Rylee Texter

Problem 1

```
# reads a file and outputs a new file with numbered lines along with original data
def line_number(inputFile, outputFile):
    lineNum = 0
    lines = ''
    try:
        pyFile = open(inputFile, "r")
        txtFile = open(outputFile, "w")
        lines = pyFile.readlines()


        #loops through lines and prints lineNum counter (line numbers) and original data
        #on the line
        for line in lines:
            lineNum += 1
            line_str = "{}. {}".format(lineNum, line)
            print(line_str, file = txtFile)

        pyFile.close()
        txtFile.close()

    except Exception:
        print("File does not exist")

#reads a file and outputs a list of tuples containing function information
def parse_function(fileName):

    file = open(fileName, "r")
    lines = file.readlines()
    line_count = 0
    lineNums = []
    names = []
    arguments = []
    func_content = []
    func_lines = []

    #finds which lines are functions, saves their indexes, names, info
    for line in lines:
        line_count+=1
```

```
    if ("def" in line):
        ##found function
        lineNums.append(line_count)
        char_count = 3

        #gets index of the end of the function name
        while(True):
            char_count  += 1
            if line[char_count] == "(":
                break;

        names.append(line[4:char_count])
        argument_index = char_count + 1

        #gets arguments from line and appends them to a list arguments
        while(True):
            char_count+=1

            if line[char_count] == ")":
                arguments.append(line[argument_index: char_count])
                break


#loops through indexes in lineNums for the found functions
#finds the amount of lines each function has and appends the index at the
#end of the function to a list, func_lines
for num in lineNums:
    counter = num

    #loops through the lines until finding the next line outside of the function
    #and appends the index of the nextline to func_lines
    while(True):
        if len(lines) <= counter:
            func_lines.append(counter)
            break;
        #removes empty lines
        if((lines[counter][0] == " ") and (lines[counter].isspace() == False)):
            counter += 1

        else:
            func_lines.append(counter)
            break;
```

```
    #loops through func lines and uses i to get the range
    for i in range(len(func_lines)):
        func_string = ""
        rang = range(lineNums[i]-1, func_lines[i])

        #loops through the function lines and appends function content and
        #formatting
        for j in rang:
            func_string += lines[j].strip()
            if j < func_lines[i]-1:
                func_string += "/n"
        func_content.append(func_string)

    list_of_tuples = []

    #loops through each index and creates a tuple with function info
    for i in range(len(lineNums)):
        tup = (lineNums[i], names[i], arguments[i], func_content[i])
        list_of_tuples.append(tup)

    #displays information
    for tup in list_of_tuples:
        print(tup)


def main():
    line_number("test.py", "output.txt")
    output_file = open("output.txt", "r")
    lines = output_file.readlines()

    for line in lines:
        print(line)

    output_file.close()

    print("Parsed: ")
    parse_function("test.py")

main()
```

Output screenshots:

```
Browse tabs  1/A
In [83]: runfile('C:/Users/rylee/Desktop/p1_Texter_Rylee.py', wdir='C:/Users/rylee/Desktop')
1. # -*- coding: utf-8 -*-


2. """


3. Created on Thu Feb  8 20:17:14 2024


4.


5. @author: rylee


6. """


7.


8. def adding(x, y):


9.     total = x+y


10.     return total


11.
```

Console 1/A

```
11.


12. def sayHello(name):


13.     return ("Hello "+name)


14.


15. def subracting(x, y):


16.     total = x-y


17.     return total


Parsed:
(8, 'adding', 'x, y', 'def adding(x, y):/ntotal = x+y/nreturn total')
(12, 'sayHello', 'name', 'def sayHello(name):/nreturn ("Hello "+name)')
(15, 'subracting', 'x, y', 'def subracting(x, y):/ntotal = x-y/nreturn total')

In [84]:
```

Problem 2:

```
"""
Created on Fri Feb  2 16:29:47 2024

@author: rylee
"""

#part a: returns tuples with distinct integers, where a^2 + b^2 = c^2 + d^2
#and 1<=a,b,c,d<=10

groups = [(a,b,c,d) for a in range(1,11) for b in range (1,11) for c in range (1,11) for
        d in range(1,11) if((a**2+b**2 == c**2 + d**2) and (((a!= b and a!=c) and (a!=d and b!=
c))and (b!= d and c!= d)))]

print("part a: ")
print(groups)
print()


#part b: returns a list of tuples containing the lowercase string of
#stringList along with lengths that are less than 5 characters
stringList = ['one', 'SEvEN','Three', 'twO', 'ten' ]
lengths = [(string.lower(), len(string)) for string in stringList if len(string) < 5]

print("part b: ")
print(lengths)
print()


#part c: returns a list of formatted names
names = ['Christopher Ashton Kutcher', 'Elizabeth Stamatina Fey']

names_formatted = ["".join([name.split()[0], name.split()[1][0] + '.', name.split()[2]])
        for name in names]

print("part c")
print(names_formatted)
print()

#part d: returns a list of tuples of anagrams between the two lists
lst1 = ["Spam", "Trams", "Elbows", "Tops", "Astral"]
lst2 = ["Bowels", "Sample", "Altars", "Stop", "Course", "Smart"]
```

```python
anagrams = [(x,y) for x in lst1 for y in lst2 if sorted(x.lower()) == sorted(y.lower())]
print("part d")
print(anagrams)
print()

#part e: maps each string to its length
s = ['one', 'two', 'three']

string_lengths = {string: len(string) for string in s}
print("part e: ")
print(string_lengths)
print()

#returns disctionary with index of vowel as key and vowel as value
text = "Hello world"

vowelIndexes = {i:text[i] for i in range(len(text)) if text[i] in ['a','e','i','o','u']}
print("part f")
print(vowelIndexes)
print()
```

Output Screenshots

```
In [85]: runfile('C:/Users/rylee/Desktop/p2_Texter_Rylee.py', wdir='C:/Users/rylee/Desktop')
part a:
[(1, 8, 4, 7), (1, 8, 7, 4), (2, 9, 6, 7), (2, 9, 7, 6), (4, 7, 1, 8), (4, 7, 8, 1), (6, 7, 2, 9), (6, 7, 9, 2), (7, 4, 1, 8), (7, 4, 8, 1), (7, 6, 2, 9),
(7, 6, 9, 2), (8, 1, 4, 7), (8, 1, 7, 4), (9, 2, 6, 7), (9, 2, 7, 6)]

part b:
[('one', 3), ('two', 3), ('ten', 3)]

part c
['ChristopherA.Kutcher', 'ElizabethS.Fey']

part d
[('Trams', 'Smart'), ('Elbows', 'Bowels'), ('Tops', 'Stop'), ('Astral', 'Altars')]

part e:
{'one': 3, 'two': 3, 'three': 5}

part f
{1: 'e', 4: 'o', 7: 'o'}


In [86]:
```

Problem 3

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Feb  5 17:15:24 2024

@author: rylee
"""

import csv


##adds a user to a social network dictionary username: (fullname, [friends])
def add_user(sn, username, fullname):
    if username in sn:
        print("Username Already Exists")
        return False
    sn[username] = (fullname, [])
    return True

#adds connections between users
def add_friend(sn, user1, user2):
    try:
        if user1 in sn and user2 in sn:
            sn[user1][1].append(user2)
            sn[user2][1].append(user1)
            return True
        else: return False
    except Exception:
        print("User does not exist")
        return(False)

##returns a list of friends which come from the user's friends - friend's
def get_friends(sn, user1, distance):
    friends = []

    #loops through friends of the user and adds them all to the friends list
    #subtracts distance to keep count of round number
    for user in sn[user1][1]:
        friends.append(user)
        dist = distance -1


    while(dist > 0):
```

```
        friends_copy = []

        #makes a copy of list friends
        for friend in friends:
            friends_copy.append(friend)

        #goes through the copied list of friends and adds friends from list
        for friend in friends_copy:
            for user in sn[friend][1]:
                if user != user1 and user not in friends:
                    friends.append(user)

        dist = dist - 1


    return(friends)

#saves a dictionary to a csv file
def save_network(filename, sn):
    ##returns a CSV File
    try:
        csv_file = open(filename, 'w')
        writer = csv.writer(csv_file)
        #loops through social network and writes it to the csv file
        for (username, friends) in sn.items():
            writer.writerow([username] + [friends[0]]+ [friends[1]])
    except FileNotFoundError:
        raise FileNotFoundError("No file found")
    except Exception as e:
        raise e

##returns a dictionary from a filename
def load_network(filename):
    try:

        social_network = {}
        csv_file = open(filename, 'r')
        reader = csv.reader(csv_file)

        #reads the file, and adds the username key and (fullname, friends) as values
        #in the dictionary
        for line in reader:
            if len(line)<1:
                continue
```

```
            username = line[0]
            fullname = line[1]
            friends = line[2]
            social_network[username] = (fullname, friends)
        return(social_network)

    except FileNotFoundError:
        raise FileNotFoundError("No File was found")
    except Exception as e:
        raise e

def main():
    social_network = {}

    ## adding users to the dictionary social_network
    add_user(social_network, "rylee", "Rylee Texter")
    print("add_user():")
    print("social network before")
    print(social_network)
    add_user(social_network, "maddie", "Madison Gamache")
    print("social network after: ")
    print(social_network)
    print()

    ## adding connections between those users
    print("add_friend()")
    print("before add_friend()")
    print(social_network)
    add_friend(social_network, "rylee", "maddie")
    print("after add_friend")
    print(social_network)
    print()


    add_user(social_network, "amy", "Amy Texter")
    add_user(social_network, "tyler", "Tyler Broyles")
    add_user(social_network, "celina", "Celina O")
    add_user(social_network, "arwen", "Arwen Finwells")
    add_user(social_network, "ash", "Ash Irvine")
    add_user(social_network, "megan", "Megan Cooley")
    add_user(social_network, "dylan", "Dylan Vargo")


    add_friend(social_network, "rylee", "ella")
```

```
    add_friend(social_network, "rylee", "amy")
    add_friend(social_network, "rylee", "tyler")
    add_friend(social_network, "maddie", "celina")
    add_friend(social_network, "amy", "arwen")
    add_friend(social_network, "ash", "tyler")
    add_friend(social_network, "megan", "arwen")
    add_friend(social_network, "ash", "dylan")

    #finding friends at a distance of 3
    print("get_friends()")
    friends = get_friends(social_network, "rylee",3)
    print("friends: {}".format(friends))
    print()

    #saving social_network to a csv_file
    print("save_network()")
    print("network saved")
    save_network("socialNetowrk0", social_network)
    print()

    #loading the file and returning the dictionary
    print("load_network()")
    sn1 = load_network("socialNetowrk0")
    print(sn1)


main()
```

Output Results

```
Console 1/A

In [96]: runfile('C:/Users/rylee/Desktop/p3_Texter_Rylee.py', wdir='C:/Users/rylee/Desktop')
add_user():
social network before
{'rylee': ('Rylee Texter', [])}
social network after:
{'rylee': ('Rylee Texter', []), 'maddie': ('Madison Gamache', [])}

add_friend()
before add_friend()
{'rylee': ('Rylee Texter', []), 'maddie': ('Madison Gamache', [])}
after add_friend
{'rylee': ('Rylee Texter', ['maddie']), 'maddie': ('Madison Gamache', ['rylee'])}

get_friends()
friends: ['maddie', 'amy', 'tyler', 'celina', 'arwen', 'ash', 'megan', 'dylan']

save_network()
network saved

load_network()
{'rylee': ('Rylee Texter', "['maddie', 'amy', 'tyler']"), 'maddie': ('Madison Gamache', "['rylee', 'celina']"), 'amy': ('Amy Texter', "['rylee',
'arwen']"), 'tyler': ('Tyler Broyles', "['rylee', 'ash']"), 'celina': ('Celina O', "['maddie']"), 'arwen': ('Arwen Finwells', "['amy', 'megan']"), 'ash':
('Ash Irvine', "['tyler', 'dylan']"), 'megan': ('Megan Cooley', "['arwen']"), 'dylan': ('Dylan Vargo', "['ash']")}

In [97]:
```

Problem 4:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Feb  8 15:40:39 2024

@author: rylee
"""

##Problem 4


#file one: imdb-top-rated.csv, Rank,Title,Year,IMDB Rating
#file two: imdb-top-grossing.csv, Rank,Title,Year,USA Box Office
#file three: imdb-top-casts.csv, Title,Year,Director,Actor1,Actor2,Actor3,Acotr4,Actor5

import csv

#displays the ranking of tuples (director, actor, # of Movies) for movies in which
#the director and actor worked together and the movie is also rated in the top
#rated movie list
def display_top_collaborations():
    ranking_of_tuples = []
    sortedTuples = []
    collabs = []

    #opens two cvs files
    file3 = open("imdb-top-casts.csv", 'r', encoding="utf-8")
    file2 = open("imdb-top-grossing.csv", 'r', encoding="utf-8")
    reader = csv.reader(file3)
    reader2 = csv.reader(file2)

    #loops through each line of the file and assigns connections in
    #list 'collabs'
    for line in reader:
        director = line[2]
        actor1 = line[3]
        actor2 = line[4]
        actor3 = line[5]
        actor4 = line[6]
        actor5 = line[7]
        movie = line[0]

        connection1 = (director, actor1, movie)
        connection2 = (director, actor2, movie)
```

```
    connection3 = (director, actor3, movie)
    connection4 = (director, actor4, movie)
    connection5 = (director, actor5, movie)

    connections = [connection1, connection2, connection3, connection4, connection5]
    for connection in connections:
        collabs.append(connection)

##loops through each collab and removes movies that are not on the top
#grossing list
for collab in collabs:
    movie = collab[2]
    isHere = False
    for line in reader2:
        if movie == line[1]:
            isHere = True
    if(isHere == False):
        collabs.remove(collab)

#loops through collabs and appends tuples containing the movie director,
#actor, and the number of movies together
for collab in collabs:
    director = collab[0]
    actor = collab[1]
    #count is number of movies together
    count = 0
    for connection in collabs:
        if connection[0] == director and connection[1] == actor:
            count = count + 1

    tup = (director, actor, count)
    ranking_of_tuples.append(tup)

nums = []

#loops through the tuples to append the movies to nums
#same number of movies is removed in nums
for i in ranking_of_tuples:
    movieCount = i[2]
    isHere = False
    if movieCount in nums:
        isHere = True
    if(isHere == False):
        nums.append(movieCount)
```

```
    ##sorts nums inr reverse
    nums.sort(reverse=True)

    #sorts the data using nums
    for num in nums:
        for tup in ranking_of_tuples:
            if tup[2] == num:
                sortedTuples.append(tup)


    #prints top ten tuples
    for i in range(0,10):
        print(sortedTuples[i])



#displays the ranking of actors from the top grossing list ordered by the
#total box office money they acted in
def display_top_actors():

    actorsBoxOffice = {}
    movies = []
    movies_boxoffice = {}
    actors = []
    file3 = open("imdb-top-casts.csv", 'r', encoding="utf-8")
    file2 = open("imdb-top-grossing.csv", 'r', encoding="utf-8")
    reader = csv.reader(file2)

    #loops through top grossing list and appends every movie to a list movies
    # movie: boxOffice is also added here to movies_boxOffice dictionary
    for line in reader:
        movie = line[1]
        movies.append(movie)
        movies_boxoffice[movie] = line[3]


    reader = csv.reader(file3)


    #updates actors list with new actors and updates actorsBoxOffice dictionary
    #with actor :box office
    for line in reader:
        if line[0] in movies:
```

```
        current_actors = []
        current_actors.append(line[3])
        current_actors.append(line[4])
        current_actors.append(line[5])
        current_actors.append(line[6])
        current_actors.append(line[7])
        for an_actor in current_actors:
            if an_actor in actors:
                actorsBoxOffice[an_actor] += movies_boxoffice[line[0]]
            else:
                actors.append(an_actor)
                actorsBoxOffice[an_actor] = movies_boxoffice[line[0]]


    issorted = (sorted(actorsBoxOffice.items(), key = lambda item:float(item[1]),
reverse=True))

    #prints the sorted values

    for actor,money in issorted:
        print("{}:{}".format(actor, money))


def main():
    print("display top collaborators: (first ten)")
    display_top_collaborations()

    # print("display top actors")
    # display_top_actors()

main()
```

Output Screenshots:

```
Console 1/A

In [113]: runfile('C:/Users/rylee/Desktop/p4_Texter_Rylee.py', wdir='C:/Users/rylee/Desktop')
display top collaborators: (first ten)
('Francis Ford Coppola', 'Robert Duvall', 3)
('David Yates', 'Daniel Radcliffe', 3)
('David Yates', 'Daniel Radcliffe', 3)
('David Yates', 'Daniel Radcliffe', 3)
('Barry Sonnenfeld', 'Will Smith', 3)
('Barry Sonnenfeld', 'Will Smith', 3)
('Barry Sonnenfeld', 'Will Smith', 3)
('Francis Ford Coppola', 'Robert Duvall', 3)
('Francis Ford Coppola', 'Robert Duvall', 3)
('Peter Jackson', 'Sean Astin', 3)

In [114]:
```

Top actors:

```
Kathryn Hunter:292000866
Georgie Henley:291709845
Skandar Keynes:291709845
William Moseley:291709845
Anna Popplewell:291709845
Tilda Swinton:291709845
Henry Cavill:291021565
Michael Shannon:291021565
Eric Sykes:289994397
Timothy Spall:289994397
David Tennant:289994397
Mary Gibbs:289907418
James Coburn:289907418
Roberts Blossom:285761243
Ray Anthony:281492479
Christine Anu:281492479
Andy Arness:281492479
Alima Ashton-Sheibu:281492479
Helmut Bakaitis:281492479
Ty Olsson:281275991
Barbra Streisand:279167575
John Lithgow:267652016
Vincent Cassel:267652016
Andrew Garfield:262030663
Rhys Ifans:262030663
Holly Hunter:261437578
Dominique Louis:261437578
Taylor Momsen:260031035
Jeffrey Tambor:260031035
Christine Baranski:260031035
Bill Irwin:260031035
```