**Homework 3**

**Jamie Andrews**

**Problem 1:**

#------------Homework 4 Question 1 by Jamie Andrews-------------

#1. Import csv library.

#2. Open the file imdb-top-casts.csv and create a dictionary.

#3. Create a for loop in which it reads the rows in top casts file.

#4. Open the file imdb-top-rated.csv and create a dictionary.

#5. Create a for loop in which it reads the rows in the top rated file.

#6. Open the file imdb-top-grossing.csv and create a dictionary.

#7. Create a for loop in which it reads the rows for the top grossing file.

#8. Create the necessary variables and assign them to list().

#9.

#1.

import csv

#2.

reader1 = csv.reader(open("imdb-top-casts.csv", "rt",encoding="utf8"))

cast={}

#3.

for row in reader1:

cast[row[0]]=[row[1], row[2], row[3], row[4], row[5], row[6], row[7]]

#4.

reader2 = csv.reader(open("imdb-top-rated.csv", "rt",encoding="utf8"))

rated={}

#5.

for row in reader2:

rated[row[1]]=[row[1], row[2], row[3]]

#6.

reader3 = csv.reader(open("imdb-top-grossing.csv", "rt",encoding="utf8"))

gross={}

#7.

for row in reader3:

gross[row[0]]=[row[1], row[2], row[3]]

#8.

rated\_title = list()

rated\_year = list()

rated\_rating = list()

gross\_title = list()

gross\_year = list()

gross\_box = list()

#9.

with open('imdb-top-rated.csv', 'r') as raw\_data:

for line in raw\_data:

if line.startswith('Classification'):

continue # skip the header line

line = line.strip().split(',')

rated\_title.append(line[1])

rated\_year.append(line[2])

rated\_rating.append(line[3])

#10.

with open('imdb-top-grossing.csv', 'r') as raw\_data:

for line in raw\_data:

if line.startswith('Classification'):

continue # skip the header line

line = line.strip().split(',')

gross\_title.append(line[1])

gross\_year.append(line[2])

gross\_box.append(line[3])

#--------------------------------------------------------------------

#a.) displays a ranking(descending of the movie

# directors with the most movies in the top rated list.

# print only the top 5 directors, with a proper title above.

# imdb-top-rated.csv

count = 0

for i in range(len(rated\_title)):

if rated\_title[i] == cast.items().next():

count = count + 1

else:

count = count + 0

print(count)

#--------------------------------------------------------------------

#b) Displays a ranking (descending) of the directors with the

#most movies in the top grossing list. Print

#only the top 5 directors, with a proper title above.

#COUNTER

#--------------------------------------------------------------------

#c) Displays a ranking (descending) of the actors with

#the most movie credits from the top rated list.

#Print only the top 5 actors, with a proper title above.

#--------------------------------------------------------------------

#d)Displays a ranking of movies (descending) based on a

#combined rating/grossing score. The score for

#a movie with rating rank r and grossing rank g is 500-r-g.

#Exclude movies that are only on one list and

#not on the other. Print only the top 5 movie titles, with their

#year, with a proper title above.

#--------------------------------------------------------------------

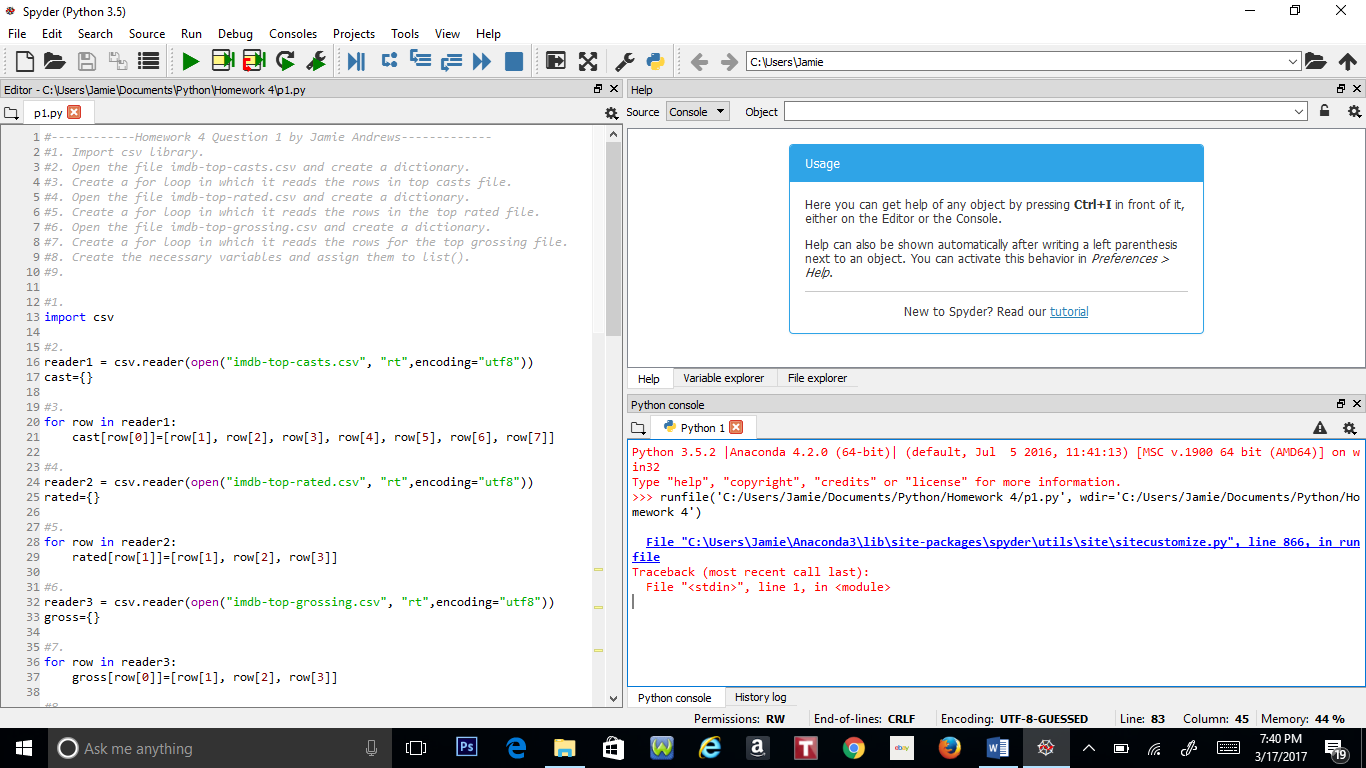
#e) Displays a ranking (descending) with the actors who brought

#in the most box office money, based on

#the top grossing movie list. For a movie with gross ticket sales

#amount s, the 5 actors on the cast list

#will split amount s in the following way:



**Problem 2:**

#--------Homework 4 Problem 2 by Jamie Andrews--------

#1. Create a class called Poly.

#2. Create a constructor in that class.

#3. Set up a list in class.

#4. Create a \_\_str\_\_ def function in the class.

#5. Create a \_\_repr\_\_ def function in the class.

#6. Create a \_\_getitem\_\_ def function in the class.

#7. Create an \_\_add\_\_ def function in the class.

#8. Create a \_\_mul\_\_ def function in the class.

#9. Create a \_\_rmul\_\_ def function in the class.

#10. Create a \_\_eq\_\_ def function in the class.

#11. Create a \_\_ne\_\_ def function in class.

#12. Create a def function in class called eval().

#1.

class Poly(object):

#2.

def \_\_init\_\_(self, numbers):

if isinstance(numbers, str):

self.a,self.b,self.c = numbers.split(',')

elif isinstance(numbers, tuple) or isinstance(numbers,list):

self.a,self.b,self.c = numbers[0],numbers[1],numbers[2]

#3.

self.mylist = [self.a,self.b,self.c]

#4.

def \_\_str\_\_(self):

result = self.a,'+',self.b,'x','+',self.c,'x^2'

self.result = ''.join(map(str,result))

return self.result

#5.

def \_\_repr\_\_(self):

return self.result

#6.

def \_\_getitem\_\_(self,p):

return self.mylist[p]

#7.

def \_\_add\_\_(self,other):

a,b,c = self.a + other.a, self.b + other.b, self.c + other.c

p3 = Poly((a,b,c))

return p3

#8.

def \_\_mul\_\_(self,other):

a,b,c = self.a \* other.a, self.b \* other.b, self.c \* other.c

p3 = Poly((a,b,c))

return p3

#9.

def \_\_rmul\_\_(self,number):

a,b,c = self.a \* number, self.b \* number, self.c \* number

p3 = Poly((a,b,c))

return p3

#10.

def \_\_eq\_\_(self,other):

first = self.a,self.b,self.c

second = other.a,other.b,other.c

if first == second:

return True

else:

return False

#11.

def \_\_ne\_\_(self,other):

first = self.a,self.b,self.c

second = other.a,other.b,other.c

if first != second:

return True

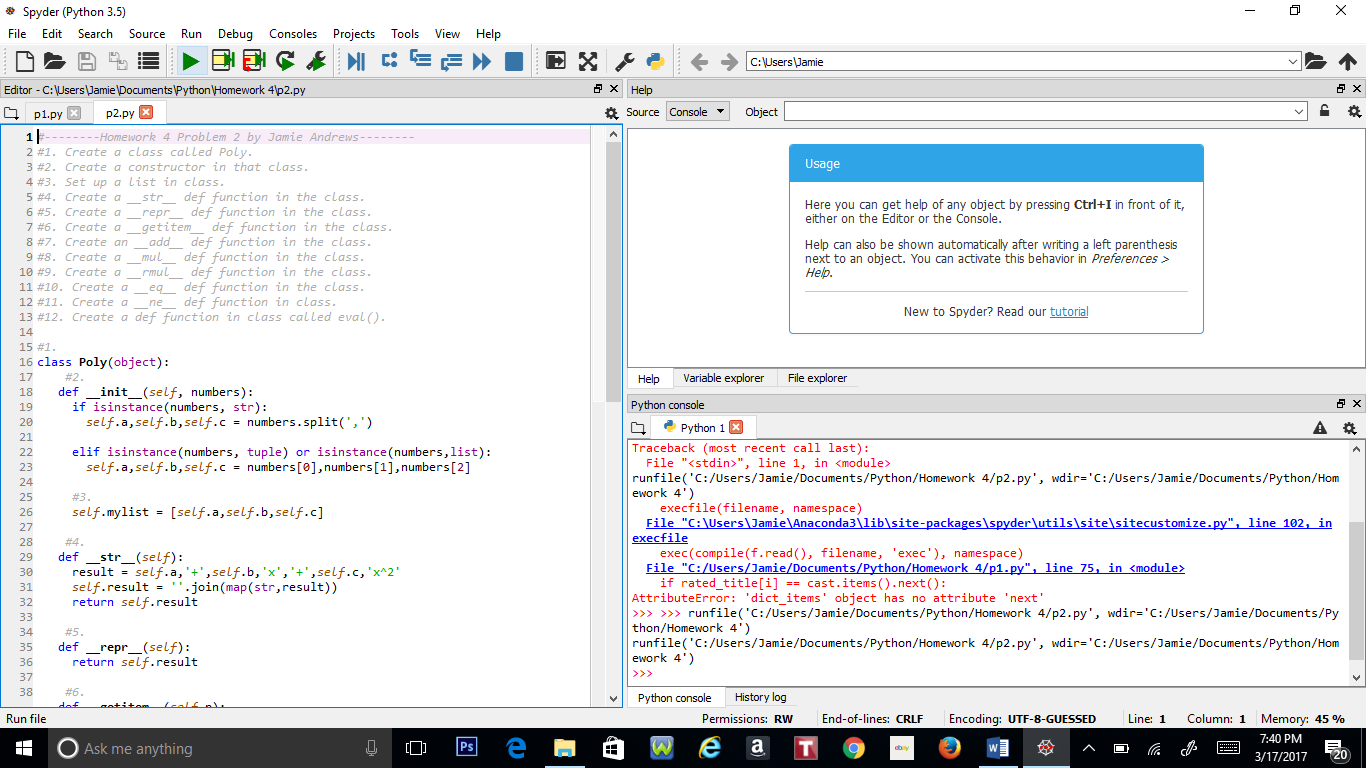
else:

return False

#12.

def eval():

pass



**Problem 3:**

#---------Homework 4 Problem 3 by Jamie Andrews---------

#1. Do Employee SuperClass.

#2. Create Constructor in the Employee Class using variables from the private class.

#3. Create an \_\_str\_\_ def function in the SuperClass.

#4. Create a \_\_repr\_\_ def function in the SuperClass.

#5. Create a def function called salary\_total in the SuperClass, which calculates the total of all the salaries earned by the employees in the starter companies

#6. Create a print\_staff function in the SuperClass, which is supposed to print out the lists of all the names, phone numbers and salaries of every employee.

#7. Create a subclass for the SuperClass called Manager.

#8. Create a constructor and two def functions called \_\_str\_\_ and \_\_repr\_\_ into the Manager Class, passing the attributes from the SuperClass.

#9. Crate a sublclass for the SuperClass called Engineer.

#10. Repeat step 8.

#11. Create a sublclass for Manager called CEO.

#12. Repeate step 8.

#1.

class Employee(object):

#2.

def \_\_init\_\_(self,name,salary,phone):

self.\_\_name = str(name)

self.\_\_salary = float(salary)

self.\_\_phone = str(phone)

self.total = Employee.salary\_total(self)

#3.

def \_\_str\_\_(self):

self.mylist1 = (self.\_\_name,",", self.\_\_phone, ",",self.\_\_salary)

self.mylist1 = ''.join(map(str,self.mylist1))

self.mylist2 = (self.\_\_name,",", self.\_\_phone, ",",self.total)

self.mylist2 = ''.join(map(str,self.mylist2))

return '(' + self.mylist1 + ')' + '\n' + 'Complete Output: ' + self.mylist2

#4.

def \_\_repr\_\_(self):

return '(' + self.mylist + ')'

#5.

def salary\_total(self):

self.total = self.salary

self.bonus = input("Bonus: ")

self.benefits = input("Benefits: ")

self.total = float(self.total) + float(self.bonus) + float(self.benefits)

return self.total

#6.

def print\_staff(self):

self.completelist = [self.\_\_name, ",", self.\_\_phone, ",",self.total]

self.completelist = ''.join(map(str,self.completelist))

return self.completelist

#7.

class Manager(Employee):

#8.

def \_\_int\_\_(self,name,salary,phone):

Employee.\_\_init\_\_(name,salary,phone)

def \_\_str\_\_(self):

self.result = Employee.\_\_str\_\_(self)

return 'Manager: (' + self.mylist1 + ')' + '\n' + 'Complete Output: ' + self.mylist2

def \_\_repr\_\_(self):

return 'Manager: (' + self.mylist1 + ')' + '\n' + 'Complete Output: ' + self.mylist2

#9.

class Engineer(Employee):

#10.

def \_\_int\_\_(self,name,salary,phone):

Employee.\_\_init\_\_(name,salary,phone)

def \_\_str\_\_(self):

self.result = Employee.\_\_str\_\_(self)

return 'Engineer: (' + self.mylist1 + ')' + '\n' + 'Complete Output: ' + self.mylist2

def \_\_repr\_\_(self):

return 'Engineer: (' + self.mylist1 + ')' + '\n' + 'Complete Output: ' + self.mylist2

#11.

class CEO(Manager):

#12.

def \_\_int\_\_(self,name,salary,phone):

Manager.\_\_init\_\_(name,salary,phone)

def \_\_str\_\_(self):

self.result = Manager.\_\_str\_\_(self)

return 'CEO: (' + self.mylist1 + ')' + '\n' + 'Complete Output: ' + self.mylist2

def \_\_repr\_\_(self):

return 'CEO: (' + self.mylist1 + ')' + '\n' + 'Complete Output: ' + self.mylist2

