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.

#---

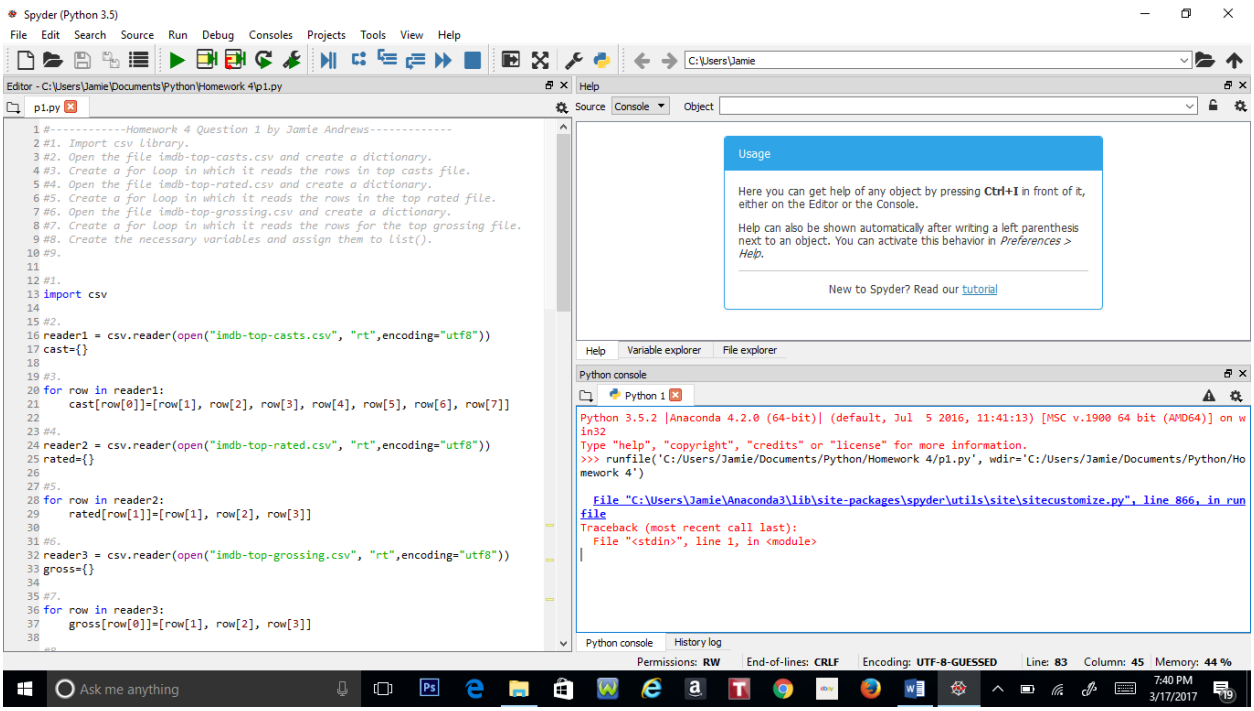
#e) Displays a ranking (descending) with the actors who brought

#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().

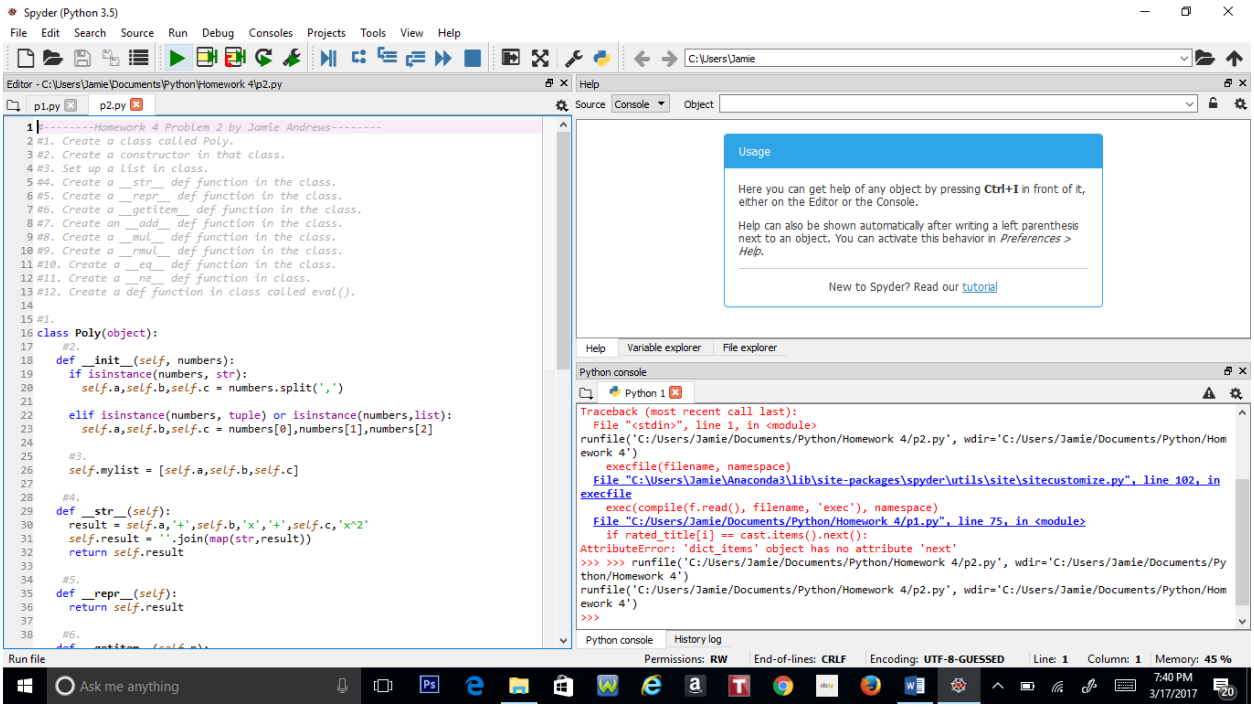
#1.

```
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.

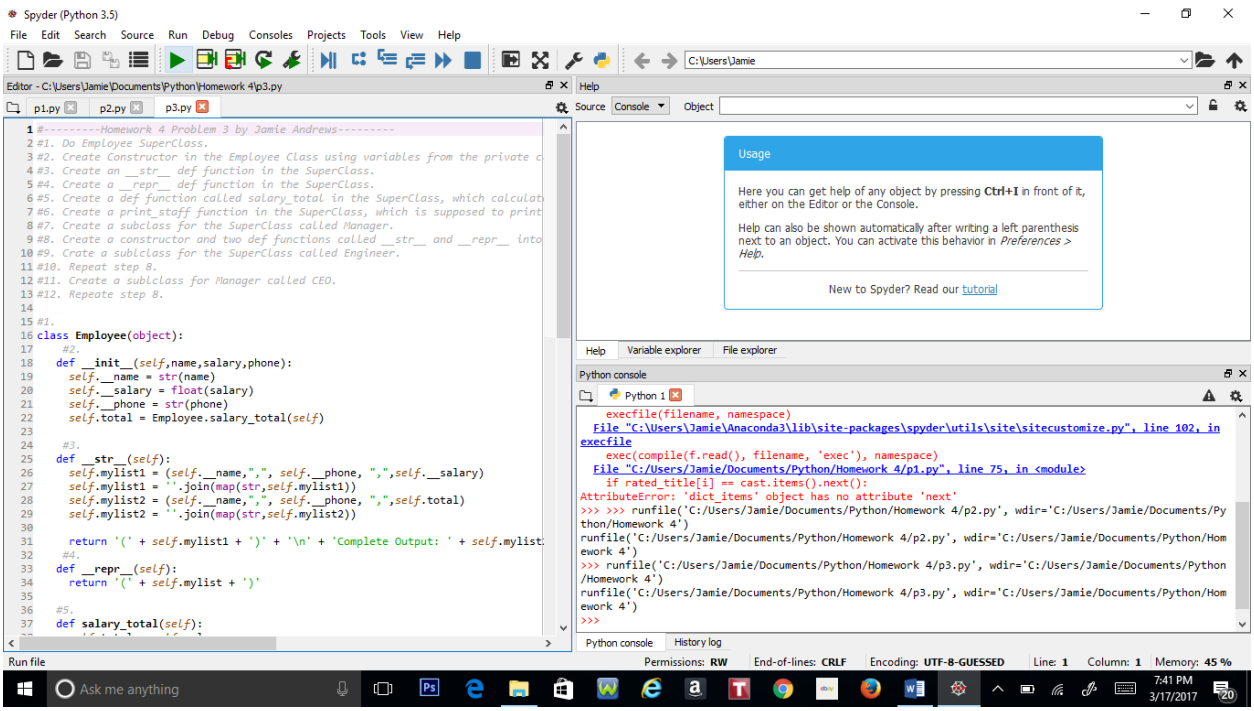
#11. Create a sublclass for Manager called CEO.

#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
```


Spyder (Python 3.5)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - C:\Users\Jamie\Documents\Python\Homework 4\p3.py
p1.py p2.py p3.py
#---------Homework 4 Problem 3 by Jamie Andrews---------
#1. Do Employee SuperClass.
#2. Create Constructor in the Employee Class using variables from the private c
#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 calculat
#6. Create a print_staff function in the SuperClass, which is supposed to print
#7. Create a subclass for the SuperClass called Manager.
#8. Create a constructor and two def functions called __str__ and __repr__ into
#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.mylist
#4.
def __repr__(self):
return '(' + self.mylist + ')'
#5.
def salary_total(self):
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Help Variable explorer File explorer
Python console
Python 1
execfile(filename, namespace)
File "C:\Users\Jamie\Anaconda3\lib\site-packages\spyder\utils\site\sitecustomize.py", line 102, in execfile
exec(compile(f.read(), filename, 'exec'), namespace)
File "C:/Users/Jamie/Documents/Python/Homework 4/p1.py", line 75, in <module>
if rated_title[i] == cast.items().next():
AttributeError: 'dict_items' object has no attribute 'next'
>>> >>> runfile('C:/Users/Jamie/Documents/Python/Homework 4/p2.py', wdir='C:/Users/Jamie/Documents/Python/Homework 4')
runfile('C:/Users/Jamie/Documents/Python/Homework 4/p2.py', wdir='C:/Users/Jamie/Documents/Python/Homework 4')
>>> runfile('C:/Users/Jamie/Documents/Python/Homework 4/p3.py', wdir='C:/Users/Jamie/Documents/Python/Homework 4')
runfile('C:/Users/Jamie/Documents/Python/Homework 4/p3.py', wdir='C:/Users/Jamie/Documents/Python/Homework 4')
>>>
Python console History log
Run file
Permissions: RW End-of-lines: CRLF Encoding: UTF-8-GUESSED Line: 1 Column: 1 Memory: 45 %
Ask me anything
7:41 PM
3/17/2017