CSC 157

Name \_James Aniciete\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date \_3/8/2020\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lab No. \_\_07\_\_\_\_\_

Purpose of the Lab Activity

**The purpose of this lab activity is to use a Student class, various student objects, logic, and files to process students’ grades.**

Given Source Code

class Student :

Scores = {}

# initializing the constructor method

def \_\_init\_\_(self, name, grade) :

self.name = name

self.grade = grade

def getScores(self) :

# begin function to obtain scores #

answer\_key = []

# read into answer\_key list, the answer key from file

answer\_key = [line.strip() for line in open("answers.txt", 'r')]

student\_answers = []

# read into student\_answers list, student answers from file

student\_answers = \

[line.strip().split(',') for line in open("data.txt", 'r')]

total\_score = 100

Student.Scores[self.getName()] = total\_score

def getName(self) :

return self.name;

@staticmethod

def sortDict() :

return sorted(Student.Scores.items());

#---end the class definition#

student\_objs = [Student('Sammy Student',65),

Student('Betty Sanchez', 45),

Student('Alice Brown', 100),

Student('Tom Schulz', 50),

]

for index in range(len(student\_objs)):

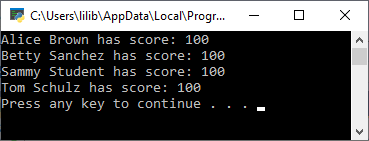
student\_objs[index].getScores()

sortList = Student.sortDict();

for k, v in sortList:

print(k, "has score:", v)

Given Source Code - Snippet(s) of Output(s)



Modified Source Code

# Programmer: James Aniciete

# Course No.: CSC 157

# Lab 07: Object Oriented Programming with Classes

class Student :

Scores = {}

# initializing the constructor method

def \_\_init\_\_(self, name, grade) :

self.name = name

self.grade = grade

def getScores(self) :

# begin function to obtain scores #

answer\_key = []

# read into answer\_key list, the answer key from file

answer\_key = [line.strip() for line in open("answers.txt", 'r')]

student\_answers = []

# read into student\_answers list, student answers from file

student\_answers = \

[line.strip().split(',') for line in open("data.txt", 'r')]

total\_score = 100

# outer loop: figures out which sublist the student is in

# inner loop: grades the student's paper by checking each answer with the answer key,

# where each incorrect answer is a 10 pt deduction

for i in range(len(student\_answers)):

if self.getName() == student\_answers[i][0]:

for j in range(len(student\_answers[0])-1):

if student\_answers[i][j+1] != answer\_key[j]:

total\_score -= 10

# add dictionary entry for (key,value) = (student name, student's score)

Student.Scores[self.getName()] = total\_score

def getName(self) :

return self.name;

@staticmethod

def sortDict() :

return sorted(Student.Scores.items());

#---end the class definition#

student\_objs = [

Student('Sammy Student',65),

Student('Betty Sanchez', 45),

Student('Alice Brown', 100),

Student('Tom Schulz', 50),

Student('James Aniciete', 100) # I guess "a" for each question

]

for index in range(len(student\_objs)):

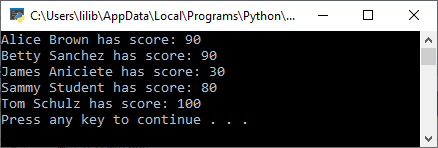
student\_objs[index].getScores()

sortList = Student.sortDict();

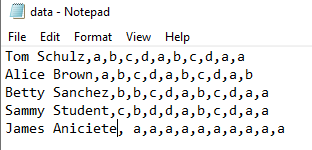
for k, v in sortList:

print(k, "has score:", v)

Snippet(s) of Output(s) from execution of modified Code



Modified data.txt file



Extra Credit – Source Code:

# Programmer: James Aniciete

# Course No.: CSC 157

# Lab 07: Object Oriented Programming with Classes

class Student :

Scores = {}

Ranges = {}

Averages = {}

Grades = []

# initializing the constructor method

def \_\_init\_\_(self, name, grade) :

self.name = name

self.grade = grade

def getScores(self) :

# begin function to obtain scores #

answer\_key = []

# read into answer\_key list, the answer key from file

answer\_key = [line.strip() for line in open("answers.txt", 'r')]

student\_answers = []

# read into student\_answers list, student answers from file

student\_answers = \

[line.strip().split(',') for line in open("data.txt", 'r')]

total\_score = 100

# outer loop: figures out which sublist the student is in

# inner loop: grades the student's paper by checking each answer with the answer key,

# where each incorrect answer is a 10 pt deduction

for i in range(len(student\_answers)):

if self.getName() == student\_answers[i][0]:

for j in range(len(student\_answers[0])-1):

if student\_answers[i][j+1] != answer\_key[j]:

total\_score -= 10

# add dictionary entry for (key,value) = (student name, student's score)

Student.Scores[self.getName()] = total\_score

return Student.Scores[self.getName()]

def getName(self) :

return self.name;

def getGrade(self):

return self.grade

def putGradeInList(self):

Student.Grades.append(self.grade)

return Student.Grades

# static method to sort each dictionary

@staticmethod

def sortScores() :

return sorted(Student.Scores.items());

@staticmethod

def sortRanges():

return sorted(Student.Ranges.items());

@staticmethod

def sortAverages():

return sorted(Student.Averages.items());

# static method that calculates average of all students' scores on the new quiz

@staticmethod

def getClassAverage():

total = 0

average = 0.0

for key in Student.Scores:

total += Student.Scores[key]

average = total/len(Student.Scores)

return average

# static method to calc overall average of students' grades (both old and new quiz grades)

@staticmethod

def getOverallClassAverage():

total = 0

average = 0.0

i = 0 # counter to iterate over Grades list

for key in Student.Scores:

total += (Student.Scores[key] + Student.Grades[i])

i += 1

average = total/(2\*len(Student.Scores)) # multiply by 2 since using 2 grades per student

return average

# method to get a student's average quiz grade

# old grade = self.grade

# new grade = corresponding value in Student.Scores dictionary

def getAverages(self):

average = 0

average = (self.grade + Student.getScores(self))/2

Student.Averages[self.getName()] = average

return Student.Averages[self.getName()]

# here, grade range is assumed to be |self.grade - Student's score|

# i.e. |old quiz grade - new quiz grade|

def getRanges(self):

student\_range = abs(self.getScores() - self.grade)

Student.Ranges[self.getName()] = student\_range

return Student.Ranges[self.getName()]

# static method to get class average range

@staticmethod

def getClassRange():

class\_average = 0.0

range\_total = 0

for key in Student.Ranges:

range\_total += Student.Ranges[key]

class\_average = range\_total/len(Student.Ranges)

return class\_average

#---end the class definition#

# initialize class objects into a list

student\_objs = [

Student('Sammy Student',65),

Student('Betty Sanchez', 45),

Student('Alice Brown', 100),

Student('Tom Schulz', 50),

Student('James Aniciete', 100) # I guess "a" for each question

]

# call methods to get corresponding values

for index in range(len(student\_objs)):

student\_objs[index].getScores()

student\_objs[index].getRanges()

student\_objs[index].getAverages()

student\_objs[index].putGradeInList()

# sort dictionaries

sortList = Student.sortScores();

sortAverages = Student.sortAverages();

sortRanges = Student.sortRanges();

# print each student's score, quiz grade, and grade range

i = 0

for k, v in sortList:

print(k, "has score:", v, "\taverage quiz grade: %s \tgrade range: %s" % (str(sortAverages[i][1]), str(sortRanges[i][1])))

i+=1

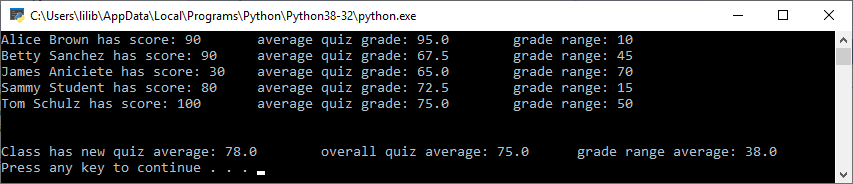
# call methods to print class averages

print("\n") # spacing

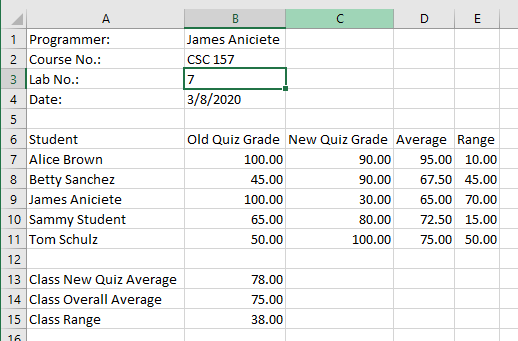
print("Class has new quiz average: {} \toverall quiz average: {} \tgrade range average: {}".format(Student.getClassAverage(), \

Student.getOverallClassAverage(), Student.getClassRange()))

Extra Credit – Output:



Excel Spreadsheet (when Calculations are involved)



Answers to Questions (Be sure to copy the questios themselves!)

**(1)** What is the purpose of this function?

**def \_\_init\_\_()**

**def \_\_init\_\_(self) initializes an object’s data attributes and assigns the *self* parameter to the object.**

**See: Textbook Chapter 10 PowerPoint Slide 14**

**(2)** Is **self** considered a keyword in Python?

**Self is not considered a keyword in Python. The use of self is merely convention as an argument in a method (function), meaning we can replace self with any other name.**

**See:** <https://www.geeksforgeeks.org/self-in-python-class/>

**(3)** What is meant by object - oriented programming?

**Object-oriented programming is a type of programming that focuses on creating objects. The objects have methods (functions) to manipulate and/or extract data and attributes (variables) to store data.**

**See: Textbook Chapter 10 PowerPoint Slide 4**

**(4)** How are text files used in your program code for this project?

**The data in answers.txt is read into a list variable that stores each line from the text file as an element in the list. The data in data.txt is also read into a list variable; however, each line from the text file is stored as a sublist within the list variable. Then, for each sublist, the last 10 elements (the student’s answers) are compared against the answers list variable to grade each student’s quiz.**

**(5)** What have you learned from performing and coding for this lab assignment?

**I have learned how to work with and around the data hiding that comes with using classes and how to use static methods. Also, I learned how to sort a dictionary using a for loop as opposed to the methods that were given in the source code. The loop is at the end of my student.py file.**