**HAUB SCHOOL OF BUSINESS**

**SAINT JOSEPH’S UNIVERSITY**

**DSS 615: Python Programming**

**Instructor: Michael Ghen**

**Assignment 10**

By:

Vinayak Suresh Tayshetye (10673718)

#1

def main():

getUnits("G:/SJU/10 - Python/Assignments/Data Files/Units.txt") #Get unit names from file.

convertDict = createDictionary("G:/SJU/10 - Python/Assignments/Data Files/Units.txt")

convertFrom = input("Units to convert from: ")

convertTo = input("Units to convert to: ")

length = input("Enter length in {:s}: ".format(convertFrom)) #Variable unit by entry.

output = convert(convertDict, convertFrom, convertTo, length)

print("Length in miles: {:.4f}".format(output))

def createDictionary(file): #Create dictionary from file.

global list1

infile = open(file, 'r')

list1 = [line.rstrip() for line in infile]

infile.close()

return dict([x.split(',') for x in list1])

def getUnits(file):

infile = open(file, 'r')

units = []

for line in infile:

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

units.append(line[0])

infile.close()

print("UNITS OF LENGTH")

print("\t\t".join(units[:2]), " ", units[3])

print("\t\t".join(units[3:6]))

print("\t\t".join(units[6:9]))

def convert(dictionary, convertFrom, convertTo, length):

#To find convertFrom unit in dictionary.

for unit in dictionary.keys():

if convertFrom == unit:

feetLength = int(dictionary[unit]) \* int(length) #Calculate length into feet.

#To find convertTo unit in dictionary.

for unit in dictionary.keys():

if convertTo == unit:

convertedLength = feetLength / int(dictionary[unit]) #Calculate length from feet to desired unit.

return(convertedLength)

main()

#Q2

def main():

scores = getScores("G:/SJU/10 - Python/Assignments/Data Files/Scores.txt")

stats = getStats(scores)

examScores = getExamScore(scores, stats) #Converting scores to grades.

freq = freqDict(examScores)

print("Number of scores: {:d}".format(stats[0]))

print("Average score: {:.1f}".format(stats[1]))

print("Standard deviation of scores: {:.2f}".format(stats[2]))

print("GRADE DISTRIBUTION AFTER CURVING GRADES.")

for ch in ["A", "B", "C", "D", "F"]:

print(ch, ":", str(freq[ch]), " ", end = "")

def getScores(file):

infile = open(file, 'r')

scores = [int(line.rstrip()) for line in infile]

infile.close()

return scores

def getStats(list1):

mean = sum(list1) / len(list1)

diff = 0

for num in list1:

diff += (num - mean) \*\* 2

sd = (diff / len(list1)) \*\* 0.5

return len(list1), mean, sd

def getExamScore(list1, stats):

scores = []

for num in list1:

if num >= stats[1] + (1.5 \* stats[2]):

scores.append("A")

elif stats[1] + (.5 \* stats[2]) <= num < stats[1] + (1.5 \* stats[2]):

scores.append("B")

elif stats[1] - (.5 \* stats[2]) <= num < stats[1] + (.5 \* stats[2]):

scores.append("C")

elif stats[1] - (1.5 \* stats[2]) <= num < stats[1] - (.5 \* stats[2]):

scores.append("D")

else:

scores.append("F")

return scores

def freqDict(list1):

freq = {}

for ch in list1:

freq[ch] = 0

for ch in list1:

freq[ch] = freq[ch] + 1

return freq

main()

#Q3

def main():

games = getGames("G:/SJU/10 - Python/Assignments/Data Files/ALE.txt")

gamesWPtc = calculatePercentage(games)

gamesWPtc.sort(key=lambda x: x[3], reverse = True)

createTextFile(gamesWPtc)

def getGames(file):

infile = open(file, 'r')

games = [line.rstrip().split(',') for line in infile]

infile.close()

return games

def calculatePercentage(games):

for team in games:

#Calculate and append win percentages

team.append(str(round(int(team[1]) / (int(team[1]) + int(team[2])), 3)))

return games

def createTextFile(list1):

outfile = open("G:/SJU/10 - Python/Assignments/Data Files/ALE\_OUTPUT.txt", 'w')

for data in list1:

outfile.write(", ".join(data) + "\n")

outfile.close()

main()

#Q6

def main():

cars = getCars("G:/SJU/10 - Python/Assignments/Data Files/Mileage.txt")[0]

carNames = getCars("G:/SJU/10 - Python/Assignments/Data Files/Mileage.txt")[1]

nameFreqDict = freqDict(carNames) #Get frequency of cars

dictTuple = createDictionaryOfTuples(nameFreqDict, cars)

print("{0:10s}{1:s}".format("Model", "MPG"))

outputList = []

for car in dictTuple.keys():

outputList.append([car, 100 / (dictTuple[car][1] / dictTuple[car][0])])

outputList.sort(key=lambda x: x[1], reverse = True)

for data in outputList:

print("{0:10s}{1:.2f}".format(data[0], data[1]))

def createDictionaryOfTuples(nameFreqDict, list1):

dicMileage = {}

for data in list1:

dicMileage[data[0]] = 0

for data in list1: #Loop to fill dict values with mileage totals.

dicMileage[data[0]] = dicMileage[data[0]] + float(data[1])

dictTuple = {}

for key in dicMileage.keys(): #Combine both car count and mileage totals in tuples.

dictTuple[key] = (nameFreqDict[key], dicMileage[key])

return dictTuple

def getCars(file):

infile = open(file, 'r')

cars = [line.rstrip().split(',') for line in infile]

carNames = []

for data in cars:

carNames.append(data[0])

infile.close()

return cars, carNames

main()

#Q7

def main():

infile = open("G:/SJU/10 - Python/Assignments/Data Files/Cities.txt", 'r')

outfile = open("G:/SJU/10 - Python/Assignments/Data Files/Cities\_OUTPUT.txt", 'w')

cities = []

for line in infile:

line = line.rstrip()

data1 = line.split(',')

#Populate cities with desired outputs by making growth calculations.

cities.append([data1[0], ((float(data1[3]) - float(data1[2])) / float(data1[2]))\* 100])

cities.sort(key=lambda x: x[1], reverse = True)

for data in cities:

outfile.write(data[0] + "," + str(round(data[1], 2)) + '\n')

infile.close()

outfile.close()

main()

#Q8

def main():

currency = getFile("G:/SJU/10 - Python/Assignments/Data Files/Exchrate.txt")

countryCurrency(currency)

countriesAscending(currency)

currencyConversion(currency)

def getFile(filename):

currency = []

infile = open(filename, 'r')

currency = [line.rstrip().split(',') for line in infile]

return currency

#a

def countryCurrency(currency):

country = input("Enter the name of a country: ")

for data in currency:

if country.title() == data[0]:

print("Currency: {:s}".format(data[1]))

print("Exchange Rate: {:s}".format(data[2]))

print("\n")

#b

def countriesAscending(currency):

currency.sort(key=lambda x: x[2])

for data in currency:

print(data[0])

print("\n")

#c

def currencyConversion(currency):

curList = []

curNameList = []

for data in currency:

curList.append([data[0], eval(data[2])]) #Appending country and exchange rate.

curNameList.append([data[0], data[1]])

curDict = dict([x for x in curList])

curNameDict = dict([x for x in curNameList])

country1 = input("Enter name of first country: ")

country2 = input("Enter name of second country: ")

money = input("Amount of money to convert: ")

#find rate for country1 in dictionary.

for country in curDict.keys():

if country1.title() == country:

dollarMoney = curDict[country] \* int(money) #Calculate money from country1 currency to dollars.

inExchangeName = curNameDict[country]

#find rate for country2 in dictionary.

for country in curDict.keys():

if country2.title() == country:

convertedMoney = dollarMoney \* curDict[country] #Calculate money from dollars to country2 currency.

outExchangeName = curNameDict[country]

output = "{0:s} {1:s}s from {2:s} equals {3:,.2f} {4:s}s from {5:s}"

print(output.format(money, inExchangeName.lower(), country1.title(), convertedMoney, outExchangeName.lower(), country2.title()))

main()