**HAUB SCHOOL OF BUSINESS**

**SAINT JOSEPH’S UNIVERSITY**

**DSS 615: Python Programming**

**Instructor: Michael Ghen**

**Assignment 9**

By:

Vinayak Suresh Tayshetye (10673718)

* Section 5.1 pp 202-207 Exercises 2-46 even

2. Hello

4. HelloAloha

6. [1, 3, 4]

8. [0, 1, 4]

10. Never give up.

12. ['z', 'o', 'n', 'e']

14. The given path should be in quotes.

16. ABC.txt was opened for reading and next line writes into it.

18. ABC.txt should be opened to append new lines to existing file by passing ‘a’ instead of ‘w’.

20. invalid syntax

22. unhashable type

24. File already exists.

26.

def findItemsInBoth(list1, list2):

return list1.intersection(list2)

28.

30. The new file will contain the names of all the people who subscribe to the *New York  
Times* and the *Wall Street Journal*.

32. The new file will contain the names of the people who subscribe to only the *New York  
Times*. (people subscribed to both journals are removed)

34.

def main():

## Display the number of numbers in the file Numbers.txt

count = getLines("G:/SJU/10 - Python/Assignments/Data Files/Numbers.txt")

print("The file Numbers.txt \ncontains {0} numbers.".format(count))

def getLines(fileName):

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

count = 0

for line in infile:

count += 1

infile.close()

return count

main()

36.

def main():

## Display the smallest number in the file Numbers.txt

smallest = getSmallest("G:/SJU/10 - Python/Assignments/Data Files/Numbers.txt")

print("The smallest number in the \nfile Numbers.txt is {0}.".format(smallest))

def getSmallest(fileName):

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

smallest = int(infile.readline())

for line in infile:

num = int(line)

if num < smallest:

smallest = num

infile.close()

return smallest

main()

38.

def main():

## Display the average of the numbers in the file Numbers.txt.

avg = getSum("G:/SJU/10 - Python/Assignments/Data Files/Numbers.txt")

print("The average of the numbers in \nthe file Numbers.txt is {0:.1f}.".format(avg))

def getSum(fileName):

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

sum1 = 0

count = 0

for line in infile:

sum1 += int(line)

count += 1

infile.close()

avg = sum1/count

return avg

main()

40.

def main():

## Creates a file SomeMonths-With-R.txt for those months which contain letter r from file SomeMonths.txt

getMonths("G:/SJU/10 - Python/Assignments/Data Files/SomeMonths.txt")

def getMonths(fileName):

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

outfile = open("G:/SJU/10 - Python/Assignments/Data Files/SomeMonths-With-R.txt", 'w')

for line in infile:

if 'r' in line:

outfile.writelines(line)

outfile.close

infile.close

main()

42.

def main():

## Creates a file SomeStates-With-VowelBegining.txt for those states from the file that begin with a vowel from file SomeStates.txt

getStates("G:/SJU/10 - Python/Assignments/Data Files/SomeMonths.txt")

def getStates(fileName):

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

outfile = open("G:/SJU/10 - Python/Assignments/Data Files/SomeStates-With-VowelBegining.txt", 'w')

vowels = ['a', 'e', 'i', 'o', 'u']

for line in infile:

if line[0].lower() in vowels:

outfile.writelines(line)

outfile.close

infile.close

main()

44.

def main():

## Creates a file UniquePresStates.txt for those states from the file that begin with a vowel from file PresStates.txt

count = getStates("G:/SJU/10 - Python/Assignments/Data Files/PresStates.txt")

print("{0} different states have".format(count))

print("produced presidents of the \nUnited States.")

def getStates(fileName):

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

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

uniquePresStates = []

for line in infile:

if line not in uniquePresStates:

uniquePresStates.append(line)

outfile.writelines(line)

count = len(uniquePresStates)

outfile.close

infile.close

return count

main()

46.

def main():

name = input("Enter name to be inserted into file: ")

sortName(setNames(name))

def setNames(name):

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

setOfNames = {name for name in infile}

infile.close()

setOfNames.add("\n" + name + "\n")

return setOfNames

def sortName(setOfNames):

listNames = list(setOfNames)

listNames.sort()

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

outfile.writelines(listNames)

outfile.close()

main()

* Section 5.2 pp 212-220 Exercises 2-26 even

2.

Afghanistan is in Asia.

Albania is in Europe.

4.

Afghanistan's pop. density is 126.30 people per sq. mile.

Albania's pop. density is 270.27 people per sq. mile.

6. Each line of the new file contains the names of a European countries and its population in millions.

The countries are shown in descending order by population.

12.

def main():

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

justice(infile)

def justiceSort(fileName):

justices = []

for line in infile:

if (int(line.split(',')[5]) == 0):

justices.append(line)

justices.sort(key=lambda x: int(x.split(',')[4]))

print("Current Justices")

for justice in justices:

print(justice.split(',')[0], justice.split(',')[1])

main()

20. (\*referred from fellow student)

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

infile = open("StatesANC.txt", 'r')

found = False

state\_data = infile.readline()

while (found == False) and (state\_data != ""):

data = state\_data.split(",")

if data[0] == state:

print("Abbreviation:", data[1])

print("Nickname:", data[2])

print("Capital:", data[3].rstrip())

found = True

state\_data = infile.readline()

infile.close()

22. (\*referred from fellow student)

films = open("Oscars.txt", 'r')

incorrect = True

while incorrect:

year = int(input("Enter a year from 1928-2013: "))

if (year >= 1928) and (year <= 2013):

incorrect = False

infile = open("Oscars.txt", 'r')

flicks = [film.rstrip() for film in infile]

infile.close()

film = flicks[year - 1928]

data = film.split(',')

print("Best File:", data[0])

print("Genre:", data[1]

films.close()

else:

print("Year must be between 1928 and 2013.\n")

incorrect = True

24. (\*referred from fellow student)

def main():

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

outfile = open("G:/SJU/10 - Python/Assignments/Data Files/Cowboy2.txt", 'w')

for line in infile:

data = line.split(',')

if data[0] == "Saddle":

newPrice = round(0.8 \* eval(data[1]), 2)

outfile.write("Saddle," + str(newPrice) + "\n")

else:

outfile.write(line)

outfile.close()

infile.close()

main()

26.

def main():

outfile = open("Cowboy.txt", 'a')

outfile.write("Winchester Rifle,20.50\n")

outfile.close()

main()

* Section 5.3 pp 231-235 Exercises 1-44 even, 45-61 odd

2. 6

4. [3.6, 1.3, 6.5, 1.5, 1.1, 0.6]

6. True

8. 1.1

10. CT

12. 5

14. {}

16. CT MA ME NH RI VT

18. 14.6

20. 6

22. 755

24. [('Bonds', 762), ('Aaron', 755)]

26. Bonds

28. False

30. ['Bonds', 'Aaron']

32. NA

34. {'Bonds': 762, 'Aaron': 755, 'Ruth': 714}

36. {'Bonds': 763, 'Aaron': 755, 'Ruth': 714}

38. Bonds

Aaron

Ruth

40. Aaron

Bonds

Ruth

42. 750

44. 755

45.

pres = input("Who was the youngest U.S. president? ")

pres = pres.upper()

correct = "Correct. He became president at age 42\nwhen President McKinley was assasinated."

incorrect = "Incorrect. He became president at age 43. However,\nhe was the youngest person elected president."

dictPres = {"THEODORE ROOSEVELT":correct, "TEDDY ROOSEVELT":correct, "JFK":incorrect, "JOHN KENNEDY":incorrect, "JOHN F. KENNEDY":incorrect}

print(dictPres[pres])

47.

topHitters = {"Gehrig":{"atBats":8061, "hits":2721},

"Ruth":{"atBats":8399, "hits":2823},

"Williams":{"atBats":7706, "hits":2654}}

def main():

for hitter in topHitters.keys():

avg = topHitters[hitter]["hits"]/topHitters[hitter]["atBats"]

print("{0:10s}{1:.3f}".format(hitter, avg))

main()

49.

topHitters = {"Gehrig":{"atBats":8061, "hits":2721},

"Ruth":{"atBats":8399, "hits":2873},

"Williams":{"atBats":7706, "hits":2654}}

def main():

sum = 0

for hits in topHitters.keys():

sum += topHitters[hits]["hits"]

avg = sum / len(topHitters)

print("The average number of hits by\nthe baseball players was {:.1f}.".format(avg))

main()

51.

import pickle

def main():

pres = input("Enter a president: ")

justices = loadDict()

justicesData = justicesAppoınted(pres, justices)

for data in justicesData:

print("{0:18s}{1:d}".format(data[0], data[1]))

def loadDict():

infile = open("G:/SJU/10 - Python/Assignments/Data Files/JusticesDict.dat", 'rb')

justices = pickle.load(infile)

infile.close()

return justices

def justicesAppoınted(pres, dictionary):

list1 = []

for justice in dictionary.keys():

if pres == dictionary[justice]["pres"]:

list1.append([justice, dictionary[justice]["yrAppt"]])

return list1

main()

53.

import pickle

def main():

justiceName = input("Enter name of justice: ")

justices = loadDict()

justicesData = justicesAppoınted(justiceName, justices)

for data in justicesData:

print("Appointed by {0:18s}".format(data[0]))

print("State: {0:s} \nYear of appointment: {1:d}".format(data[1], data[2]))

if data[3] == 0:

print("Currently serving on Supreme Court.")

else:

print("Year left: {0:s}".format(data[3]))

def loadDict():

infile = open("G:/SJU/10 - Python/Assignments/Data Files/JusticesDict.dat", 'rb')

justices = pickle.load(infile)

infile.close()

return justices

def justicesAppoınted(justiceName, dictionary):

list1 = []

for justice in dictionary.keys():

if justiceName == justice:

list1.append([dictionary[justice]["pres"], dictionary[justice]["state"], dictionary[justice]["yrAppt"], dictionary[justice]["yrLeft"]])

return list1

main()

55.

def main():

sentence = input("Enter a sentence: ")

listOfLetters = listLetters(sentence)

freq = dictFreq(listOfLetters)

displayLetters(freq)

def listLetters(string):

listOfLetters = []

string = string.upper()

for ch in string:

if ('A' <= ch <= 'Z'):

listOfLetters.append(ch)

return listOfLetters

def dictFreq(list):

freq = {}

for ch in list:

freq[ch] = 0

for ch in list:

freq[ch] += 1

return freq

def displayLetters(dictionary):

list1 = []

i = 0

for letter in dictionary.keys():

list1.append([letter, dictionary[letter]])

list1.sort(key=lambda x: x[1], reverse=True)

for letter in list1:

if i < 4:

print(" ", letter[0], ':', letter[1])

i += 1

main()

59. (\* referred from text book answers)

def main():

## Determine the day of the week for a date.

calender2015Dict = createDictionary("Calendar2015.txt")

date = input("Enter a date in 2015: ")

print(date, "falls on a", calender2015Dict[date])

def createDictionary(fileName):

infile = open(fileName, 'r')

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

infile.close()

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

main()

61. (\* referred from text book answers)

import pickle

def main():

## Determine states having a specified number of large cities.

largeCities = createDictionaryFromBinaryFile("LargeCitiesDict.dat")

number = int(input("Enter an integer from 1 to 13: "))

states = sorted(getStates(number, largeCities))

displayResult(number, states)

def createDictionaryFromBinaryFile(fileName):

infile = open(fileName, 'rb')

dictionaryName = pickle.load(infile)

infile.close()

return dictionaryName

def getStates(number, dictionaryName):

states = []

for state in dictionaryName:

if len(dictionaryName[state]) == number:

states.append(state)

return states

def displayResult(number, states):

if len(states) == 0:

print("No states have exactly", number, "large cities.")

else:

print("The following states have exactly", number, "large cities:")

print(" ".join(states))

main()