# Lab 10: File Access

This lab accompanies Chapter 10 of *Starting Out with Programming Logic & Design*.

Name: David White

**Lab 10.1 – File Access and Pseudocode**

|  |
| --- |
| Critical Review  When a program needs to save data for later use, it writes the data in a file. The data can be read from the file at a later time.  Three things must happen in order to work with a file. 1) Open a file. 2) Process the file. 3) Close the file.  An internal file must be created for an output file or input file, such as:  Declare OutputFile myFile //to write out  Declare InputFile myFile //to read in  A data file must also be created to store the output, such as:    Open myFile “thedata.txt”  New keywords and syntax include the following:    Open [InternalName] [FileName]  Write [InternalName] [String or Data]  Read [InternalName] [Data]  Close [InternalName]  AppendMode //used with Open when need to append  Loops are used to process the data in a file. For example:    For counter = 1 to 5  Display “Enter a number:”  Input number  Write myFile number  End For  When reading information from a file and it is unknown how many items there are, use the eof function. For example:  While NOT eof(myFile)  Read myFile number  Display number  End While |

This lab examines how to work with a file by writing pseudocode. Read the following programming problem prior to completing the lab. The following program from Lab 9.1 (from week 10) will be used, with some modifications.

The American Red Cross wants you to write a program that will calculate the average pints of blood donated during a blood drive. The program should take in the number of pints donated during the drive, based on a seven hour drive period. The average pints donated during that period should be calculated and written to a file. Write a loop around the program to run multiple times. The data should be appended to the file to keep track of multiple days. If the user wants to print data from the file, read it in and then display it. Store the pints per hour and the average pints donated in a file called blood.txt.

**Step 1:** Note that the getPints, getTotal, and getAverage functions do not change. Also note that the references to displayInfo, getHigh, and getLow functions are removed to meet the new requirements. In the pseudocode below, add the following:

In the Main Module

1. A variable named option of the data type Integer
2. Input option
3. Write an if statement that will determine which option to run
4. Call a module called writeToFile that passes pints and averagePints
5. Call a module called readFromFile that passes pints and averagePints

In the writeToFile Module

1. Declare an output file called outFile in AppendMode, with the name bloodFile. (Reference: Appending Data to an Existing File, Page 480).
2. Open the internal file (bloodFile) and a text file named blood.txt. (Reference: Creating a File and Writing Data to it, Page 472.)
3. Write the string “Pints Each Hour” to the file. (Reference: Writing Data to a File, Page 473).
4. In the while loop, write each element of the pints array to the bloodFile. (Reference: Using Loops to Process Files, Page 481).
5. Write the string “Average Pints” to the file.
6. Write the value of averagePints to the file.
7. Close the bloodFile. (Reference: Closing an Output File, Page 473).

In the readFromFile Module

1. Declare an input file called inFile , with the name bloodFile. (Reference: Reading Data from a File, Page 476).
2. Open the internal file (bloodFile) and a text file named blood.txt.
3. Read the string “Pints Each Hour” in from your file and store into a variable str1. This should be done such as Read bloodFile str1. The string will be stored in the variable str1.
4. Display str1 to the screen.
5. Read pints in from the bloodFile and store in the pints array.
6. Display pints to the screen.
7. Read the string “Average Pints” in from your file and store into a variable str2.
8. Display str2 to the screen.
9. Read averagePints in from the bloodFile.
10. Display averagePints to the screen
11. Close the file. (Reference: Closing an Input File, Page 477).

Module main()

//Declare local variables

Declare String again = “no”

Declare Real pints[7]

Declare Real totalPints

Declare Real averagePints

Declare int option

While again == “no”

//module calls below

Display “Enter 1 to enter in new data and store to file”

Display “Enter 2 to display data from the file”

Input option

c. If option == 1 Then

Call getPints(pints)

Call getTotal(pints, totalPints)

Call getAverage(totalPints, averagePints)

Call writeToFile(pints. averagePints)

Else

e. Call readFromFile(pints, averagePints)

End If

Display “Do you want to run again: yes or no”

Input again

End While

End Module

Module getPints(Real pints[])

Declare Integer counter = 0

For counter = 0 to 6

Display “Enter pints collected:”

Input pints[counter]

End For

End Module

Function getTotal(Real pints[], Real totalPints)

Declare Integer counter = 0

Set totalPints = 0

For counter = 0 to 6

Set totalPints = totalPints + pints[counter] End For

Return totalPints

Function getAverage(Real totalPints, Real averagePints)

averagePints = totalPints / 7

Return averagePints

Module writeToFile(Real pints[], Real averagePints)

f. Declare OutputFile AppendMode bloodFile

g. Open bloodFile “blood.txt”

h. Write bloodFile “Pints Each Hour”

Declare Integer counter = 0

i. While counter < 7

Write bloodFile pints[counter]

Set counter = counter + 1

End While

j. Write bloodFile “Average Pints”

k. Write bloodFile averagePints

l. Close bloodFile

End Module

Module readFromFile(Real pints[], Real averagePints)

m. Declare inputFile BloodFile

n. Open bloodFile “blood.txt”

o. Read bloodFile str1

p. Display str1

q. Read bloodFile pints

r. Display pints

s. Read bloodFile str2

t. Display str2

u. Read bloodFile averagePints

v. Display averagePints

w. Close bloodFile

End Module

**Lab 10.2 – File Access and Flowcharts**

This lab requires you to create a flowchart for the blood drive program in Lab 10.1. Use an application such Visio, Draw.io (must convert to a PDF file before submitting), or simply draw the flowchart by hand and upload a picture of it in JPG format.

**The final step is to upload your finished flowchart to Blackboard.** You may paste your flowchart here as well for reference.



**Lab 10.3 – File Access and Python Code**

Critical Review

Writing to a File

When writing to a file, an internal file name must be created such as outFile.

This file must then be opened using two arguments. The first argument is the name of the file and the second is the mode you want to open the file in. You can select either the ‘a’ append mode or the ‘w’ write mode. For example:

outFile = open(‘filename.txt’, ‘w’)

A string literal can be written to a file, such as:

Print >> outFile, ‘Header Information’

Variables must be converted to strings as they are written to a file and a call to write must occur. Additionally, a ‘\n’ can be appended to cause a return statement in your file. For example:

outFile.write(str(variableName) + ‘\n’)

Arrays are written to a file using a loop. For example:

counter = 0

while counter < 7:

outFile.write(str(arrayName[counter]) + '\n')

counter = counter + 1

Files must then be closed. This works the same for both input and output.

outFile.close() or inFile.close()

Reading from a File

When reading from a file, an internal file name must be created such as inFile.

This file must then be opened using two arguments. The first argument is the name of the file and the second is the mode you want to open the file in, ‘r’ for read. For example:

inFile = open(‘filename.txt’, ‘r’)

Reading from a file is done sequentially in this lab, and a call to read must occur. If a string header is done first, that must be read into a string variable. That variable can then be used for processing within the program.

A string literal can be read from a file and displayed to the screen, such as:

str1 = inFile.read()

print str1

Arrays and variables can be read as a single input, such as:

arrayName = inFile.read()

print arrayName

The goal of this lab is to convert the blood drive program from Lab 10.1 to Python code.

**Step 1:** Start the IDLE Environment for Python. Prior to entering code, save your file by clicking on File and then Save. Select your location and save this file as  *Week13Lab*10-3*.py*. Be sure to include the .py extension.

**Step 2:** Document the first few lines of your program to include your name, the date, and a brief description of what the program does.

**Step** **3:** Start your program with the following code:

#Lab 10-3 Blood Drive

#the main function

def main():

endProgram = 'no'

print

while endProgram == 'no':

option = 0

print

print 'Enter 1 to enter in new data and store to file'

print 'Enter 2 to display data from the file'

option = input('Enter now ->')

print

# declare variables

pints = [0] \* 7

totalPints = 0

averagePints = 0

if option == 1:

# function calls

pints = getPints(pints)

totalPints = getTotal(pints, totalPints)

averagePints = getAverage(totalPints, averagePints)

else:

endProgram = raw\_input('Do you want to end program? (Enter no or yes): ')

while not (endProgram == 'yes' or endProgram == 'no'):

print 'Please enter a yes or no'

endProgram = raw\_input('Do you want to end program? (Enter no or yes): ')

#the getPints function

def getPints(pints):

counter = 0

while counter < 7:

pints[counter] = input('Enter pints collected: ')

counter = counter + 1

return pints

#the getTotal function

def getTotal(pints, totalPints):

counter = 0

while counter < 7:

totalPints = totalPints + pints[counter]

counter = counter + 1

return totalPints

#the getAverage function

def getAverage(totalPints, averagePints):

averagePints = float(totalPints) / 7

return averagePints

#the writeToFile function

def writeToFile(averagePints, pints):

#the readFromFile function

def readFromFile(averagePints, pints):

# calls main

main()

**Step 4:** Under option 1 in main, add a function call to writeToFile and pass it averagePints and pints. This should be done after the other calls. This should look as follows:

writeToFile(averagePints, pints)

**Step 5:** Under option 2 in main, add a function call to readFromFile and pass it averagePints and pints. This should be done after the other calls. This should look as follows:

readFromFile(averagePints, pints)

**Step 6:** Under the documentation and the function header for the writeToFile function, create an outFile and call the open function. Pass this function the name of the text file and open it in append mode. This should look as follows:

outFile = open('blood.txt', 'a')

**Step 7:** The next step is to write the string ‘Pints Each Hour’ to the file. This is done as follows:

outFile.write('Pints each Hour\n')

**Step 8:** Initial counter to 0 and add a while loop with the condition of counter < 7. Inside the while loop, write the value of the array pints to the file. This should look as follows:

outFile.write(str(pints[counter]) + '\n')

counter = counter + 1

**Step 9:** Outside the while loop, write the string ‘Average Pints’ to the file.

**Step 10:** Next, write the averagePints variable to the file. This should look as follows:

outFile.write(str(averagePints) + '\n\n')

**Step 11:** The last item in this function is to close the outFile. This is done as follows:

outFile.close()

**Step 12:** Under the documentation and the function header for the readFromFile function, create an inFile and call the open function. Pass this function the name of the text file and open it in read mode. This should look as follows:

inFile = open('blood.txt', 'r')

**Step 13:** Next, read in the string ‘Pints Each Hour’ and print this to the screen. This is done as follows:

str1 = inFile.read()

print str1

**Step 14:** Read in the pints array as an entire list and print this to the screen. This is done as follows:

pints = inFile.read()

print pints

print #adds a blank line

**Step 15:** Read in the string ‘Average Pints’ and print this to the screen.

**Step 16:** Read in averagePints and print this to the screen.

**Step 17:** Close the inFile.

**Step 18:** Run your program and for the first execution, select option 1. Run the program more than once and enter at least 2 sets of data. The append mode should keep track of everything. The contents of your file will be stored in the same directory that your Python code is in. Paste the contents of the file below:

Pints Each Hour

1

2

3

4

5

6

7

Average Pints

4.0

Pints Each Hour

8

9

10

11

12

13

14

Average Pints

11.0

**Step 19:** Run your program again and select option 2 on the first iteration. This should display to the screen information that is stored in your file.

**Step** **20:** Execute your program so that it works and paste the final code below.

**Step** **21:** Upload your fully tested Python program in Blackboard. Make sure that you include the appropriate header and many comments as those make up 25% of your grade.

#######################################################

# Name: David White

# Class: CIS-1400

# Assignment: Lab 10-3

# File: lab10-3.py

# Purpose: energy efficiency savings

######################################################

print('\n\*\*\*David White\*\*\*\n') # Display author's name

# the main function

def main():

endProgram = 'no'

print()

while endProgram == 'no':

option = 0

print()

print('Enter 1 to enter in new data and store to file')

print('Enter 2 to display data from the file')

option = int(input('Enter now -> '))

print()

# declare variables

pints = [0] \* 7

totalPints = 0

averagePints = 0

if option == 1:

# function calls

pints = getPints(pints)

totalPints = getTotal(pints, totalPints)

averagePints = getAverage(totalPints, averagePints)

writeToFile(averagePints, pints)

else:

readFromFile(averagePints, pints)

endProgram = input('Do you want to end program? (Enter no or yes): ')

while not (endProgram == 'yes' or endProgram == 'no'):

print()

'Please enter a yes or no'

endProgram = input('Do you want to end program? (Enter no or yes): ')

# the getPints function

def getPints(pints):

counter = 0

while counter < 7:

pints[counter] = input('Enter pints collected: ')

counter = counter + 1

return pints

# the getTotal function

def getTotal(pints, totalPints):

counter = 0

while counter < 7:

totalPints = totalPints + int(pints[counter])

counter = counter + 1

return totalPints

# the getAverage function

def getAverage(totalPints, averagePints):

averagePints = float(totalPints) / 7

return averagePints

# the writeToFile function

def writeToFile(averagePints, pints):

outFile = open('blood.txt', 'a')

outFile.write('Pints Each Hour\n')

counter = 0

while counter < 7:

outFile.write(str(pints[counter] + '\n'))

counter = counter + 1

outFile.write('Average Pints\n')

outFile.write(str(averagePints) + '\n\n')

outFile.close()

# the readFromFile function

def readFromFile(averagePints, pints):

inFile = open('blood.txt', 'r')

str1 = inFile.read()

print(str1)

print() # blank line for separation

pints = inFile.read()

print(pints)

print() # blank line for separation

str2 = inFile.read()

print(str2)

inFile.close()

# calls main

main()