SOLUTION CP1200 – Practical 6

Aim

1. Practise writing programs to read and write text files
2. Practise writing programs that can handle exceptions

Reading And Writing Files

Quick Programs

The solutions for these programs are at the bottom of the practical, to help you get going - or to confirm that your solution was valid.

1. Write a program that asks the user for their name, then opens a file called "name.txt" and writes that name to it.
2. Write a program that opens "name.txt" and reads the name (as above) then prints,
   "Your name is Bob" (or whatever the name is in the file).
3. First, create a text file called "numbers.txt" (save it in your project).
   Put the numbers 17 and 42 on separate lines in the file and save it.
   Write a program that opens "numbers.txt", reads the numbers and adds them together then prints the result, which should be… 59.
   Extended (perhaps only do this if you're cruising… if you are struggling, just read the solution):
   Now extend your program so that it can print the total for a file containing any number of numbers.

1. Finishing/Testing Tutorial Problems

Test out your programs from the Files part of the tutorial: 2.1, 2.2 and 3.1.

Tutorial Program 2.1

Make sure your program produces a file with all the odd numbers from 1 up to and including 99.
I.e. run the program and check that the text file contains those numbers.

Tutorial Program 2.2

Download **numbers.txt** from Learn JCU, and make sure your program has the following output:

```
6.22
11.41
22.81
42.59
42.7
77.92
96.42
99.99
```

Tutorial Program 3.1

Test 1

Download **RecentRain.txt** from LearnJCU, and verify the output as follows

```
Enter filename: RecentRain.txt
Enter lower limit: 10
Processing...
6 out of 10 (60.0%) of values are greater than 10.
```

Tests 2 and 3
Download **DailySales.txt** from LearnJCU, and verify the output as follows

```
Enter filename: DailySales.txt
Enter lower limit: 1000
Processing...
183 out of 365 (50.1%) of values are greater than 1000.0.
```

```
Enter filename: DailySales.txt
Enter lower limit: 1599.99
Processing...
34 out of 365 (9.3%) of values are greater than 1599.99.
```

2. Problems For You To Fill In The Blanks

2.1 Very Simple File Reader/Writer

Let's pretend that it's 1977 and we don't have any text editors on our computer (obviously we do if we're programming). We're going to write a very simple file reader/writer.
Some of the code has been provided for you (down below and in a .py file on LearnJCU).
The output will match the following:

```
VSFRW

Menu:
w. write a new file
r. read a file and display to the screen
q. quit
Enter your choice: x
Invalid choice!

Menu:
w. write a new file
r. read a file and display to the screen
q. quit
Enter your choice: r
Enter the filename: TheBeatles.txt
Displaying beatles.txt...
John Lennon
Paul McCartney
George Harrison
Ringo Starr

Menu:
w. write a new file
r. read a file and display to the screen
```

q. quit
Enter your choice: **w**
Enter the filename: **TheClash.txt**
Enter line 1: **Joe Strummer**
Enter another line (y/n)? **que?**
Please enter y or n.
Enter another line (y/n)? **y**
Enter line 2: **Mick Jones**
Enter another line (y/n)? **y**
Enter line 3: **Paul Simonon**
Enter another line (y/n)? **y**
Enter line 4: **Topper Headon**
Enter another line (y/n)? **n**
Wrote 4 lines to theclash.txt

Menu:
w. write a new file
r. read a file and display to the screen
q. quit
Enter your choice: **q**

Adios.

```
def main():
    print("VSFRW")
    print()

    choice = getMenuChoice()
    while choice != 'q':
        if choice == 'w':
            <...>
        elif choice == 'r':
            <...>
        else:
            print("Invalid choice!")
        choice = getMenuChoice()

    print()
    print("Adios.")

# display the menu and return the choice
def getMenuChoice():
    print("Menu:")
    print("w. write a new file")
    print("r. read a file and display to the screen")
    print("q. quit")
    choice = input("Enter your choice: ")

    return choice

def getNonEmptyString(prompt, errorMessage):
    string = input(prompt)
    while string == "":
        string = input(prompt)
        print(errorMessage)

    return string
```

```
# get the user to enter y or n
# return True if y, False if n
def getConfirmation(prompt, errorMessage):
    string = input(prompt).lower()
    while string[0] not in 'yn':
        print(errorMessage)
        string = input(prompt).lower()

    if string == 'y':
        return True
    else:
        return False

def writeNewFile():
    filename = <CALL THE APPROPRIATE FUNCTION>

    <OPEN THE FILE>

    keepGoing = True
    lineCount = 0
    while keepGoing:
        lineCount += 1
        <GET THE LINE FROM THE USER>
        <WRITE THE LINE TO THE FILE>

        keepGoing = getConfirmation("Enter another line (y/n)?", \
                                    "Please enter y or n.")

    print("Wrote", lineCount, "lines to", filename)
    <CLOSE THE FILE>

def readAndDisplayFile():
    filename = <CALL THE APPROPRIATE FUNCTION>

    <OPEN THE FILE>
    <GET ALL THE text FROM THE FILE>
    <CLOSE THE FILE>

    print("Displaying ", filename, "...", sep="")
    print(text)

main()
```

SOLUTION

```
def main():
    print("VSFRW")
    print()

    choice = getMenuChoice()
    while choice != 'q':
        if choice == 'w':
            writeNewFile()
        elif choice == 'r':
            readAndDisplayFile()
        else:
            print("Invalid choice!")
        choice = getMenuChoice()

    print()
    print("Adios.")
```

```
def getMenuChoice():
    print("Menu:")
    print("w. write a new file")
    print("r. read a file and display to the screen")
    print("q. quit")
    choice = input("Enter your choice: ")

    return choice

def getNonEmptyString(prompt, errorMessage):
    string = input(prompt)
    while string == "":
        string = input(prompt)
        print(errorMessage)

    return string

def getConfirmation(prompt, errorMessage):
    string = input(prompt).lower()
    while string[0] not in 'yn':
        print(errorMessage)
        string = input(prompt).lower()

    if string == 'y':
        return True
    else:
        return False

def writeNewFile():
    filename = getNonEmptyString("Enter the filename: ", \
                                 "Filename cannot be blank!")
    fileOut = open(filename, 'w')

    keepGoing = True
    lineCount = 0
    while keepGoing:
        lineCount += 1
        line = input("Enter line " + str(lineCount) + ": ")
        fileOut.write(line + '\n')

        keepGoing = getConfirmation("Enter another line (y/n)?", \
                                    "Please enter y or n.")

    print("Wrote", lineCount, "lines to", filename)
    fileOut.close()

def readAndDisplayFile():
    filename = getNonEmptyString("Enter the filename: ", \
                                 "Filename cannot be blank!")

    fileIn = open(filename, 'r')
    text = fileIn.read()
    fileIn.close()

    print("Displaying ", filename, "...", sep="")
    print(text)

main()
```

3. Problems For You To Solve

3.1 Line-at-a-time File Viewer

Write a program that reads file one line at a time, asking the user to press enter to see the next line.
Example output:

Enter the name of the file (enter nothing to quit): **sales.txt**
Line 1: Sales
Press enter to see line 2...***<ENTER>***
Line 2: 44.81
Press enter to see line 3...***<ENTER>***
Line 3: 109.90
You have reached the end of the file. Press enter to finish...***<ENTER>***

Enter the name of the file (enter nothing to quit): **thebeatles.txt**
Line 1: John Lennon
Press enter to see line 2...***<ENTER>***
Line 2: Paul McCartney
Press enter to see line 3...***<ENTER>***
Line 3: George Harrison
Press enter to see line 4...***<ENTER>***
Line 4: Ringo Starr
You have reached the end of the file. Press enter to finish...***<ENTER>***
Enter the name of the file (enter nothing to quit):***<ENTER>***

Thank you for using the line-at-a-time file viewer.

SOLUTION

```
filename = input("Enter the name of the file (enter nothing to quit): ")
while filename != "":
    fileIn = open(filename, 'r')
    count = 1
    line = fileIn.readline().strip()
    while line != "":
        print("Line", str(count) + ":", line)
        line = fileIn.readline().strip()
        if line != "":
            count += 1
            input("Please enter to see line " + str(count) + "...")
        else:
            input("You have reached the end of the file. Press enter to finish...")

    filename = input("Enter the name of the file (enter nothing to quit): ")
```

3.2 Mail Merge Form Letter Program

SOLUTION – See Python File

Exceptions

1. Testing Tutorial Problem

Test out your `getInteger()` function from the tutorial.
Does it handle bad input correctly?
Does it only return when a valid integer is entered?
E.g. test it out by entering “a”, “a1”, “1”, “-1”.

2. Problem For You To Solve

Rewrite `getValidInteger(prompt, minimum, maximum)` (from earlier pracs) so that it doesn’t crash when the input is non-numerical. **Hint:** just use getInteger() to get the integer.

SOLUTION

```
def getValidInteger(prompt, minimum, maximum):
    result = getInteger(prompt)
    while result < minimum or result > maximum::
        print("Please enter an integer between ", minimum, \
        " and ", maximum, ".", sep="")
        result = getInteger(prompt)
    return result
```

Extension

1. Generating A File Of Random Floating-Point Numbers
   The **DailySales.txt** file you used earlier for testing was not written by hand. Instead of typing out 365 numbers, I wrote a program to do it for me. But alas! I accidentally deleted that program. Please rewrite it for me. Match the sample output as follows:

Enter the filename: **RandomNumbers.txt** Enter the minimum floating-point number: **1.0** Enter the maximum floating-point number: **5.0** Enter the number of random floats to generate: **10**	**RandomNumbers.txt:** 2.7121733154776684 4.014094001639894 3.834851329128939 3.4153513571843725 2.178530126797532 1.6251032860226386 1.8351248558788504 4.866790116359345 3.1386145788551034 3.6058896170001806

SOLUTION

```
from random import uniform

filename = input("Enter the filename: ")
minimum = float(input("Enter the minimum floating-point number: "))
maximum = float(input("Enter the maximum floating-point number: "))
number = int(input("Enter the number of random floats to generate: "))
```

```
outputFile = open(filename, 'w')
for i in range(number):
    outputFile.write(str(uniform(minimum, maximum)) + "\n")
outputFile.close()
```

2. Simple File Statistics

   Given a file with a floating-point number on each line, write a program that will calculate:

   a. the **total**
   b. the **average** (arithmetic mean)
   c. the **minimum**
   d. the **maximum**
   e. **(harder)** the **geometric mean** (look it up on Wikipedia)
   f. **(harder - requires more Python)** the **median** (you will need to use a list)
   g. **(harder - requires more Python)** the **mode** (you will need to use a dictionary)

SOLUTION

```
from math import pow
filename = input("Enter the filename: ")
inputFile = open(filename, 'r')

# read the first line and set min, max, total to the first number
minimum = float(inputFile.readline())
maximum = minimum
total = minimum
product = minimum
count = 1 # assume file has at least one number

# read the other numbers
for line in inputFile:
    number = float(line)
    total += number
    product *= number
    count += 1
    if number < minimum:
        minimum = number
    elif number > maximum: # elif since it can never be both
        maximum = number
inputFile.close()

print("Total:", total)
print("Minimum:", minimum)
print("Maximum:", maximum)
print("Average:", total / count)
print("Geometric mean:", pow(product, 1 / count))
```

More Practice

These questions are not as challenging as the extension questions usually are, but just guides for you to keep practising with.

1. Write a program that asks the user for a filename, opens that file and then simply prints how many lines are in the file.
2. Write a program that asks the user for a filename, opens that file and then simply prints whether or not the file has any contents.
    a. Adjust this to print the number of characters (and then words) in the file if you can figure that out.
3. Write a program that asks the user for a phrase and a filename, opens that file and then searches the file to find a line that is equal to the phrase entered. If it finds it, the program should print what line it was found on; otherwise it should print "Not found".
    a. Adjust this to find the phrase anywhere in the file, including as part of a line, not the whole line.

SOLUTIONS – See Python file

Quick Program Solutions

1. Write a program that asks the user for their name, then opens a file called "name.txt" and writes that name to it.
2. Write a program that opens "name.txt" and reads the name (as above) then prints,
   "Your name is Bob" (or whatever the name is in the file).
3. First, create a text file called "numbers.txt" (save it in your project).
   Put the numbers 17 and 42 on separate lines in the file and save it.
   Write a program that opens "numbers.txt", reads the numbers and adds them together then prints the result, which should be… 59.

```
# Quick Program 1
outFile = open("name.txt", "w")
name = input("What is your name? ")
outFile.write(name)
outFile.close()

# Quick Program 2
inFile = open("name.txt", "r")
name = inFile.read().strip()
print("Your name is", name)
inFile.close()

# Quick Program 3
inFile = open("numbers.txt", "r")
number1 = int(inFile.readline().strip())
number2 = int(inFile.readline().strip())
print(number1 + number2)
inFile.close()
```

```python
# Quick Program 3 extended - sum of all numbers
inFile = open("numbers.txt", "r")
total = 0
for line in inFile:
    number = int(line.strip())
    total += number
print(total)
inFile.close()
```