# COMP1811 – Python Project Report

| **Name** | **Iustin-Andrei Moisa Tudor** | **Student ID** | **001228763** |
| --- | --- | --- | --- |

# Brief statement of features you have completed

| 1.1 Circle the parts of the coursework you have **fully completed and are fully working**. Please be accurate. | **Features F1:** i ii  **F2:** i ii  **F3:** i ii  iii iv v |  |
| --- | --- | --- |
| 1.2 Circle the parts of the coursework you have **partly completed or are partly working.** | **Features F1:** i ii  **F2:** i ii  **F3:** i ii  iii iv v |  |
| Briefly explain your answer if you circled any parts in 1.2 | | |

# Concise List of Bugs and Weaknesses

*A concise list of bugs and/or weaknesses in your work (if you don't think there are any, then say so). Bugs that are declared in this list will lose you fewer marks than ones that you don't declare! (****100-200 word****, but word count depends heavily on the number of bugs and weaknesses identified.)*

## Bugs

*List each bug plus a brief description*

1. The first bug I encountered with my code is that I have to type the name of the individual in the social media network twice in the console instead of only once
2. The 2nd bug in my code is that when I try to display the contents of Yasmine’s text file containing names not numbers, Not all the contents of the social network are showing up, only the first 3 lines. But on my own social network file, all members are present.

## Weaknesses

*List each weakness plus a brief description*

# My first weakness of my code is having to input the individual’s name twice instead of only once in the python console.

# Description of the features implemented

*Describe your implementation of the required features and how well do they work. Provide some exposition of the design decisions made and indicate how the features developed were integrated.*

This code defines three classes: **socialMediaNetwork**, **SocialNetworkAnalytics**, and **CommonFriends**.

The **socialMediaNetwork** class allows the user to open a file, display its contents, and recommend a friend to an individual in the social network represented by the file.

The **SocialNetworkAnalytics** class contains a method to show friends of a specific individual and another method to find indirect connections for all members of the social network.

The **CommonFriends** class calculates the number of common friends for each member of the social network.

The first feature I was asked to implement was **Feature 1** which opens the social media file where all names of the social network members/individuals are get then simulate the social network and display their names in a pretty print method. All of this code works with no problems on my code

The 2nd feature implemented is **feature 2** which asks me to get a common friend count for all individuals of the social media network which is then pretty printed in the output, then recommend a friend to the individual the user inputted. This feature works well and can work oon various different social network files as expected.

The 3rd feature I was aked to implement is **feature 3** which contains social media analytics and statistics. It’s supposed to show the amount of friends a member has and show the members with the least friends and don’t have any. And this feature should also show the indirect relationships of the members in the network compared to the individual’s name the user entered.

# Classes and OOP Features

*List all the classes used in your program and include the attributes and behaviours for each. You may use a class diagram to illustrate these classes. Your narrative for section 3.2 should describe the design decisions you made and the OOP techniques used. List the classes you developed and provide an exposition on the choice of classes, class design and OOP features implemented. (****200-400 words****).*

## Classes Used

My code contains 3 classes: socialMediaNetwork which has function and methods to open and display the social network and recommend friends.

The 2nd class is called SocialNetworkAnalytics which has functions and methods to show the individual’s friends and find indirect friends between other members in the social media network file.

And the 3rd class is called CommonFriends which gets the common friend names for each user in the social network

## Brief Explanation of Class Design and OOP Features Used

Below I listed all the classes and the functions they use as well as the OOP features used into my code.

1. socialMediaNetwork: The primary social media platform is represented by this class. It possesses the following qualities and practises:

* social nw: A dictionary to keep track of user connections on social networks.
* \_\_init\_\_: Creates the social nw dictionary from scratch.
* showFiles: This command requests the file name from the user, then reads the social network connections from that file and stores them in the social\_nw dictionary.
* display\_social\_nw: Shows the links to social networks that are stored in social nw.

Using the CommonFriends class, recommend friend suggests a friend to a particular user depending on how many friends they share with other users.

1. SocialNetworkAnalytics: This class conducts social network analytics. It possesses the following methods and functions:

* Social\_nw: A dictionary to keep track of user connections on social networks.
* \_\_init\_\_: Creates the social nw dictionary from scratch.
* showFriends: Displays the users' friends.
* find indirect connections: Discovers connections between users in an indirect way.

1. CommonFriends: This class counts the number of friends that two users have in common. It possesses the following qualities and practises:

social nw: A dictionary to keep track of user connections on social networks.

\_\_init\_\_: Creates the social nw dictionary from scratch.

The number of common friends each user has with other users is returned in a dictionary by the get common friends command.

Several Object-Oriented Programming (OOP) concepts are used in the code.

**Inheritance**: The social nw dictionary comes from the socialMediaNetwork class and is passed on to CommonFriends and SocialNetworkAnalytics.

**Encapsulation**: By being defined as an instance variable inside of each class's function Object() { [native code] }, the social nw attribute is shielded from external alteration.

**Polymorphism**: The showFiles method can read data from any file that the user provides, making it polymorphic.

**Abstraction**: The user just needs to be aware of how to utilise the methods given by the socialMediaNetwork class; the implementation specifics of the analytics and recommendation functions are abstracted away from the user.

# Code for the Classes Created

*Add the* ***code for each of the classes you have implemented yourself*** *here. If you have contributed to parts of classes, please highlight those parts in a different colour. Copy and paste relevant code - actual code please, no screenshots! Make it easy for the tutor to read. Add explanation if necessary – though your in-code comments should be clear enough. You will lose marks if screenshots are provided instead of code.*

*(DO NOT provide a listing of the entire code. You will be marked down if a full code listing is provided.)*

## Class socialmedianetwork:

class socialMediaNetwork :  
 def \_\_init\_\_(self):  
 self.social\_nw = {}  
  
#The following function opens the file the user chose  
 def showFiles(self):  
 while True:  
 #Program asks the user to input file name in terminal  
 file\_name = input("Please enter the file name you with to open: ")  
 try:  
 with open(file\_name, 'r') as file:  
 for line in file:  
 names = line.strip().split()  
 # The line.strip() will remove any white spaces in between the values inside the file and then split the values into more.  
 if names:  
 self.social\_nw[names[0]] = names[1:]  
 return self.display\_social\_nw()  
 except FileNotFoundError:  
 #If name of file isn't found, a FileNotFoundError is shown  
 print("The file specified can't be found.")  
 choice = input("Would you like to search again? Y/N")  
 #The user will be prompted to try again  
 #If they press N, the program closes  
 if choice == 'n':  
 exit()  
  
 # Displays the social network  
 def display\_social\_nw(self):  
 for name, friends in self.social\_nw.items():  
 print(name + " -> " + ", ".join(friends))  
  
 def recommend\_friend(self, individualName):  
 # Create an instance of the CommonFriends class and pass the social network dictionary  
 common\_friends = CommonFriends(self.social\_nw)  
 # Get a dictionary of common friends count for each member and check if the member is in the common friends count dictionary using an if statement  
 common\_friends\_count = common\_friends.get\_common\_friends()  
 if individualName not in common\_friends\_count:  
 # Return message if member not found  
 return "Recommended friend for {} is none".format(individualName)  
 # Get a list of friend counts for the member  
 friend\_counts = common\_friends\_count[individualName]  
 # Get a list of friend names  
 friend\_names = list(common\_friends\_count.keys())  
 # Find the maximum count of common friends  
 max\_count = max(friend\_counts)  
 # Return message if no common friends found  
 if max\_count == 0:  
 return "The recommended friend for {} is none".format(individualName)  
 # Get the index of the member with the maximum count of common friends  
 max\_index = friend\_counts.index(max\_count)  
 # Check if the member with the maximum count of common friends is the input member  
 if max\_index == friend\_names.index(individualName):  
 # Set the count of common friends to 0  
 friend\_counts[max\_index] = 0  
 # Find the next maximum count of common friends  
 max\_count = max(friend\_counts)  
 max\_index = friend\_counts.index(max\_count)  
 # Get the name of the friend with the maximum count of common friends  
 friend\_name = friend\_names[max\_index]  
 # Check if the recommended friend is already a friend of the input member  
 if friend\_name in self.social\_nw[individualName]:  
 # Return message if the recommended friend is already a friend of the input member  
 return "The recommended friend for {} is none".format(individualName)  
 return "The recommended friend for {} is {}".format(individualName, friend\_name)

## Class socialnetworkanalytics

1. lass SocialNetworkAnalytics:  
    def \_\_init\_\_(self, social\_nw):  
    #Initialize the class with the social network data  
    self.social\_nw = social\_nw  
    def showFriends(self, userName):  
    if individualName in self.social\_nw:  
    print("Friends of {}: {}".format(individualName, ", ".join(self.social\_nw[individualName])))  
    else:  
    print("Error: Member not found.")  
     
    def find\_indirect\_connections(self): #This function is to find the indirect connections for all sn members  
    indirect\_friends = {}  
    for member in self.social\_nw:  
    indirect\_friends[member] = set()  
    for friend in self.social\_nw[member]:  
    # Add friends of friend to the indirect friends set  
    indirect\_friends[member].update(self.social\_nw.get(friend, []))  
    indirect\_friends[member].discard(member)  
    return indirect\_friends

## Class commonfriends

1. class CommonFriends:  
    def \_\_init\_\_(self, social\_nw):  
    self.social\_nw = social\_nw  
     
    def get\_common\_friends(self):  
    #The function is supposed to return a dictionary with keys and values representing the number of common friends with other members.  
    common\_friends = {}  
    for key1, value1 in self.social\_nw.items():  
    common\_friends[key1] = []  
    for key2, value2 in self.social\_nw.items():  
    common = set(value1) & set(value2) # Finds common friends between the 2 named members  
    common\_friends[key1].append(  
    len(common)) # Appends the length of mutual friends to the list of common friends of the first member  
    return common\_friends

# Testing

*Describe the process you took to test your code and to make sure the program functions as required. Provide the detailed test plan used.*

*Aftereach feature I implemented, I tested each method and function in accordance with the requirements to check if the code works and what doesn’t work so well. I developed a test plan to help me classify the main problems I encountered while creating the code which is listed below:*

|  |  |  |  |
| --- | --- | --- | --- |
| ***TASK NAME*** | ***TASK DESCRIPTION*** | ***Working? If no, why?*** | ***Changes made*** |
| *F1: Opening the social media network file* | *The user enters a file name and the file opens and displays the social network data* | *No, because the nw\_data1.txt file wasn’t in the same directory as the main .py file* | *I moved the .txt file to the correct place in the directory and it started working* |
| *F1: Output should be shown in the pycharm console pretty printed* | *The output from the .txt file should show in a neat manner into the pycharm console* | *No, the output from the .txt file is in 1 line containing many parentheses and commas* | *I imported the pprint package into the code and pretty printed the results using that module.* |
| *F2: Recommending a friend to the individual the user chose* | *The program should use the classes and methods to define recommended friends for the individual the user inputted manually from the social media network file* | *Yes* | *None* |
| *F3: Showing how many friends the individual chosen by the user has* | *This feature should be able to show the amount of friends that the individual the user chose has as well as list their name and pretty print them.* | *Yes* | *None* |
| *F3: Finding indirect relationships between friends* | *The program should be able to determine the indirect relationship between the individual the user chose and the other members in the social media file and output the names of the indirect friends* | *Yes* | *None* |

# Annotated Screenshots Demonstrating Implementation

*Provide screenshots that demonstrate the features implemented. Annotate each screenshot and if necessary, provide a brief description for* ***each*** *(****up to 100 words****) to explain the code in action. Make sure the screenshots make clear what you have implemented and achieved.*

## Feature F1

## Sub-feature i- screenshots …

Text

Description automatically generatedText

Description automatically generated

In the 1st screenshot it can be seen that the program asks the user to input a file name into the prompt, if a non-existent name is put in, it will give an error through the FileNotFoundError in python and ask the user if they want to enter a name again, in case they press N, the program will close, if yes, they will be prompted to do so again and if they put a valid file name the content will be shown

## Sub-feature ii- screenshots …

Text

Description automatically generated

This part of the code is feature 2 which then shows the content of the file and pretty prints the results through the pprint module in python which was imported at the start of the code.

## Feature F2

## Sub-feature i- screenshots …

Text

Description automatically generated

The code will get a common friend count for all members of the social network in the file by working out 2 members of the social network and do the same for the rest, then pretty print the results

## Sub-feature ii- screenshots …Text Description automatically generated

Here the code asks the user to enter an individual’s name int eh network to recommend a friend, in this case I used Hyunjin which has no recommended friends

## Feature F3

## Sub-feature i- screenshots …

Text

Description automatically generated

In this part of the code, the individual’s name should be put in the prompt again then show its friends

## Sub-feature ii- screenshots …

## Sub-feature iii- screenshots …

## Sub-feature iv- screenshots …

Text

Description automatically generated

In this part of the code, the program will find the indirect friends of Hyunjin and display them into the PyCharm console.

## Sub-feature v- screenshots …

## Sub-feature vi- screenshots …

# Evaluation

*Give a reflective, critical self-evaluation of your experience developing the project and discuss what you would do if you had more time to work on the project. Answer the following questions for the reflection and write* ***350-400 words overall****. Please include an actual word count for this section.*

## Evaluate how well your design and implementation meet the requirements

My code works well in most parts apart from when I want it to read a specific file provided by Yasmine for the demo day

## Evaluate you own performance

It took me overall 2 months to finish the code during the other coursework going on and I started in November and finished in January

## What went well?

Almost all the features worked perfectly as requested after the implementation I’ve done and the results are as expected, accurate

## What went less well?

Some members in the demo.txt file couldn’t be read and the reason is still unknown because there isn’t any error in my code.

## What was learnt?

I learned how to develop a social media network which recommends friends to individuals already in the database and how to implement different object oriented programming techniques and classes to make the different parts of the code work together

## How would a similar task be completed differently?

-

## How could the module be improved?

In my opinion, one way the module could be improved is for the teachers to communicate more with students and explain the code in the labs in depth. Overall excellent teaching experience.

## Self-assessment

*Please assess yourself objectively for each section shown below and then enter the total mark you expect to get. Marks for each assessment criteria are indicated between parentheses.*

## Code development (70)

Features Implemented [30]

Feature 1 (up to 6)

Sub-features have not been implemented – 0

Attempted, not complete or very buggy – 1 or 2

Implemented and functioning without errors but not integrated – 3

Implemented and fully integrated but buggy – 4

Implemented, fully integrated and functioning without errors – 5 or 6

Feature 2 (up to 12)

Sub-features have not been implemented – 0

Attempted, not complete or very buggy – 1 or 3

Implemented and functioning without errors but not integrated – 4 to 6

Implemented and fully integrated but buggy – 7 to 9

Implemented, fully integrated and functioning without errors – 10 to 12

Feature 3 (up to 12)

Sub-features have not been implemented – 0

Attempted, not complete or very buggy – 1 to 3

Implemented and functioning without errors but not integrated – 4 to 6

Implemented and fully integrated but buggy – 7 to 9

Implemented, fully integrated and functioning without errors – 10 to 12

**For this criterion I think I got: 18 out of 30**

Use of OOP techniques [25]

Abstraction (up to 9)

No classes have been created – 0

Classes have been created superficially and not instantiated or used – 1 or 2

Classes have been created but only some have been instantiated and used – 3 or 4

Useful classes and objects have been created and used correctly – 5 to 7

The use of classes and objects exceeds the specification – 8 or 9

Encapsulation (up to 9)

No encapsulation has been used – 0

Class variables and methods have been encapsulated superficially – 1 to 3

Class variables and methods have been encapsulated correctly – 4 to 6

The use of encapsulation exceeds the specification – 7 to 9

Inheritance (up to 7)

No inheritance has been used – 0

Classes have been inherited superficially – 1 or 2

Classes have been inherited correctly – 3 to 5

The use of inheritance exceeds the specification – 6 or 7

Bonus marks will be awarded for the appropriate use of polymorphism (bonus marks up to 5)

**For this criterion I think I got: 16 out of 25**

Quality of Code [15]

Code Duplication (up to 8)

Code contains too many unnecessary code repetition – 0

Regular occurrences of duplicate code – 1 to 3

Occasional duplicate code – 4 to 5

Very little duplicate code – 6 to 7

No duplicate code – 8

PEP8 Conventions and naming of variables, methods and classes (up to 4)

PEP8 and naming convention has not been used – 0

PEP8 and naming convention has been used occasionally – 1

PEP8 and naming convention has been used, but not regularly – 2

PEP8 and naming convention has been used regularly – 3

PEP8 convention used professionally and all items have been named correctly – 4

In-code Comments (up to 3)

No in-code comments – 0

Code contains occasional in-code comments – 1

Code contains useful and regular in-code comments – 2

Thoroughly commented, good use of docstrings, and header comments describing.py files – 3

**For this criterion I think I got: 12 out of 15**

## Documentation (20)

Design (up to 10) clear exposition about the design and decisions for OOP use

The documentation cannot be understood on first reading or mostly incomplete – 0

The documentation is readable, but a section(s) are missing – 1 to 3

The documentation is complete – 4 to 6

The documentation is complete and of a high standard – 7 to 10

Testing (5)

Testing has not been demonstrated in the documentation – 0

Little white box testing has been documented – 1 or 2

White box testing has been documented for all the coursework – 3 or 4

White box testing has been documented for the whole system – 5

Evaluation (5)

No evaluation was shown in the documentation – 0

The evaluation shows a lack of thought – 1 or 2

The evaluation shows thought – 3 or 4

The evaluation shows clear introspection, demonstrates increased awareness – 5

**For this criterion I think I got: 11 out of 20**

## Acceptance Tests - Demonstrations (10)

Final Demo (up to 10)

Not attended or no work demonstrated – 0

Work demonstrated was not up to the standard expected – 1 to 3

Work demonstrated was up to the standard expected – 4 to 7

Work demonstrated exceeded the standard expected – 8 to 10

**For this criterion I think I got: 7 out of 10**

**I think my overall mark would be: 64 out of 100**

# Appendix A: Code Listing

*Provide a complete listing of all the \*.py files in your PyCharm project. Make sure your code is well commented and applies professional Python convention (refer to* [*PEP 8*](https://www.python.org/dev/peps/pep-0008/) *for details). The code listed here must match that uploaded to Moodle. Please copy and paste the actual code – no screenshots please! You will lose marks if screenshots are provided instead of code.*

*The .py file in the zip folder is called: mainCW.py which contains the following code:*

import os  
import pprint  
  
# 1st class will represent the main social media program  
class socialMediaNetwork :  
 def \_\_init\_\_(self):  
 self.social\_nw = {}  
  
#The following function opens the file the user chose  
 def showFiles(self):  
 while True:  
 #Program asks the user to input file name in terminal  
 file\_name = input("Please enter the file name you with to open: ")  
 try:  
 with open(file\_name, 'r') as file:  
 for line in file:  
 names = line.strip().split()  
 # The line.strip() will remove any white spaces in between the values inside the file and then split the values into more.  
 if names:  
 self.social\_nw[names[0]] = names[1:]  
 return self.display\_social\_nw()  
 except FileNotFoundError:  
 #If name of file isn't found, a FileNotFoundError is shown  
 print("The file specified can't be found.")  
 choice = input("Would you like to search again? Y/N")  
 #The user will be prompted to try again  
 #If they press N, the program closes  
 if choice == 'n':  
 exit()  
  
 # Displays the social network  
 def display\_social\_nw(self):  
 for name, friends in self.social\_nw.items():  
 print(name + " -> " + ", ".join(friends))  
  
 def recommend\_friend(self, individualName):  
 # Create an instance of the CommonFriends class and pass the social network dictionary  
 common\_friends = CommonFriends(self.social\_nw)  
 # Get a dictionary of common friends count for each member and check if the member is in the common friends count dictionary using an if statement  
 common\_friends\_count = common\_friends.get\_common\_friends()  
 if individualName not in common\_friends\_count:  
 # Return message if member not found  
 return "Recommended friend for {} is none".format(individualName)  
 # Get a list of friend counts for the member  
 friend\_counts = common\_friends\_count[individualName]  
 # Get a list of friend names  
 friend\_names = list(common\_friends\_count.keys())  
 # Find the maximum count of common friends  
 max\_count = max(friend\_counts)  
 # Return message if no common friends found  
 if max\_count == 0:  
 return "The recommended friend for {} is none".format(individualName)  
 # Get the index of the member with the maximum count of common friends  
 max\_index = friend\_counts.index(max\_count)  
 # Check if the member with the maximum count of common friends is the input member  
 if max\_index == friend\_names.index(individualName):  
 # Set the count of common friends to 0  
 friend\_counts[max\_index] = 0  
 # Find the next maximum count of common friends  
 max\_count = max(friend\_counts)  
 max\_index = friend\_counts.index(max\_count)  
 # Get the name of the friend with the maximum count of common friends  
 friend\_name = friend\_names[max\_index]  
 # Check if the recommended friend is already a friend of the input member  
 if friend\_name in self.social\_nw[individualName]:  
 # Return message if the recommended friend is already a friend of the input member  
 return "The recommended friend for {} is none".format(individualName)  
 return "The recommended friend for {} is {}".format(individualName, friend\_name)  
  
class SocialNetworkAnalytics:  
 def \_\_init\_\_(self, social\_nw):  
 #Initialize the class with the social network data  
 self.social\_nw = social\_nw  
 def showFriends(self, userName):  
 if individualName in self.social\_nw:  
 print("Friends of {}: {}".format(individualName, ", ".join(self.social\_nw[individualName])))  
 else:  
 print("Error: Member not found.")  
  
 def find\_indirect\_connections(self): #This function is to find the indirect connections for all sn members  
 indirect\_friends = {}  
 for member in self.social\_nw:  
 indirect\_friends[member] = set()  
 for friend in self.social\_nw[member]:  
 # Add friends of friend to the indirect friends set  
 indirect\_friends[member].update(self.social\_nw.get(friend, []))  
 indirect\_friends[member].discard(member)  
 return indirect\_friends  
  
  
class CommonFriends:  
 def \_\_init\_\_(self, social\_nw):  
 self.social\_nw = social\_nw  
  
 def get\_common\_friends(self):  
 #The function is supposed to return a dictionary with keys and values representing the number of common friends with other members.  
 common\_friends = {}  
 for key1, value1 in self.social\_nw.items():  
 common\_friends[key1] = []  
 for key2, value2 in self.social\_nw.items():  
 common = set(value1) & set(value2) # Finds common friends between the 2 named members  
 common\_friends[key1].append(  
 len(common)) # Appends the length of mutual friends to the list of common friends of the first member  
 return common\_friends  
  
  
# Creating an object of the socialMediaNEtwork class and calling it  
sn = socialMediaNetwork()  
sn.showFiles()  
  
common\_friends = CommonFriends(sn.social\_nw) # Showing the common friend count for all individuals in the network file  
print("This is the common friend counter for each friend in the social network")  
pprint.pprint(common\_friends.get\_common\_friends())  
  
individualName = input("Please enter an individual's name")  
print(sn.recommend\_friend(individualName)) #This prints the recommended friend's name  
  
sna = SocialNetworkAnalytics(sn.social\_nw)  
# Asking for a member name from the user  
individualName = input("Please enter an individual's name")  
sna.showFriends(individualName)  
  
indirect\_friends = sna.find\_indirect\_connections()  
print("Indirect friends:")  
pprint.pprint(indirect\_friends)