines on index 1 of the tuple.

`records_lost_by_year(dictionary) -> list`

This function accepts the breach dictionary and produces a sorted list of total records lost within each year. This is similar to the previous function, except that instead of counting records lost by entities, we are counting records lost by year. Also, instead of returning only the top 10, this function should return all records by year. The list will contain tuples such that each tuple contains the year and the corresponding total records lost in that year. For example:

```
(2018, 4062466578)
```

The list should be sorted in descending order by total records lost and by year (to break ties)
Hint: you could start by creating a dictionary where ‘year’ is the key and the record lost for that year are the values.

`top_methods_by_sector(dictionary) -> dictionary`

This function takes in the breach dictionary and returns a dictionary of dictionaries. The returned dictionary contains keys for all sectors found in the breach dictionary (e.g. ‘web’, ‘tech’ etc.). The values of this dictionary are dictionaries that contain the attack vector (method) as key and the corresponding count as value. For example,

{‘web’:{'inside job': 1, 'oops!': 2, 'hacked': 81, 'unknown': 1, 'poor security': 12}}

Within the ‘web’ sector, we have counted 81 instances of the method ‘hacked’, 12 instances of ‘poor security’ etc. The returned dictionary should be sorted in ascending order by the sector name. Hint: Remember that dictionaries are insertion ordered. Insertion Ordered means the order in which we are inserting the data in the dictionary is maintained.

`top_rec_lost_plot(names,records)`

This function is provided. It plots a bar graph for both top records lost by entities and the total records lost by year when the relevant datasets are provided correctly. Call this with the appropriate data within main().

`top_methods_by_sector_plot(methods_list)`

This function is provided. It plots a pie chart for distribution of methods of attack used within each sector. Call this within main() with the appropriate input.

`main()`

This function prints provided BANNER and MENU and then asks the user to make a choice between the various available options shown in the menu. If the choice is 1,2,3, or 4, and it is the first time asking for a choice, it calls the open_file() function with the appropriate message string to obtain a handle to the file pointer. Next, csv.reader is used to read data from the file pointer. This reader is then sent to build_dict() function to build the breach dictionary.

1. **Option 1**: If the choice is 1, the program calculates the top 10 entities that lost the most records by calling the function top_rec_lost_by_entity() and displays the results to the user in descending order Each displayed row contains the rank, the entity and the number of records lost. Each row is separated by dashed line containing 45 dashes (“-“*45). Use the following string formatting for each row:
"[{:2d}] | {:15.10s} | {:10d}"

It then asks the user if they want to plot the results. Depending on user’s response, plot can be displayed. The corresponding plot function, top_rec_lost_plot(), expects two lists as function input, the first list is the names of all entities in descending order of most records lost, the second list contains the corresponding quantities. For example:
names = ['Aadhaar', 'Yahoo', 'River City Media', …]
records = [2100000000, 1532000000, 1370000000, …]

2. **Option 2**: If the choice is 2, the program calculates the records lost in each year by calling the function records_lost_by_year() and displays the results to the user in descending order. Each displayed row contains the rank, the year (string) and the number of records lost. Each row is separated by dashed line containing 45 dashes (“-“*45). Use the following string formatting for each row:

"[{:2d}] | {:15.10s} | {:10d}"

It then asks the user if they want to plot the results. Depending on user's response, plot can be displayed. The corresponding plot function, top_rec_lost_plot(), expects two lists as function input, the first list is the years in descending order of most records lost, the second list contains the corresponding quantities. For example:
years = [2018, 2017, 2019, …]
records = [4062466578, 2455191309, 2016515298, …]

3. **If the choice is 3**, the program calculates the top methods used for breaching each sector by calling the function top_methods_by_sector() and displays the names of sectors discovered separated by a space. It then asks the user to input a sector name. The program should keep asking for a valid sector name. Once valid, the methods and their counts pertaining to this sector are then displayed to the user in descending order by count. Each displayed row contains the rank, the method and its counts. Each row is separated by dashed line containing 45 dashes ("-"*45). Use the following string formatting for each row:

"[{:2d}] | {:15.10s} | {:10d}"

Finally, the program asks if the user wants to plot the results. Results are plotted by the top_methods_by_sector_plot()function based on the user's response.

4. **If the choice is 4**, the user is asked to input the name of an entity (e.g. AOL). The user is then shown all stories discovered within the breach dictionary pertaining to that entity. If an incorrect entity name is entered (one that is not found in the dictionary) then an error message is displayed and user is asked to enter the entity name again.

Note that the overall data structure is a master dictionary. This master dictionary has key 'entity' (string), and value as a list. This list contains tuples for each instance of this entity observed in the data file including stories. For example, AOL occurs 3 times in the breachdata.csv CSV file.

```
{'AOL':[
({'AOL': (92000000, 2004, 'Jun 2004. A former America Online software engineer stole 92 million screen names and e-mail addresses and sold them to spammers who sent out up to 7 billion unsolicited e-mails.', ['CNN'])}, {2004: ('web', 'inside job')}),
({'AOL': (20000000, 2006, 'Aug 2006. Durp. AOL VOLUNTARILY released search data for roughly 20 million web queries from 658,000 anonymized users of the service. No one is quite sure why.', ['Tech Crunch'])}, {2006: ('web', 'oops!')}),
({'AOL': (2400000, 2014, "Apr 2014. Users' accounts were compromised to send out spam messages.", ['NBC News'])}, {2014: ('web', 'hacked')})
]}
```

Your program should output 3 stories:
[+] Found 3 stories:
[+] Story 1: Jun 2004. A former America Online software engineer stole 92 million screen names and e-mail addresses and sold them to spammers who sent out up to 7 billion unsolicited e-mails.
[+] Story 2: Aug 2006. Durp. AOL VOLUNTARILY released search data for roughly 20 million web queries from 658,000 anonymized users of the service. No one is quite sure why. [+] Story 3: Apr 2014. Users' accounts were compromised to send out spam messages.

5. **If the choice is 5**, the program prints the closing message and exits.

Note: the program needs to catch exceptions in all the input sequences. For example, if the choice is not 1,2,3,4 or 5, then an error message is displayed, and user is asked for input again. Similarly, for sector name in choice 3, if the user enters an invalid sector name then an error message is displayed, and the user is asked to input the sector name again. Also, for entity name in choice 4, if the user enters an invalid entity name then an error message is displayed, and the user is asked to input the entity name again.

Deliverables

The deliverable for the midterm is the following file:

finalexam.py – the source code for your Python program

Be sure to use the specified file name ("**finalexam.py**") and to submit it along with the screenshots of your output to Blackboard.

Notes

1. Use itemgetter() from the operator module to specify the key for sorting.

Sample output:

Test 1:
```
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 1
[ ? ] Enter the file name:
[ + ] Most records lost by entities...
---------------------------------------------
[ 1 ] | Aadhaar | 2100000000
---------------------------------------------
[ 2 ] | Yahoo | 1532000000
---------------------------------------------
[ 3 ] | River City | 1370000000
---------------------------------------------
[ 4 ] | First Amer | 885000000
---------------------------------------------
[ 5 ] | Spambot | 711000000
---------------------------------------------
[ 6 ] | Friend Fin | 412000000
---------------------------------------------
[ 7 ] | Marriott H | 383000000
---------------------------------------------
[ 8 ] | Twitter | 330250000
---------------------------------------------
[ 9 ] | MongoDB | 275265298
---------------------------------------------
[ 10 ] | Chinese re | 202000000
[ ? ] Plot (y/n)? n
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit

[ ? ] Choice: 5
[ + ] Done. Exiting now...
```

Test 2:

```
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 2
[ ? ] Enter the file name:
[ + ] Most records lost in a year...
---------------------------------------------
[ 1 ] | 2018 | 4062466578
---------------------------------------------
[ 2 ] | 2017 | 2455191309
---------------------------------------------
[ 3 ] | 2019 | 2016515298
---------------------------------------------
```

```
[ 4 ] | 2016 | 1801353869
---------------------------------------------
[ 5 ] | 2013 | 1653263579
---------------------------------------------
[ 6 ] | 2015 | 474847000
---------------------------------------------
[ 7 ] | 2014 | 328674396
---------------------------------------------
[ 8 ] | 2009 | 254152778
---------------------------------------------
[ 9 ] | 2012 | 232898177
---------------------------------------------
[ 10 ] | 2011 | 199841734
---------------------------------------------
[ 11 ] | 2007 | 150597405
---------------------------------------------
[ 12 ] | 2004 | 92000000
---------------------------------------------
[ 13 ] | 2008 | 88455500
---------------------------------------------
[ 14 ] | 2006 | 46825000
---------------------------------------------
[ 15 ] | 2005 | 44100000
---------------------------------------------
[ 16 ] | 2010 | 10149285 [ ? ] Plot (y/n)? n
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 5
[ + ] Done. Exiting now...
```

Test 3

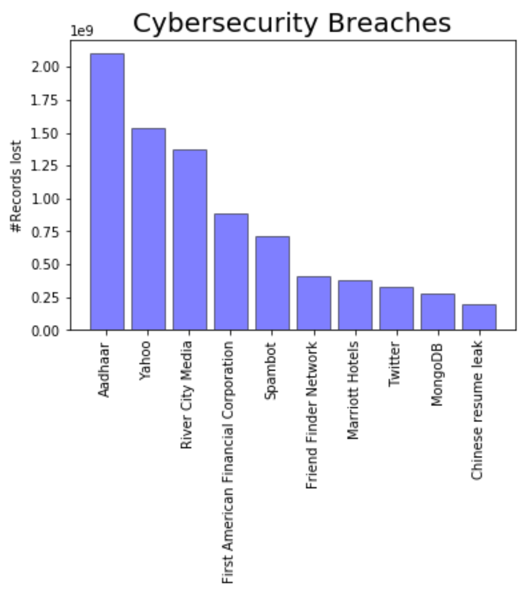
```
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 6
[ - ] Incorrect input. Try again.
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: youdoyou
[ - ] Incorrect input. Try again.
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: -435gh
[ - ] Incorrect input. Try again.
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 5
[ + ] Done. Exiting now...
```

Test 4:

```
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 4
[ ? ] Enter the file name:
[ ? ] Name of the entity (case sensitive)? theBeatles
[ - ] Entity not found. Try again.
[ ? ] Name of the entity (case sensitive)? Liverpool
[ - ] Entity not found. Try again.
[ ? ] Name of the entity (case sensitive)? AOL
[ + ] Found 3 stories:
[ + ] Story 1: Jun 2004. A former America Online software engineer
stole 92 million screen names and e-mail addresses and sold them to
spammers who sent out up to 7 billion unsolicited e-mails.
[ + ] Story 2: Aug 2006. Durp. AOL VOLUNTARILY released search data
for roughly 20 million web queries from 658,000 anonymized users of
the service. No one is quite sure why.
[ + ] Story 3: Apr 2014. Users' accounts were compromised to send out
spam messages.
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 5
[ + ] Done. Exiting now...
```

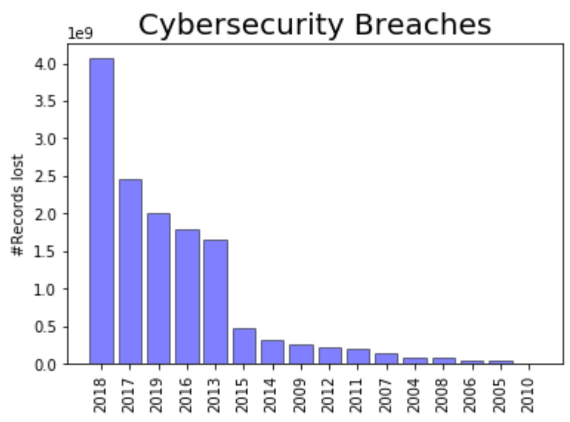
Test 5 (with plotting)

```
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 1
[ ? ] Enter the file name:
[ + ] Most records lost by entities...
---------------------------------------------
[ 1 ] | Aadhaar | 2100000000
---------------------------------------------
[ 2 ] | Yahoo | 1532000000
---------------------------------------------
[ 3 ] | River City | 1370000000
---------------------------------------------
[ 4 ] | First Amer | 885000000
---------------------------------------------
[ 5 ] | Spambot | 711000000
---------------------------------------------
[ 6 ] | Friend Fin | 412000000
---------------------------------------------
[ 7 ] | Marriott H | 383000000
---------------------------------------------
[ 8 ] | Twitter | 330250000
---------------------------------------------
[ 9 ] | MongoDB | 275265298
---------------------------------------------
[ 10 ] | Chinese re | 202000000
[ ? ] Plot (y/n)? y
```

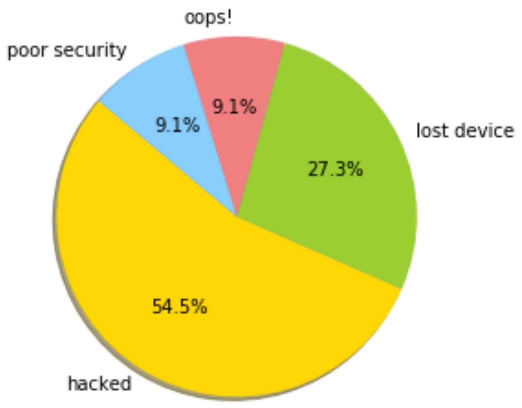


```
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 2
[ + ] Most records lost in a year...
---------------------------------------------
[ 1 ] | 2018 | 4062466578
---------------------------------------------
[ 2 ] | 2017 | 2455191309
---------------------------------------------
[ 3 ] | 2019 | 2016515298
---------------------------------------------
[ 4 ] | 2016 | 1801353869
---------------------------------------------
[ 5 ] | 2013 | 1653263579
---------------------------------------------
[ 6 ] | 2015 | 474847000
---------------------------------------------
[ 7 ] | 2014 | 328674396
---------------------------------------------
[ 8 ] | 2009 | 254152778
---------------------------------------------
[ 9 ] | 2012 | 232898177
---------------------------------------------
[ 10 ] | 2011 | 199841734

---------------------------------------------
[ 11 ] | 2007 | 150597405
---------------------------------------------
[ 12 ] | 2004 | 92000000
---------------------------------------------
[ 13 ] | 2008 | 88455500
---------------------------------------------
[ 14 ] | 2006 | 46825000
---------------------------------------------
[ 15 ] | 2005 | 44100000
---------------------------------------------
[ 16 ] | 2010 | 10149285 [ ? ] Plot (y/n)? y
```



```
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 3
[ + ] Loaded sector data.
academic app energy financial gaming government healthcare legal
media military retail tech telecoms transport web
[ ? ] Sector (case sensitive)? academic
[ + ] Top methods in sector academic
---------------------------------------------
[ 1 ] | hacked | 6
---------------------------------------------
[ 2 ] | lost devic | 3
---------------------------------------------
[ 3 ] | oops! | 1
---------------------------------------------
[ 4 ] | poor secur | 1
[ ? ] Plot (y/n)? y
```



```
[ 1 ] Most records lost by entities
[ 2 ] Records lost by year
[ 3 ] Top methods per sector
[ 4 ] Search stories
[ 5 ] Exit
[ ? ] Choice: 5
[ + ] Done. Exiting now...
```