**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?

This produces a runtime error because the command aren’t given properly

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

Hello!

How are you?

Whats' the weather like?

Do you like cats or Dogs?

What do you pefer ice cream or a popsicle?

Is programming hard?

What time is it?

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

The several lines of text from file “myfile.txt” gets printed out because the command on the first page is asking python to open “myfile.txt” and print out everything on that page and then close it when it’s done.

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()

6. Compare and contrast the output of the first and second program

a.    How is the read() function similar to the readline() function?

They are similar because they both process lines of text that are on “myfile.txt” and print them out.

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

The read() function is different because it reads the entire file while the readline() functions reads only one line in the file.

7.    Research the Python open() function for file I/O (input / output).

a.    How do you specify which file to open?

You can specify which file for it to open by adding the string value ‘r’ as a second argument for python to open(). This specifies which file you want open and python will run it.

b.    Modify the program to open a different file.

fileHandle = open("wordfile.txt","r")

fileContents = fileHandle.read()

print(fileContents)

fileHandle.close()

8. Research how to open a file in a sub-directory.

* 1. Modify the second program to open a file in a sub-directory.

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

line = fileHandle.readline()

count = 1

while line :

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

line = fileHandle.readline()

count += 1

fileHandle.close()

* 1. Demo your program to Mr. Nestor
  2. List your program modifications below

I created a new folder and then added the folder’s name to fileHandle = open("myfile.txt","r"). The line looked like this after changing it fileHandle = open("COmputer/myfile.txt","r").

**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?

Open for reading and writing. The stream **is** positioned at the beginning of the file

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

R+

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

|  |
| --- |
| **rb**  Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode. |
| **r+**  Opens a file for both reading and writing. The file pointer placed at the beginning of the file. |
| **rb+**  Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file. |
| **w**  Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| **wb**  Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing. |
| **w+**  Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| **wb+**  Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing. |
| **a**  Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| **ab**  Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing. |
| **a+**  Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |
| **ab+**  Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing. |

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)

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

print("Type STOP to end the program")

lineNumber = 0

while True :

lineNumber += 1

userPrompt = "Whats our favorite colour? " + str(lineNumber) + " : "

userText = input(userPrompt)

if userText == "STOP" :

break

fileHandle.write(userText+'\n')

fileHandle.close()

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

file=open("userText","w")

print("test")

print("STOP")

print(" ")

lineNumber = 0

while True :

lineNumber += 1

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

userText = input(userPrompt)

if userText == "STOP" :

break

open("userText")

I replaced the first line with fileHandle = open("userText","w") so I can run the program and type in myfile.txt. I also modified the last line to open up what the user typed in.

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

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

print("Type STOP to end the program")

print(" ")

lineNumber = 0

while True :

lineNumber += 1

userPrompt ="What's your favourite movie?Why?" + str(lineNumber) + " : "

userText = input(userPrompt)

if userText == "STOP" :

break

fileHandle.write(userText+'\n')

fileHandle.close()

* 1. List your program modifications below  
     I changed the first line to open a file for writing and asked the user a question. I also told the program to save whatever the user types into that file and close it.

**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

"""

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("images/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. 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 byte 066 represents the number 6 and the byte 077 represents a ?.

* 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 byte 066 represents the number 6 and the byte 077 represents a ?.

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 shows the first 2 bytes of the file

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

This compares to question #4 because it tells what are the first 2 bytes of data in the file and is usually called the Signature of the file.

<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 is 4 bytes.

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

The size is 031 001 000 000.

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

The value in decimal is 000 000 001 031.

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 size is 045 001 000 000.

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

The size is 000 000 001 045.

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 the width is 287 and height is 301.

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

If you compare both outputs together they both have the same width and height.