**Experiment No. 8**

**Title: File Handling in Python.**

**Batch: B1 Roll No: 1914078 Experiment No.:8**

**Aim:** To introduce File Handling in Python .

**Resources needed:** Python IDE

### Theory:

### A file is a contiguous set of bytes used to store data. Python supports 2 types of files:

### ASCII Text File: A stream of characters that can be sequentially processed by computer in forward direction. It is human readable. Examples: Python source code, HTML file, text file, markdown file etc.

### Binary File: A file which can contain any type of data encoded in binary for computer storage and processing purposes. It is a character stream with collection of bytes. Examples: executable files, images, audio etc

### Open()

### In Python, a physical file must be mapped to a built-in file object with the help of built-in function open(). Open() function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

### Syntax: file object = open(file name[, access mode][, buffersize])

### >>> f = open("test.txt",'w') # write in text mode

### >>> f = open("test.txt") # open file in current directory

### >>> f = open("C:/Python33/README.txt") # specifying full path

### The default is reading in text mode. In this mode, we get strings when reading from the file.

### Python File Access Modes

|  |  |
| --- | --- |
| Code | Description |
| 'r' | Open a file for reading. (default) |
| 'w' | Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists. |
| 'x' | Open a file for exclusive creation. If the file already exists, the operation fails. |
| 'a' | Open for appending at the end of the file without truncating it. Creates a new file if it does not exist. |
| 't' | Open in text mode. (default) |
| 'b' | Open in binary mode. |
| '+' | Open a file for updating (reading and writing) |

### Close()

### Opening files create a stream of the data buffer. Closing a file will free up the resources that were tied with the file ie close this stream and is done using Python close() method.

### f = open("test.txt",encoding = 'utf-8') # perform file operations

### f.close()

### Write()

### To write into a file in Python, we need to open it in write 'w', append 'a' or exclusive creation 'x' mode. The 'w' mode will overwrite into the file if it already exists. All previous data are erased. This method returns the number of characters written to the file.

### #Writing to a File

### f=open("D:\myfile.txt","w")

### f.write("Learn Python.")

### f.close()

### Python File Methods

|  |  |
| --- | --- |
| Method | Description |
| close() | Close an open file. It has no effect if the file is already closed. |
| detach() | Separate the underlying binary buffer from the TextIOBase and return it. |
| fileno() | Return an integer number (file descriptor) of the file. |
| flush() | Flush the write buffer of the file stream. |
| isatty() | Return True if the file stream is interactive. |
| read(n) | Read atmost n characters form the file. Reads till end of file if it is negative or None. |
| readable() | Returns True if the file stream can be read from. |
| readline(n=-1) | Read and return one line from the file. Reads in at most n bytes if specified. |
| readlines(n=-1) | Read and return a list of lines from the file. Reads in at most n bytes/characters if specified. |
| seek(offset,from=SEEK\_SET) | Change the file position to offset bytes, in reference to from (start, current, end). |
| seekable() | Returns True if the file stream supports random access. |
| tell() | Returns the current file location. |
| write(s) | Write string s to the file and return the number of characters written. |
| writelines(lines) | Write a list of lines to the file. |

### With

### Syntax: *f.seek(offset, from)*

### Here, the *from* parameter takes the following values:

### 0 : offset calculated from the beginning

### 1 : offset calculated from the current position

### 2 : offset calculated from the end

### f=open("D:\myfile.txt","r+")

### f.seek(6,0)

### lines=f.readlines()

### for line in lines:

### print(line)

### f.close()

### World

### myfile.txt contains "Hello World" text

### tell(): returns our current position (in number of bytes).

### >>> f.tell() # get the current file position

### Activity:

### Write a Python to copy the contents of a text file in the reverse the of original text file and also compute the number of characters, words, spaces and lines in a file.

f = open("./exp.txt", "w")

f.write("""

        It's been a long day without you, my friend

        And I'll tell you all about it when I see you again

        We've come a long way from where we began

        Oh, I'll tell you all about it when I see you again

        When I see you again

    """)

f.close()

spaces = 0

chars = 0

words = 0

f = open("./exp.txt")

f2 = open("./expOut.txt", "a+")

lines = f.readlines()

print("Number of lines :", len(lines))

for line in reversed(lines):

    words += len(line.split())

    for i in line:

        if (i == " "):

            spaces += 1

        else:

            chars += 1

    f2.write(line)

print("Number of words :", words)

print("Number of characters :", chars)

print("Number of spaces :", spaces)

f.close()

f2.close()

### Output:

### 

### And expOut.txt :-

### 

### Outcomes: Apply different File Handling Operations in Python

**Conclusion:** We were introduced to File Handling in Python .

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of faculty in-charge with date**

**References:**

**Books:**

1. Reema Thareja , *Python Programing: Using Problem Solving Approach,* Oxford University Press, First Edition 2017, India
2. Dr. R. Nageswara Rao, *Core Python Programming,* Wiley Publication, Second Edition 2018,India
3. Sheetal Taneja and Naveen Kumar, *Python Programing: A Modular Approach,* Pearson India
4. Swarroop C.H, *Byte of python,* e-book, Kindle edition
5. Martin C Brown, *The Complete Reference Python,* Brandon A Nordin, First Edition 2001