CSCI 1100 — Computer Science 1 Homework 6
Files and Sets

Homework Overview

Exterminate!

This homework is worth **110 points** toward your overall homework grade and is due **Thursday November 3, 2016 at 11:59:59 pm**. It consists of one single program.

Please carefully review the rules about excess collaboration from HW 3 to make sure you are turning in a program that is your own. We had some trouble on previous HW and we want to re-emphasize in particular that it is **NOT OK** to help your fellow students to a solution by showing them your solution or talking them through your approach and solution line-by-line or by changing their code yourself during the debugger process!

We strongly suggest that you study the example in file `example_list_of_lists.py` before you get started. Put in print statements and step through it in the debugger to see how it works.

As always, you are expected to match your output to the one given on the submission server. Write your program early, get it running and match the example output we give you in this document using your development environment and the Wing IDE. Then, once it is working correctly in your local environment, submit it to Submitty and verify you actually get the expected output. Remember that we may change the data on the submission server! It may not give the same results as an identical input on your local enviroment. Be sure that you follow the rules we lay out for numbering, sorting, and organization and you will get maximum points on the homework!

You are not expected to use dictionaries (covered in Lecture on Monday 10/31) to answer this homework, but you are allowed to. You must use **sets**. It will become quite obvious why as you start working on the program.

As mentioned in previous homeworks, your program must have the following basic structure (adherence to program structure and organization will be graded):

```
'''
   A short comment detailing the purpose of the program and major
   assumptions that you are making. Your name and a date may help you later.
'''

## all imports
import math ## shouldn't be necessary, but here is where it would go

## all functions first
def function1(x):
    '''
    Write a short comment here about what function1 does, including
    what it assumes about each parameter.
    '''
    return x + 1
```

```
def function2(y):
    '''
    Write a short comment here about what function2 does, including
    what it assumes about each parameter.
    '''
    return y -1

## all your main code after this if statement
if __name__ == '__main__':
    z = 10
    print(function1(z))
    print(function2(z))
```

This homework brings together many little things you learned in different parts of this class. But, we will not tell you how to do the whole thing. We have a solution that is about 60 lines including blank lines for readability and comments. It is not a long program if you plan it properly. We recommend constructing it in little pieces, similar to the way we've constructed examples in lecture.

A final word: The input file is from 2013, so it will not have the most recent information.

Problem: Six Degrees of Villainy

You are given a file called `DrWhoVillains.tsv`. It is a TAB separated file containing villains in a science fiction show called Dr. Who. The show is known for its extremely low budget and silly creature effects. Although the modern version is slicky produced and has a higher budget, it still pays homage to the original look and feel of the old villains. Look at the picture of a Dalek (one of the main villains) above to see this for yourself. Each line of `DrWhoVillains.tsv` contains information on a single villain. No villain is repeated on more than one line. Each line contains the following information (separated by TABs):

```
1. Villain: name of the villain (e.g. Dalek)
2. Year first: the first year this villain was introduced
3. Year last: the last year this villain was shown
4. Doc. no.: the list of doctor ids with this villain, separated by commas
5. Doctor actor: the list of actors for each doctor above
6. Episodes: number of episodes featuring this villain
7. Motivation (invasion earth, kill humans, etc): the description of the main
    motivation of this villain
8. Story titles: the titles of the stories that the villain was involved with,
    a list of strings separated by commas.
```

Note that some of the names of the villains are very long!

Your program must do the following:

1. Read through the file to find the villain names and the (set of) stories for each villain. **Be careful:** the file is manually created and may have repeated story titles, incorrect case, extra spaces before and after names, etc. We will guarantee that each villain only appears on one line and that there are no misspellings, but you must use consistent capitalization and sets to find unique stories.

2. Ask the user for the name of a "target villain" to try to reach: we are going to see how many steps it will take us to find this villain in a chain from one of the top ten most popular villains.

3. Find the top 10 most popular villains: those with the top 10 highest number of stories listed in the file and display them to the user, numbered 1 to 10, in decreasing order of popularity. When there is a tie, break it in reverse alphabetic order (starting at Z and going to A). These top ten villains become our initial `candidates`.

4. In a loop, do the following:

   (a) Display the candidate villains.

   (b) Ask the user for a villain number from the candidates list.

   (c) If the user enters something other than a number from 1 up to and including the number of candidates, the program displays the current list again and asks for another input.

   (d) If the user enters a valid option then check if the villain at the index (actually one less than the index) in the candidates list is the target villain. If so, report that the target villain has been found and exit. This output should include the number of steps to get to this villain. Incorrect input does not count as a step. If the villain is found right away it is 1 step.

   (e) Otherwise, print out how many stories this selected candidate villain was in and then generate new `candidate` villains by finding out all of those villains who appeared in a story with the currently selected candidate villain. Order the new candidate villains in the same way as the original villains.

Some Hints

We are not going to tell you exactly how to structure your code, but we are going to give you a few hints.

1. Parsing the villain file is an important of this HW. We suggest you create a function that takes a single string parameter corresponding to a single line of the input file, parses that line, and then returns a list containing:

   ```
   [number of shows, villain name, set of show titles]
   ```

   Think about this a little. It is not overly hard if you realize that we only need information from 2 of the 8 fields contained in the file. The remaining 6 fields – Year first, Year last, Doc. no., Doc. Actor, Episodes, and Motivation – are not used in this assignment. Take the line, split it at the TABS. Item[0] is the villain name and item[7] is a comma separated string that you can split and turn into a set of story titles.

   Test your parsing very thoroughly before proceeding. If you come to use for help after you believe you have parsing working, you must be prepared to demonstrate to us that it is correct!

2. If you maintain a list of lists, for example if you make a list of lists returned by the parsing function just described, it is simple to sort this list. As mentioned at the start of this assignment, check out `example_list_of_lists.py` for code that will help you with this. If you understand `example_list_of_lists.py` you will see how to easily do the sorting required by this project as well. We suggest you experiment by modifing that program to put all of the states at the same rank, and look at the sorted list before you try the sorting in your program.

3. Start small and build up. Make sure you can parse the lines correctly into a list first and test it. Then build up the list of lists. Then sort it. Then figure out how to generate the list of villains, etc. Once your program is working correctly, get the formatting right.

4. Lab 8 touches on some of the same topics and requires many of the same skills required here. If you are having problems, work through the material in Lab 8. It will help.

Deliverables – Final Check

Here is a summary of all the requirements for this homework. Double check this before you submit:

- Your program must use **sets**, this will also help you quite a lot.
- Submit a single file called `hw6.py` that assumes the existence of a text file called `DrWhoVillains.tsv`.
- The main loop of the program should be able to handle incorrect entry, variable number of selections, and exit when you reach the target villain.
- Print all input that you read immediately.
- Your program should have the required structure.

Expected Output

Below you can see the expected functioning of this program with the file we gave you (note: we might change the file in the homework submission server):

```
Who are you trying to reach? Neo-Nazis, WindSOR
Neo-Nazis, WindSOR

1. Daleks
2. Cybermen
3. Master (the)
4. Dalek Supreme (Supreme Dalek), inc Progenitor
5. Sontarans
6. Ice Warriors (inc benevolent appearences)
7. Davros
8. Silurians (exc Sea Devils)
9. Kovarian, Madame
10. Cyber-leader
Enter a selection => 2
2

Cybermen appeared in 24 stories and overlapped with:

1. Daleks
2. Master (the)
3. Dalek Supreme (Supreme Dalek), inc Progenitor
4. Sontarans
5. Ice Warriors (inc benevolent appearences)
6. Silurians (exc Sea Devils)
7. Kovarian, Madame
```

```
8. Cyber-leader
9. Black Guardian (the)
10. Ood, the (Oodkind), inc benevolent appearances
11. Cybermat
12. Borusa
13. Yeti
14. Terrible Zodin
15. Rassilon
16. Lytton, Commander Gustav
17. Cyber-controller
18. Cult of Skaro (inc Dalek Sec and three others)
19. Atraxi
20. van Statten, Henry
21. War Lord
22. War Chief
23. Vaughn, Tobias
24. Raston Warrior Robot
25. Peinforte, Lady
26. Neo-Nazis, Windsor
27. Mawdryn
28. Manton, Colonel
29. Mailer, Harry
30. Lumic, John
31. Kellman
32. Keller Machine
33. Kalik
34. Headless Monks
35. Hartman, Yvonne
36. Hartigan, Mercy
37. Drashigs
38. De Flores
39. Cybershades
40. Cyberman Android
41. Alliance - onscreen - Daleks, Cybermen, Sontarans & Autons, (cameos Silurians, Sycorax,
Enter a selection => 38
38

De Flores appeared in 1 story and overlapped with:

1. Cybermen
2. Cyber-leader
3. Peinforte, Lady
4. Neo-Nazis, Windsor
Enter a selection => 4
4

You reached the villain Neo-Nazis, Windsor in 3 steps.
Have a nice day.
```

A second example:

```
Who are you trying to reach? Daleks
```

```
Daleks

1. Daleks
2. Cybermen
3. Master (the)
4. Dalek Supreme (Supreme Dalek), inc Progenitor
5. Sontarans
6. Ice Warriors (inc benevolent appearences)
7. Davros
8. Silurians (exc Sea Devils)
9. Kovarian, Madame
10. Cyber-leader
Enter a selection => stop
stop

1. Daleks
2. Cybermen
3. Master (the)
4. Dalek Supreme (Supreme Dalek), inc Progenitor
5. Sontarans
6. Ice Warriors (inc benevolent appearences)
7. Davros
8. Silurians (exc Sea Devils)
9. Kovarian, Madame
10. Cyber-leader
Enter a selection => -1
-1

1. Daleks
2. Cybermen
3. Master (the)
4. Dalek Supreme (Supreme Dalek), inc Progenitor
5. Sontarans
6. Ice Warriors (inc benevolent appearences)
7. Davros
8. Silurians (exc Sea Devils)
9. Kovarian, Madame
10. Cyber-leader
Enter a selection => 1
1

You reached the villain Daleks in 1 steps.
Have a nice day.
```