**Level 1: Reading a Text File**

1. Open a new Python Repl and run the following program.

fileHandle = open("myfile.txt","r")

fileContents = fileHandle.read()

print(fileContents)

fileHandle.close()

1. Why does this program produce a run-time error?

The reason why the program produces a run-time error is because there is no file with the name “myfile.txt” on line 1.

1. Add a text file to your project as follows:
   * Click on “Add File” icon in the files pane/window.
   * Type “myfile.txt” and return.
   * “myfile.txt” is now open in the editor pane/window.
   * Type some text into “myfile.txt”
   * Make sure to add several lines of text

1. Switch back to main.py pile and run the program.
   1. What gets printed out?

The multiple lines of text that were written in the myfile.txt file get printed out.

* 1. Explain the result.

The text that is contained in the “myfile.txt” file gets printed out on the screen with the “open()” command.

1. Load and run the following program.

fileHandle = open("myfile.txt","r")

line = fileHandle.readline()

count = 1

while line :

print("Line ", count, " : ",line.strip())

line = fileHandle.readline()

count += 1

fileHandle.close()

1. Compare and contrast the output of the first and second program
   1. How is the read() function similar to the readline() function?

The read() function is similar to the readline() function because it prints out the text that is contained inside the “myfile.txt” file.

* 1. How is the read() function different from the readline() function?

The read() function is different from the readline() function because it does not print the line number in front of each line of text.

1. Research the Python open() function for file I/O (input / output).
   1. How do you specify which file to open?

You can specify which file to open by entering the file name inside the brackets of the open() function using quotations.

* 1. Modify the program to open a different file.

fileHandle = open("main.py","r")

line = fileHandle.readline()

count = 1

while line :

print("Line ", count, " : ",line.strip())

line = fileHandle.readline()

count += 1

fileHandle.close()

1. Research how to open a file in a sub-directory.
   1. Modify the second program to open a file in a sub-directory.
   2. Demo your program to Mr. Nestor
   3. List your program modifications below

To open a file in a sub-directory, I first created a folder named “images” and then I added another folder with the name "Folder”. Under the “Folder” sub-directory, I created a file named “myfile.txt”. I modified the second program by changing line 1 into:

fileHandle = open("images/Folder/myfile.txt","r")

I added a slash (/) symbol to locate the file that is in the sub-directory.

fileHandle = open("images/Folder/myfile.txt","r")

line = fileHandle.readline()

count = 1

while line :

print("Line ", count, " : ",line.strip())

line = fileHandle.readline()

count += 1

fileHandle.close()

**Level 2: Writing a Text File**

1. Research the Python open() function for file I/O (input / output).
   1. What does the file mode “r” mean?

The file mode “r” is used to read the file.

* 1. What mode is used to open a file for writing?

The file mode “w” is used to open a file in writing.

* 1. What other file modes can be used? List and explain their meanings.

The files modes “x”, “a”, “t”, “b”, and “+” can be used. The “x” file mode is used to open a file for exclusive creation which means that it can only be successfully performed if the specific file name does not exist. Without truncating a specific file, the “a” file mode is used for appending at the end of the file. If there is no file under the specific file name, then the “a” file mode creates a new file. The “t” file mode is used to open a file in default text mode. The “b” file mode is used to open a file in binary mode. In order to update a specific file, the “+” file mode is used.

1. Load and run the following program.

print("Enter test to write to a file")

print("Type STOP to end the program")

print(" ")

lineNumber = 0

while True :

lineNumber += 1

userPrompt = "Enter Line " + str(lineNumber) + " : "

userText = input(userPrompt)

if userText == "STOP" :

break

print(userText)

1. Modify the program to open a text file for writing.
   1. Demo your program to Mr. Nestor
   2. List your program modifications below

fileHandle = open("myfile.txt","w")

print("Enter test to write to a file")

print("Type STOP to end the program")

print(" ")

lineNumber = 0

while True :

lineNumber += 1

userPrompt = "Enter Line " + str(lineNumber) + " : "

userText = input(userPrompt)

if userText == "STOP" :

break

print(userText)

fileHandle.write(userText)

fileHandle.close()

1. Replace the line “print(userText)” with a command to write the value of “userText” to an open file.
   1. Verify that text was written to your file
   2. Demo your program to Mr. Nestor
   3. List your program modifications below

fileHandle = open("myfile.txt","w")

print("Enter text to write to a file")

print("Type STOP to end the program")

print(" ")

lineNumber = 0

while True :

lineNumber += 1

userPrompt = "Enter Line " + str(lineNumber) + " : "

userText = input(userPrompt)

if userText == "STOP" :

break

line = fileHandle.write(userText)

fileHandle.close()

**Level 3: Binary Files**

1. Add a folder to your repl workspace and call it “images”.
2. Locate and download a “BMP” format image file and add it to your images folder.
   1. The file must be a BMP file. JPG, GIF, PNG, etc. will not work
   2. Add the image by using “drag-and-drop” onto your images folder.
   3. You can use the “Penguin.bmp” file from the GitHub Topic B folder if you want
3. Load the following program
   1. Add it to your repl
   2. Modify the “open” command to read your image file
   3. Run the program and examine the data output.

"""

Function to convert 4 bytes (1 word) into a decimal integer

"""

def convertWordToInteger(dataWord) :

result = int(dataWord[3])

result += 256 \* int(dataWord[2])

result += 512 \* int(dataWord[1])

result += 1024 \*int(dataWord[0])

return result

"""

Function to display raw file data

Each data byte is displayed in row order

"""

def dumpRawData(rawData) :

idx = 0

for row in range(8) :

rowText = " ";

for col in range(8) :

rowText += str(rawData[idx]).zfill(3) + " "

idx += 1

print(rowText)

"""

Main program code begins here

- Start with opening and reading the data file

"""

handle = open("Penguin.bmp", "rb")

rawData = handle.read(64)

handle.close()

"""

Print out the RAW data contained at the start of the file

- This is the Header Information

- A BPM (Bitmap) Image has a well defined Header

- Each grouping of bytes has a specific meaning

"""

print(" ")

print("RAW Image Header Data (64 bytes)")

dumpRawData(rawData)

print(" ")

"""

According to the BMP specification the first two bytes

have the value "BM".

"""

print("First Two Bytes")

print(str(rawData[0]).zfill(3), str(rawData[1]).zfill(3))

print(" ")

"""

According to the BMP specification the image Width

is contained in the 4 bytes (1 word) biginning at

position 18

"""

print("Image Width Data")

dataText = str(rawData[18]).zfill(3) + " "

dataText += str(rawData[19]).zfill(3) + " "

dataText += str(rawData[20]).zfill(3) + " "

dataText += str(rawData[21]).zfill(3)

print("Image Width: (raw)", dataText)

dataText = str(rawData[21]).zfill(3) + " "

dataText += str(rawData[20]).zfill(3) + " "

dataText += str(rawData[19]).zfill(3) + " "

dataText += str(rawData[18]).zfill(3)

print("Image Width: (re-ordered)", dataText)

dataWord = [rawData[21],rawData[20],rawData[19],rawData[18]]

print("Image Width: (pixels)", convertWordToInteger(dataWord))

print(" ")

"""

According to the BMP specification the image Height

is contained in the 4 bytes (1 word) biginning at

position 22

"""

print("Image Height Data")

dataText = str(rawData[22]).zfill(3) + " "

dataText += str(rawData[23]).zfill(3) + " "

dataText += str(rawData[24]).zfill(3) + " "

dataText += str(rawData[25]).zfill(3)

print("Image Width: (raw)", dataText)

dataText = str(rawData[25]).zfill(3) + " "

dataText += str(rawData[24]).zfill(3) + " "

dataText += str(rawData[23]).zfill(3) + " "

dataText += str(rawData[22]).zfill(3)

print("Image Width: (re-ordered)", dataText)

dataWord = [rawData[25],rawData[24],rawData[23],rawData[22]]

print("Image Width: (pixels)", convertWordToInteger(dataWord))

print(" ")

"""

END OF PROGRAM

"""

1. Decode the meaning of the first two bytes of data in the header data of a BMP file.
   1. What are the values of the first two bytes?

The values of the first two bytes are 066 and 077.

* 1. Look up the values in an ASCII character table. Google “ASCII Character Table” or Download the ASCII Conversion Chart from the GitHub Topic B folder.
  2. What ASCII characters do these two bytes represent?

The ASCII characters that these two bytes represent are “BM”.

1. Open and examine the BMP file format specification for the “Signature” data field
   1. Open the URL listed below to access the document
   2. According to the document, the first two bytes of data are the “Signature”
   3. What is the description of the “Signature” in a BMP file?

The description of the “Signature” in a BMP file is ‘BM’ which is an indication that the given file is a bitmap type.

* 1. How does this compare to your answer to question #4 above?

This compares to my answer to question #4 above because the first two bytes in the image header are “BM” and it is also the description of the signature of the file type.

<http://www.ece.ualberta.ca/~elliott/ee552/studentAppNotes/2003_w/misc/bmp_file_format/bmp_file_format.htm>

1. Examine the BMP file format specification for the Width data field
   1. Locate the “Width” data field in the BMP specification document.
   2. What is the size, in bytes, of this field?

The size of the “Width” data field is 4 bytes.

* 1. What is the value, in bytes, of this field for your image file? (Look at the program output)

The value of this field for my image file is 031 001 0 bytes.

* 1. What is the value, in decimal, of this field for your image file? (Look at the program output)

The value, in decimal, of this field for my image file is 287.

1. Examine the BMP file format specification for the Height data field
   1. Locate the “Height” data field in the BMP specification document.
   2. What is the size, in bytes, of this field?

The size of this field is 4 bytes.

* 1. What is the value, in bytes, of this field for your image file? (Look at the program output)

The value of this field for my image file is 045 001 0 bytes.

* 1. What is the value, in decimal, of this field for your image file? (Look at the program output)

The value, in decimal, of this field for my image file is 301.

1. Open your BMP image file in an application program like Paint or Photoshop.
   1. What is the size of your image file?

The size of my image file is 287 pixels in width and 301 pixels in height.

* 1. How does this compare to the output of the program?

This is the same size as shown in the output of the program.