Experiment No.3
To implement File Handling in Python.
Date of Performance:
Date of Submission:

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('bcd.txt', 'r')

# open other file in write mode
fn1 = open('nfile.txt', 'w')

# read the content of the file line by line
cont = fn.readlines()
print(len(cont))
type(cont)
for i in range(0, len(cont)):
```

```
 if(i % 2 != 0):
 fn1.write(cont[i])
 else:
 pass

# close the file
fn1.close()

# open file in read mode
fn1 = open('nfile.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()
```

bcd.txt

hello how are you Line1

I am fine line2

Python Programming Line 3

Numpy

Pandas line5

OUTPUT:

I am fine line2

Numpy

Program 3.2:

```
# Python implementation to
compute # number of
characters, words, spaces #
and lines in a file



# Function to count number

# of characters, words, spaces

# and lines in a file

def counter(fname):

 # variable to store total word
count  num_words = 0

 # variable to store total line
count  num_lines = 0

 # variable to store total character
count  num_charc = 0

 # variable to store total space
count  num_spaces = 0



 # opening file using with()
method  # so that file gets
closed

 # after completion of work
```

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


# loop to iterate file

# line by line

for line in f:









# incrementing value of

# num_lines with each

# iteration of loop to

# store total line count

num_lines += 1


# declaring a variable word
# and assigning its value as Y
# because every file is

# supposed to start with

# a word or a character

word = 'Y'


# loop to iterate every

# line letter by letter

for letter in line:
```

```
 # condition to check

 # that the encountered character  #
is not white space and a word   if
(letter != ' ' and word == 'Y'):

 # incrementing the word  #
count by 1

 num_words += 1


 # assigning value N to   # variable
word because until  # space will not
encounter  # a word can not be
completed  word = 'N'


 # condition to check

 # that the encountered character
# is a white space

 elif (letter == ' '):


 # incrementing the space  #
count by 1

 num_spaces += 1


 # assigning value Y to  # variable
word because after  # white space
a word
```

```
# is supposed to occur
 word = 'Y'


 # loop to iterate every
 # letter character by
 # character
 for i in letter:


 # condition to check
 # that the encountered character   # is
not white space and not
 # a newline character
 if(i !=" " and i !="\n"):


 # incrementing character
 # count by 1
 num_charc += 1
 # printing total word count
 print("Number of words in text file: ",
num_words)
 # printing total line count
 print("Number of lines in text file: ",
num_lines)  # printing total character count
```

```
 print('Number of characters in text file: ',
num_charc)  # printing total space count

 print('Number of spaces in text file: ',
num_spaces)  # Driver Code:

#if __name__ == '__main__':

fname = 'File1.txt'

try:

 counter(fname)

except:

 print('File not found')
```

File1.txt

Hello how are you

what is todays date

1111111111111111111

222222222222222222
3333333333333333
4444444444444444444444

55555555555555555555555555

OUTPUT

Number of words in text
file: 13 Number of lines
in text file: 6

Number of characters in text
file: 135 Number of spaces in
text file: 7

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