

## Programming Project 05

This assignment is worth 40 Points (4% of the course grade) and must be completed and turned in before 11:59 on Monday, October 16.

This assignment is worth 40 points (4.0% of the course grade) and must be **completed and turned in before 11:59 PM on Monday, October 16, 2023 (2 weeks). After the due date, your score will be deducted by 25% for every 12 hours late or a fraction of it.**

**You are not allowed to use advanced data structures other than lists such as sets, dictionaries and classes. Use of these prohibited concepts earns zero points for the whole project.**

## 1. Assignment Overview (learning objectives)

This assignment will give you more experience on the use of:

1. Files and exceptions
2. Lists and Nested lists
3. Functions

Have you ever wondered how social media sites suggest friends to you? Well, most of the sites do indeed use highly sophisticated algorithms, but for this project you will implement a naive method to recommend the most likely new friend to users of a social network.

## 2. Assignment Deliverable

The deliverable for this assignment is the following file:

`proj05.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **Codio system** before the project deadline.

## 3. Assignment Background

A social network such as Facebook consists of a set of users and connections between the users (i.e., there exists a connection between you and everyone who you have befriended). An interesting problem is to write a computer program that can automatically suggest possible new connections (i.e., friends) for each user. For this project you will implement a system that will, for each user, suggest the most probable user to befriend based upon the intersection of your common friends. In other words, the user that you will suggest to Person A is the person who has the most friends in common with Person A, but who currently is not friends with Person A.

The algorithm begins with: *for* each user go through all the other users (another “for”) and calculate the number of friends they have in common. Then for a given user the friend you will suggest to them is the user in the social network who they are currently not friends with, but have the most friends in common. Intuitively, it makes sense why they might want to be connected as friends. (Note that when you look for your most common friend in this scheme it will be you, i.e. you will have to remember to remove yourself from consideration.)

## 4. Project Description / Specification

1) Two files have been provided for you to run your program with. One is a small test file containing a made-up set of users and connections between them (that file is `small_network_data.txt`). The other is a subset of a real Facebook dataset, which was obtained from: <https://snap.stanford.edu/data/egonets-Facebook.html>

The format of both files is the same:

The first line of the file is an integer representing the number of users in the given network.

The following lines are of the form: `user_u user_v`

where `user_u` and `user_v` are the IDs of two users who are friends.

For example, here is a very small file that has 5 users in the social network:

```
5
0 1
1 2
1 4
2 3
```

The above is a representation of a social network that contains 5 users.

User ID=0 is friends with User IDs = 1

User ID=1 is friends with User IDs = 0, 2, 4

User ID=2 is friends with User IDs = 1, 3

User ID=3 is friends with User IDs = 2

User ID=4 is friends with User IDs = 1

2) We provide a starter file with function definitions. You must implement the following functions:

a) `open_file()` → file pointer (`fp`)

This function does not accept any argument. It prompts the user to enter a file name. The program will try to open the data file. Appropriate error message should be shown if the data file cannot be opened. This function will loop until it receives proper input and successfully opens the file. It returns a file pointer.

b) `read_file(fp)` → List of Lists of Integers

The function will have the file pointer `fp` passed as argument. The first line of the file contains the value for the variable  $n$ , the number of users in the social network. This function will return a list *network* containing  $n$  nested lists. This list of lists will be used to hold the list of friends for each of the  $n$  users. The remaining lines of the file contain the information about each user. For each line of the file, you will want to obtain the two user id values on that line—we denote those user ids as  $u$  and  $v$ . You need to place  $v$  into  $u$ 's list and place  $u$  into  $v$ 's list, since they are mutual friends (hint: use the `append()` method to place each user id into a list). Finally, you will return the *network* list, which is the list of lists that were created.

For example, let's say that the file content is the following:

```
5
0 1
1 2
1 4
2 3
```

Then the return *network* list will be the following:

```
network = [[1], [0, 2, 4], [1, 3], [2], [1]]
```

c) `num_in_common_between_lists(list1, list2)` → integer (`int`)

The function calculates the number of common items between two lists. This function is to return an integer, which is the number of items that the two lists had in common (i.e., the number of items that were found in both of the lists).

(e.g., if `list1 = [1,3,5,6]` and `list2 = [1, 2, 3]`, then the value 2 would be returned since they have two items in common (the values 1 and 3)).

d) `calc_similarity_scores(network)` → List of Lists of Integers

The function takes the social network as the parameter `network` and creates a “similarity matrix” for all the users in the social network (forget the math term matrix, it is just another list of lists). Here the definition of “similarity” is the number of friends that any pair of users have in common. The first thing you will want to do is initialize a list of lists that is  $n \times n$  in size to all zeros that will hold the similarity (i.e., number of common friends) for each pairing of users. In other words, you will create a list of lists that will have  $n$  nested lists, each containing  $n$  zeros. (e.g.,  $n=3$  gives the following list `[ [0,0,0], [0,0,0], [0,0,0] ]`).

You will then use two nested `for` loops to iterate over all pairs of users in the *network*. For each pair of users, you can obtain their list of friends from *network*, and then call the function

`num_in_common_between_lists(user1_friend_lst, user2_friend_lst)`, which will return an integer, `num_in_common`, representing the number of friends those two users have in common. The int `num_in_common` will need to be stored in locations `[user1][user2]` and `[user2][user1]` of the list of lists *similarity matrix*. Return *similarity\_matrix*

Suggestion, to iterate over all pairs of users use the following pseudo-code:

```
for each integer i in range [0,n):
    for each integer j in range [0,n):
        # call num_in_common_between_lists using
        user i's list and user j's list as arguments
```

e) `recommend(user_id, network, similarity_matrix)` → integer (int)

This `user_id` will be used as an index into the *similarity\_matrix* which will give you access to the list of similarity scores for that given user (`user_id`) with all other users. You should then determine the largest value in this list and return its index as the most similar. Of course, you don't want to recommend someone who is already a friend, and it also doesn't make sense to recommend the person as their own friend. It is the most similar because by having the largest value in the list, it means that it is the user who had the most friends in common with `user_id`. (Hint: while developing this function using the small network file, I printed out `similarity_matrix[user_id]` and `network[user_id]` so I could check that the correct recommendation was calculated.). To be consistent with testing, if there is a tie recommend the user with the lowest `user_id`.

f) `main()`

In the main program, you will need to first call `open_file()` to get the file pointer, followed by calling `read_file(fp)` to get the list for the network, and then `calc_similarity_scores(network)` to get the *similarity matrix*. After obtaining the *similarity\_matrix*, your program needs to repeatedly do the following:

1. ask the user for a `user_id`,
2. get friend recommendation,
3. display the suggested friend for the user that had their id as `user_id`
4. you should prompt the user if they want to see the suggestion for another user input or if they want to exit.
5. repeat step 1 through 4 until the user wants to exit the program. To exit the program you should accept all forms of the string 'no' (i.e., 'NO', 'No', 'nO', and 'no').

g) Note that when prompting the user for the `user_id` you should validate that the input is in the correct range, which is 0 to  $n$ . Also, you must use a `try-except` for the cases when the user inputs a non-integer value (e.g., for the case if they try to input the string “hello” instead of an int). that should be an integer in the range of  $[0,n)$  (i.e., from 0 to  $n-1$ , inclusive).

## Scoring Rubric

Computer Project #05

Scoring Summary

### General Requirements

\_\_\_\_\_ (6 pts) Coding Standard 1-9  
(descriptive comments, function header, etc...)

### Implementation:

\_\_0\_\_ (3 pts) Pass test1:

\_\_0\_\_ (3 pts) Pass test2

\_\_0\_\_ (3 pts) Pass test3

\_\_0\_\_ (9 pts) Pass hidden tests (3 pts each).

\_\_0\_\_ (8 pts) All functions work as specified.  
          (2 pt) read\_file  
          (2 pt) num\_in\_common\_between\_lists  
          (2 pt) calc\_similarity\_scores  
          (2 pt) recommend

\_\_0\_\_ (8 pts) Pass All hidden functions tests.  
          (2 pt) read\_file  
          (2 pt) num\_in\_common\_between\_lists  
          (2 pt) calc\_similarity\_scores  
          (2 pt) recommend

Note: hard coding an answer earns zero points for the whole project  
-10 for not using Main, correct loop control, error checking

## Optional Testing

This test suite has a test program for each of the three functions and two for the entire program. They are handled slightly differently.

Test individual functions using `function_testX.py` and the entire program using `run_file.py`. You need to have the files `test1.txt`, `test2.txt`, and `test3.txt` from the project directory. Make sure that you have the following lines at the top of your program (only for testing, do not copy them to Codio):

```
import sys
def input( prompt=None ):
    if prompt != None:
        print( prompt, end="" )
    aaa_str = sys.stdin.readline()
    aaa_str = aaa_str.rstrip( "\n" )
    print( aaa_str )
    return aaa_str
```

## Sample Outputs

### **read\_file Function Test**

(reads the file `small_network_data.txt`)

network:

```
[[1, 2, 3], [0, 4, 6, 7, 9], [0, 3, 6, 8, 9], [0, 2, 8, 9], [1, 6, 7, 8], [9], [1, 2, 4, 8], [1, 4, 8], [2, 3, 4, 6, 7], [1, 2, 3, 5]]
```

-----  
network printed nicely:

```
0 : [1, 2, 3]
1 : [0, 4, 6, 7, 9]
2 : [0, 3, 6, 8, 9]
3 : [0, 2, 8, 9]
4 : [1, 6, 7, 8]
5 : [9]
6 : [1, 2, 4, 8]
7 : [1, 4, 8]
8 : [2, 3, 4, 6, 7]
9 : [1, 2, 3, 5]
```

### **num\_in\_common\_between\_lists Function Test**

(network[1] and network[2] from read\_file test)

should be 3: 3

should be 1: 1

### **calc\_similarity\_scores Function Test**

(uses network from read\_file test as the argument)

similarity matrix

```
[[3, 0, 1, 1, 1, 0, 2, 1, 2, 3], [0, 5, 3, 2, 2, 1, 1, 1, 3, 0], [1, 3, 5, 3, 2, 1, 1, 1, 2, 1], [1, 2, 3, 4, 1, 1, 2, 1, 1, 1], [1, 2, 2, 1, 4, 0, 2, 2, 2, 1], [0, 1, 1, 1, 0, 1, 0, 0, 0, 0], [2, 1, 1, 2, 2, 0, 4, 3, 2, 2], [1, 1, 1, 1, 2, 0, 3, 3, 1, 1], [2, 3, 2, 1, 2, 0, 2, 1, 5, 2], [3, 0, 1, 1, 1, 0, 2, 1, 2, 4]]
```

-----  
similarity matrix printed nicely:

```
0 : [3, 0, 1, 1, 1, 0, 2, 1, 2, 3]
1 : [0, 5, 3, 2, 2, 1, 1, 1, 3, 0]
2 : [1, 3, 5, 3, 2, 1, 1, 1, 2, 1]
3 : [1, 2, 3, 4, 1, 1, 2, 1, 1, 1]
4 : [1, 2, 2, 1, 4, 0, 2, 2, 2, 1]
5 : [0, 1, 1, 1, 0, 1, 0, 0, 0, 0]
6 : [2, 1, 1, 2, 2, 0, 4, 3, 2, 2]
7 : [1, 1, 1, 1, 2, 0, 3, 3, 1, 1]
8 : [2, 3, 2, 1, 2, 0, 2, 1, 5, 2]
9 : [3, 0, 1, 1, 1, 0, 2, 1, 2, 4]
```

### **recommend Function Test**

(uses similarity\_matrix from calc\_similarity\_scores Test)

person,friend: 3 1

person,friend: 0 9

## Test 1

Facebook friend recommendation.

Enter a filename: small\_network\_data.txt

Enter an integer in the range 0 to 9: 0

The suggested friend for 0 is 9

Do you want to continue (yes/no)? yes

Enter an integer in the range 0 to 9: 3

The suggested friend for 3 is 1

Do you want to continue (yes/no)? yes

Enter an integer in the range 0 to 9: 8

The suggested friend for 8 is 1

Do you want to continue (yes/no)? No

## Test 2

Facebook friend recommendation.

Enter a filename: xxxxxx

Error in filename.

Enter a filename: small\_network\_data.txt

Enter an integer in the range 0 to 9: abc

Error: input must be an int between 0 and 9

Enter an integer in the range 0 to 9: -1

Error: input must be an int between 0 and 9

Enter an integer in the range 0 to 9: 15

Error: input must be an int between 0 and 9

Enter an integer in the range 0 to 9: 10

Error: input must be an int between 0 and 9

Enter an integer in the range 0 to 9: 0

The suggested friend for 0 is 9

Do you want to continue (yes/no)? nO

### Test 3

**Note that this test takes time because of the size of the social network.**

Facebook friend recommendation.

Enter a filename: facebook\_1000\_data.txt

Enter an integer in the range 0 to 999: 88

The suggested friend for 88 is 213

Do you want to continue (yes/no)? yes

Enter an integer in the range 0 to 999: 20

The suggested friend for 20 is 116

Do you want to continue (yes/no)? yes

Enter an integer in the range 0 to 999: 5

The suggested friend for 5 is 67

Do you want to continue (yes/no)? yes

Enter an integer in the range 0 to 999: 500

The suggested friend for 500 is 559

Do you want to continue (yes/no)? no

## **Educational Research**

**When you have completed the project in Codio please answer the following questions in the D2L survey.**

For each of the following statements, please respond with how much they apply to your experience completing the programming project, on the following scale:

**1** = Strongly disagree / Not true of me at all

**2**

**3**

**4** = Neither agree nor disagree / Somewhat true of me

**5**

**6**

**7** = Strongly agree / Extremely true of me

*\*\*\*Please note that your responses to these questions will not affect your project grade, so please answer as honestly as possible.\*\*\**

**Q1: Upon completing the project, I felt proud/accomplished**

**Q2: While working on the project, I often felt frustrated/annoyed**

**Q3: While working on the project, I felt inadequate/stupid**

**Q4: Considering the difficulty of this course, the teaching staff, and my skills, I think I will do well in this course.**

**Q5: I ran the optional test cases in PyCharm (choose 7=Yes, 1=No)**

**Q6: I used Copilot for this project (choose 7=Yes, 1=No)**