**FILE HANDLING IN PYTHON**

TASK (07/11/2019)

**Content:**

* + Opening a File in Python …………..…………….2

# Closing a File in Python ……………….………….3

# Reading and Writing Files …………………………...3

## Appending text to a File …………………...….… ….4

## Cursor Position in File …………………………….... 5

## Renaming and Deleting Files………………………..6

Opening a File in Python:

* Python provides basic functions and methods necessary to manipulate files by default. You can do most of the file manipulation using a **file** object.
* Before you can read or write a file, you have to open it using Python's built-in *open()* function. This function creates a **file** object, which would be utilized to call other support methods associated with it.
* Here for this open() function, we should pass different parameters like file\_name, file\_mode, buffer.
* We have different types of File\_modes, they are:

1. "r" - Read - Default value. Opens a file for reading, error if the file does not exist
2. "a" - Append - Opens a file for appending, creates the file if it does not exist
3. "w" - Write - Opens a file for writing, creates the file if it does not exist
4. "x" - Create - Creates the specified file, returns an error if the file exists
5. "t" - Text - Default value. Text mode
6. "b" - Binary - Binary mode (e.g. images)
7. “+” This will open a file for reading and writing (updating)

Example:

Code:

f = open("ex.txt", "wt") #we are opening the file in write and text mode

print("Name of the file: ", f.name)

print("Opening mode : ", f.mode)

Execution:

Name of the file:  eg.txt

Opening mode :  wt

Closing a File in Python:

Example:

Code:

f= open("eg.txt", "wt")

print("Name of the file: ", f.name)

print("Opening mode : ", f.mode)

f.close()

print("Closed or not : ", f.closed)

Execution:

Name of the file:  eg.txt

Opening mode :  wt

Closed or not :  True

Reading and Writing Files:

The *file* object provides a set of access methods to make our lives easier. We would see how to use *read()* and *write()* methods to read and write files.

Example 1:

Code:

f= open("eg.txt", "w")

r=f.write("hello world")#writing into file

if r!=0:

print("the file has edited successful")

f.close()

Execution:

the file has edited successful

Example 2:

Code:

f= open("eg.txt", "r")

str=f.read(5)#reading text from file

print(str)

f.close()

Execution:

Hello

Appending text to a File:

We can also append a new text to the already existing file or the new file.

Example:

Code:

f= open("eg.txt", "a")#opening in append mode

r=f.write("\n Welcome to python ")

if r!=0:

print("the text has appended successful")

f.close()

Execution:

the text has appended successful

Cursor Position in File:

* The *tell()* method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.
* The *seek(offset[, from])* method changes the current file position. The *offset* argument indicates the number of bytes to be moved. The *from* argument specifies the reference position from where the bytes are to be moved.

Example:

Code:

f= open("eg.txt", "r")

print(f.read(5))

pos= f.tell() # Tells the cursor position

print("Current file position : ", pos)

print("After using seek() function")

pos = f.seek(0, 0) #Changes the cursor position to zeroth position

pos= f.tell()

print("Current file position : ", pos)

f.close()

Execution:

hello

Current file position :  5

After using seek() function

Current file position :  0

Renaming and Deleting Files:

* Python **os** module provides methods that help you perform file-processing operations, such as renaming and deleting files.

Example:

Code:

import os

r=os.rename( "eg.txt", "ex.txt" )

if r!=0:

print("successfully changed")

s=os.remove("ex.txt")

if s!=0:

print("successfully deleted")

Execution:

successfully changed

successfully deleted