# 

# Experiment No 3:

**Aim: To implement File Handling in Python. Theory:**

The key function for working with files in Python is the open() function. The open() function takes two parameters; *filename*, and *mode*.

There are four different methods (modes) for opening a file:

" r" - Read - Default value. Opens a file for reading, error if the file does not exist "a" - Append - Opens a file for appending, creates the file if it does not exist "w" - Write - Opens a file for writing, creates the file if it does not exist "x"

- Create - Creates the specified file, returns an error if the file exists In addition you can specify if the file should be handled as binary or text mode "t" - Text -

Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

# Python has a set of methods available for the file object.

**Method Description**

close() Closes the file

detach() Returns the separated raw stream from the buffer

fileno() Returns a number that represents the stream, from the operating system's perspective flush() Flushes the internal buffer

isatty()Returns whether the file stream is interactive or not read() Returns the file content

readable() Returns whether the file stream can be read or not readline() Returns one line from the file

readlines() Returns a list of lines from the file



seek() Change the file position

seekable() Returns whether the file allows us to change the file position tell() Returns the current file position

truncate() Resizes the file to a specified size

writable() Returns whether the file can be written to or not write() Writes the specified string to the file

writelines() Writes a list of strings to the file

# PROGRAM:

**Program 3.1: Python program to copy odd noline from one file to other**

# # open file in read mode

# fn = open('myfile.txt', 'r')

# # open other file in write mode

# fn1 = open('myfile.txt', 'w')

# # read the content of the file line by line

# cont = fn.readlines()

# print(len(cont)) # Print the number of lines in the file

# print(type(cont)) # Print the type of cont variable

# # Loop through each line in the file

# for i in range(0, len(cont)):

# # Check if the line number is odd

# if i % 2 != 0:

# # Write the line to the new file

# fn1.write(cont[i])

# else:

# pass

# 

# #

# close the file

# fn1.close()

# # open file in read mode

# fn1 = open('myfile.txt', 'r')

# # read the content of the file

# cont1 = fn1.read()

# # print the content of the file

# print(cont1)

# # close all files

# fn.close()

# fn1.close()

# OUTPUT:

# 

# Program 3.2:

# Function to count number

# of characters, words, spaces, and lines in a file

def counter(fname):

# Variables to store total counts

num\_words = 0

num\_lines = 0

num\_charc = 0

num\_spaces = 0



with open(fname, 'r') as f:

# Loop to iterate file line by line

for line in f:

# Incrementing total line count

num\_lines += 1

# Flag to track word presence in the line

word = 'Y'

# Loop to iterate every character in the line

for letter in line:

# Condition to check if the character is not a white space and a word

if letter != ' ' and word == 'Y':

# Incrementing the word count

num\_words += 1

word = 'N'

# Condition to check if the character is a white space

elif letter == ' ':

# Incrementing the space count

num\_spaces += 1

word = 'Y'

# Incrementing character count for every character except space and newline

if letter != " " and letter != "\n":

num\_charc += 1

# Printing total counts

print("Number of words in text file:", num\_words)

print("Number of lines in text file:", num\_lines)

print('Number of characters in text file:', num\_charc)

print('Number of spaces in text file:', num\_spaces)

Driver Code

if \_\_name\_\_ == '\_\_main\_\_':

fname = 'myfile.txt'

try:

counter(fname)

except FileNotFoundError:

print('File not found')



# OUTPUT

# 

**Conclusion:**

The experiment successfully demonstrated the implementation of File Handling in Python, showcasing its versatility in reading, writing, and manipulating various file formats.