The one where we help the police

Due 1 Nov by 23:59 **Points** 3 **Submitting** an external tool
Available until 1 Nov at 23:59

Successfully solving this problem will award the student **3 point** towards Examination 1

Requirements

- The script must pass all the automatic tests
- The script must pass a manual inspection
- **The student must write the script on his or her own**, any suspicion about cheating, cooperation, and/or plagiarism will be reported to the Discipline Board (see the **Study Guide (https://hkr.instructure.com/courses/5190/files/1027789/download?download_frd=1)** for more information). The students script will be checked both manually and using anti-plagiarism software

Instructions

1. Analyse the problem below using Polya's Problem solving Technique (https://math.berkeley.edu/~gmelvin/polya.pdf)
2. Create a script named **Lab3_3.py** that solves the problem according to the given instructions
3. Pay special attention when formatting the output as it must match the specified format **exactly** (character by character) to pass the tests
4. Upload the script using the link below

Task description

A lot of police investigations start off by examining where the potential suspects were at the

time of the crime. This is often done by asking phone companies to provide lists of all mobile phones that were in the area where the crime was committed around the time when it was committed. If there is only one crime scene this is not a big problem but if there are several crime scenes then the task of cross referencing the lists gets quite tedious. To combat this the local police department has decided to automate the process by creating a script that automatically does most of the work for them.

The script shall work as follows:

(Step 1) Start the script
(Step 2) Provide one or more paths to files with phone numbers
(Step 3) The script finds out what phone numbers occur in all files
(Step 4) The script looks up who owns the phone numbers found
(Step 5) The script displays the found names

To break it all down a little bit further this is what should happen during each step.

1. When starting the script, a path to a file that contains the mapping between name and number shall be given as a command line argument. This file shall contain a pickled dictionary on the following format {'0709-12345': 'Anna Karlsson'}.
2. To be able to provide an arbitrary number of files a menu is needed. The menu shall have the following options: *1. Add file* and *2. Calculate*. All file paths added shall be saved in a list for later use.
3. This step is part of the menu option *2. Calculate*. The script shall cross reference the numbers in all the files provided and create a Set with the ones that occur in all files. The files with the phone numbers store one phone number per line and is saved as a regular text file.
4. This step is part of the menu option *2. Calculate*. Using the Set of found numbers the script looks up the owner's names using the file sent as a command line argument. The names shall be saved in a list.
5. All the found names located in the set from Step 4 is displayed on the screen.

The student must implement the following functions:

display_menu()
This function shall display the following menu on screen:

```
1. Add file
2. Calculate
```

cross_reference(files)
The parameter *files* shall be a list of filenames (full path). The files shall be files containing one phone number per line. The function shall create, and return, a Set containing all the phone numbers that occurs in all files. If given an invalid filename the function shall raise a FileNotFoundException.

map_numbers_to_names(numbers, filename)
This function shall match each number from the Set *numbers* to a name and store the

names in a list. To accomplish this, the function shall use a dictionary that is loaded (using pickle) from the file *filename*. The dictionary shall use the format {'0709-12345': 'Anna Karlsson'}. If a number can not be found in the dictionary it shall be listed as Unknown with the phone number attached, like can be seen in this example: *Unknown (0709-11111).* The function shall return the list with the names and/or unknown tags. If given an invalid filename the function shall raise a FileNotFoundException.

display_suspects(names)
As the name of the function suggests this function shall display the names of all suspects sent to the function as a list of strings. The format shall be **exactly** as shown here:

```
The following persons was present on all crime scenes:
------------------------------------------------------
Max Olsson
Zara Walthersson
Unknown (0703-23464)
```

If the list sent as a parameter is empty the following message shall be shown:

```
The following persons was present on all crime scenes:
------------------------------------------------------
No matches
```

main()
This function shall have a loop that displayed the menu and handles the user input. Depending on the input from the user the function shall either read more filenames or try to find the suspects and print their names using the file from the command line argument for the mapping between numbers and names. Besides having the "main loop" this function shall also be responsible for the error handling in the script and print the error message below if the file given as command line arguments or the files specified while running the script are not valid

```
Error: There was a problem with at least one of the files.
```

To avoid problems with the testing framework, and to build a good habit, call the main function using the following construct:

```
if __name__ == '__main__':
    main()
```

Comments and additional resources

- Students are highly recommended to first solve all the tasks from the corresponding lectures and exercises before attempting to solve this problem. Having a solid plan before starting to code will also help
- The student can upload as many times as he or she likes to before the deadline

- These files (**owners (https://hkr.instructure.com/courses/5190/files/1017840/download?download_frd=1)** , crime scene 1 (https://gist.github.com/andreas-hkr/dd4c0e2c5aacccaea968a81787e3c978) , crime scene 2 (https://gist.github.com/andreas-hkr/f74d2fc0c9e65d9957639e867f4105e0) and crime scene 3 (https://gist.github.com/andreas-hkr/532a689feba30cf8f12c3e95bd71cf5d)) can be used as an example of how the file with owners mapped to phone numbers and the list of phone numbers from the crime scenes might look like.
- Watch this (https://youtu.be/lOeIDvyRUQs) video if you want to learn more about why we call main the way we do
- This video (https://youtu.be/o1Rxrtiqml8) covering how to use command line arguments might be useful.
- This video (https://youtu.be/w8CDujp40Gs) recorded while running the finished script showcases how the script shall look and work when finished.

This tool needs to be loaded in a new browser window

Load The one where we help the police in a new window