**Open a File in Python**

## Access Modes for Opening a file

The access mode parameter in the open() function primarily mentions **the purpose of opening the file** or the type of operation we are planning to do with the file after opening. in Python, the following are the different characters that we use for mentioning the file opening modes.

| **File Mode** | **Meaning** |
| --- | --- |
| **r** | Opens a file for reading (default) |
| **w** | Open a file for writing. If a file already exists, it deletes all the existing contents and adds new content from the start of the file. |
| **x** | Open a file for exclusive creation. If the file already exists, this operation fails. |
| **a** | Open a file in the append mode and add new content at the end of the file. |
| **b** | Open the file in binary mode. |
| **t** | Opens a file in a text mode (default). |
| **+** | Open a file for updating (reading and writing). |

Examples:

# Opening the file with absolute path

fp = open(r'E:\demos\files\sample.txt', 'r')

# read file

print(fp.read())

# Closing the file after reading

fp.close()

# path if you using MacOs

# fp = open(r"/Users/myfiles/sample.txt", "r")

# Opening the file with relative path

try:

    fp = open("sample.txt", "r")

    print(fp.read())

    fp.close()

except FileNotFoundError:

    print("Please check the path.")

# Open and Append at last

fp = open("sample2.txt", "a")

fp.write(" Added this line by opening the file in append mode ")

# Opening the file again to read

fp = open("sample2.txt", "r")

print(fp.read())

fp.close()

# Opening a File for multiple operations

# In Python, we can open a file for performing multiple operations simultaneously by using the '+' operator. When we pass r+ mode then it will enable both reading and writing options in the file.

with open("Sample3.txt", "r+") as fp:

    # reading the contents before writing

    print(fp.read())

    # Writing new content to this file

    fp.write("\nAdding this new content")

fp.close()

# Opening a Binary file

with open("sample.bin", "rb") as f:

    byte\_content = f.read(1)

    while byte\_content:

        # Printing the contents of the file

        print(byte\_content)

# Read File in Python

## Access Modes for Reading a file

To read the contents of a file, we have to open a file in reading mode. Open a file using the built-in function called open(). In addition to the file name, we need to pass the file mode specifying the **purpose of opening the file**.

The following are the different modes for reading the file. We will see each one by one.

| **File Mode** | **Definition** |
| --- | --- |
| **r** | The default mode for opening a file to read the contents of a text file. |
| **r+** | Open a file for both reading and writing. The file pointer will be placed at the beginning of the file. |
| **rb** | Opens the file for reading a file in binary format. The file pointer will be placed at the beginning of the file. |
| **w+** | Opens a file for both writing as well as reading. The file pointer will be placed in the beginning of the file. For an existing file, the content will be overwritten. |
| **a+** | Open the file for both the reading and appending. The pointer will be placed at the end of the file and new content will be written after the existing content. |

# read file with absolute path

try:

    fp = open(r"E:\demos\files\read\_demo.txt", "r")

    print(fp.read())

    fp.close()

except FileNotFoundError:

    print("Please check the path")

# Reading files using 'with'

with open('read\_demo.txt', 'r') as file:

    print(file.read())

## File Read Methods

Python provides three different methods to read the file. We don’t have to import any module for that.. Below are the three methods

| **Method** | **When to Use?** |
| --- | --- |
| **read()** | Returns the entire file content and it accepts the optional size parameter that mentions the bytes to read from the file. |
| **readline()** | The readline() method reads a single line from a file at a time. . Accepts optional size parameter that mentions how many bytes to return from the file. |
| **readlines()** | The readlines() method returns a list of lines from the file. |

Examples:

# readline(): Read a File Line by Line

with open('read\_demo.txt', 'r') as fp:

    # read first line

    # assign it to string variable

    line = fp.readline()

    print(line)

# Reading First N lines From a File Using readline()

with open('read\_demo.txt', 'r') as file:

    # read first 3 lines

    for i in range(3):

        print(file.readline().strip())

# Reading Entire File Using readline()

with open('read\_demo.txt', 'r') as fp:

    # Read the first line

    line = fp.readline()

    # Iterate the file till it reached the EOF

    while line != '':

        print(line, end='')

        line = fp.readline()

# Reading First and the Last line using readline()

with open("read\_demo.txt", "r") as file:

    # reading the first line

    first\_line = file.readline()

    print(first\_line)

    for last\_line in file:

        pass

    # printing the last line

    print(last\_line)

# Reading File into List

with open('read\_demo.txt', 'r') as fp:

    # Read file into list

    lines = fp.readlines()

    print(lines)

# Reading first N lines from a file

N = 2

with open("readdemo.txt","r") as file:

    head = [next(file) for x in range(N)]

print(head)

# Reading the last N lines in a file

n = 2

with open('readdemo.txt', 'r') as f:

    lines = f.readlines()[n:]

print(lines)

# Reading and Writing to the same file

with open('readdemo.txt', 'r') as f:

  print(f.read())

  f.write("Reading fresh")

# Reading File in Reverse Order

with open('readdemo.txt', 'r') as f:

  lines = f.readlines()

  for line in reversed(lines):

    print(line)

File seek() method:

1. The seek() method to move the file cursor ahead or backward from the current position.
2. The seek() function sets the position of a file pointer and the tell() function returns the current position of a file pointer.

f.seek(offset, whence)

How many points the pointer will move is **computed from adding offset to a reference point**; the reference point is given by the whence argument.

The allowed values for the whence argument are: –

* A whence value of **0** means from the beginning of the file.
* A whence value of **1** uses the current file position
* A whence value of **2** uses the end of the file as the reference point.

The default value for the whence is the beginning of the file, which is **0**

Refer to the below**table for clear understanding**.

| **Seek Operation** | **Meaning** |
| --- | --- |
| **f.seek(0)** | Move file pointer to the beginning of a File |
| **f.seek(5)** | Move file pointer five characters ahead from the beginning of a file. |
| **f.seek(0, 2)** | Move file pointer to the end of a File |
| **f.seek(5, 1)** | Move file pointer five characters ahead from the current position. |
| **f.seek(-5, 1)** | Move file pointer five characters behind from the current position. |
| **f.seek(-5, 2)** | Move file pointer in the reverse direction. Move it to the 5th character from the end of the file |

Example 1:

# reading the file directly from the 6th character.

with open(r'D:\test\_code\happy.txt', "r") as fp:

    # Moving the file handle to 6th character

    fp.seek(6)

    # read file

    print(fp.read())

Example 2:

# open file in write and read mode w+

with open(r'D:\test\_code\happy.txt', "w+") as fp:

    # add content

    fp.write('My First Line\n')

    fp.write('My Second Line')

    # move pointer to the beginning

    fp.seek(0)

    # read file

    print(fp.read())

Set whence to **2** and the **offset to 0** to move the file pointer to the end of the file.

* In the below example, we will perform the following three operations
* We will move the file pointer at the end of the file and write new content
* Next, we will move the file pointer at the start of the file and write fresh content at the beginning of the file.
* Again, we will move the file pointer to the end of the file and write more content.

Example 3:

# open file for reading and writing  a+

with open(r'D:\test\_code\happy.txt', "r+") as fp:

    # Moving the file handle to the end of the file

    fp.seek(0, 2)

    # Inserting new content to the end of the file

    fp.write("\nThis content is added to the end of the file")

    # moving to the beginning

    # again read the whole file

    fp.seek(0)

    print(fp.read())

### Renaming a file in Python

Example 4:

import os

# Absolute path of a file

old\_name = r"D:\test\_code\happy.txt"

new\_name = r"D:\test\_code\new\_details.txt"

# Renaming the file

os.rename(old\_name, new\_name)

Example 5:

import os

old\_name = r"E:\demos\files\reports\details.txt"

new\_name = r"E:\demos\files\reports\new\_details.txt"

# enclosing inside try-except

try:

    os.rename(old\_name, new\_name)

except FileExistsError:

    print("File already Exists")

    print("Removing existing file")

    # skip the below code

    # if you don't' want to forcefully rename

    os.remove(new\_name)

    # rename it

    os.rename(old\_name, new\_name)

    print('Done renaming a file')

Example 6:

# Rename Multiple Files in Python

import os

folder = r'E:\demos\files\reports\\'

count = 1

# count increase by 1 in each iteration

# iterate all files from a directory

for file\_name in os.listdir(folder):

    # Construct old file name

    source = folder + file\_name

    # Adding the count to the new file name and extension

    destination = folder + "sales\_" + str(count) + ".txt"

    # Renaming the file

    os.rename(source, destination)

    count += 1

print('All Files Renamed')

print('New Names are')

# verify the result

res = os.listdir(folder)

print(res)

Example 7:

# Renaming files with a timestamp

import os

from datetime import datetime

# adding date-time to the file name

current\_timestamp = datetime.today().strftime('%d-%b-%Y')

old\_name = r"E:\demos\files\reports\sales.txt"

new\_name = r"E:\demos\files\reports\sales" + current\_timestamp + ".txt"

os.rename(old\_name, new\_name)

# Delete (Remove) Files and Directories in Python

# Example 1:

import os

# remove file with absolute path

os.remove(r"E:\demos\files\sales\_2.txt")

# Example 2:

import os

file\_path = r'E:\demos\files\sales\_2.txt'

if os.path.exists(file\_path):

    os.remove(file\_path)

else:

    print("The system cannot find the file specified")

# Example 3:

# Delete all Files from a Directory

import os

path = r"E:\demos\files\reports\\"

for file\_name in os.listdir(path):

    # construct full file path

    file = path + file\_name

    if os.path.isfile(file):

        print('Deleting file:', file)

        os.remove(file)

Example 4:

# Delete an Empty Directory (Folder) using rmdir()

import os

# Deleting an empty folder

directory = r"E:\santosh\old\_logs"

os.rmdir(directory)

print("Deleted '%s' directory successfully" % directory)

Example 5:

import pathlib

# Deleting an empty folder

empty\_dir = r"D:\santosh\old\_images"

path = pathlib.Path(empty\_dir)

path.rmdir()

print("Deleted '%s' directory successfully" % empty\_dir)

Example 6:

# Delete a file or a Non-Empty Directory using shutil module

import os

import shutil

def delete(path):

    """path could either be relative or absolute. """

    # check if file or directory exists

    if os.path.isfile(path) or os.path.islink(path):

        # remove file

        os.remove(path)

    elif os.path.isdir(path):

        # remove directory and all its content

        shutil.rmtree(path)

    else:

        raise ValueError("Path {} is not a file or dir.".format(path))

# file

delete(r'E:\demos\files\reports\profits.txt')

# directory

delete(r'E:\demos\files\reports')

Example 1:

# Copy Single File

import shutil

src\_path = r"E:\demos\files\report\profit.txt"

dst\_path = r"E:\demos\files\account\profit.txt"

shutil.copy(src\_path, dst\_path)

print('Copied')

Example 2:

# Copy All Files From A Directory

import os

import shutil

source\_folder = r"E:\demos\files\reports\\"

destination\_folder = r"E:\demos\files\account\\"

# fetch all files

for file\_name in os.listdir(source\_folder):

    # construct full file path

    source = source\_folder + file\_name

    destination = destination\_folder + file\_name

    # copy only files

    if os.path.isfile(source):

        shutil.copy(source, destination)

        print('copied', file\_name)

Example 3:

# Copy Entire Directory

import shutil

source\_dir = r"E:\demos\files\reports"

destination\_dir = r"E:\demos\files\account"

shutil.copytree(source\_dir, destination\_dir)

Example 4:

# Move a Single File

import shutil

# absolute path

src\_path = r"E:\santosh\reports\sales.txt"

dst\_path = r"E:\santosh\account\sales.txt"

shutil.move(src\_path, dst\_path)

Example 5:

# Move All Files From A Directory

import os

import shutil

source\_folder = r"E:\santosh\reports\\"

destination\_folder = r"E:\santosh\account\\"

# fetch all files

for file\_name in os.listdir(source\_folder):

    # construct full file path

    source = source\_folder + file\_name

    destination = destination\_folder + file\_name

    # move only files

    if os.path.isfile(source):

        shutil.move(source, destination)

        print('Moved:', file\_name)

Example 6:

# Move Multiple Files

import shutil

source\_folder = r"E:\santosh\reports\\"

destination\_folder = r"E:\santosh\account\\"

files\_to\_move = ['profit.csv', 'revenue.csv']

# iterate files

for file in files\_to\_move:

    # construct full file path

    source = source\_folder + file

    destination = destination\_folder + file

    # move file

    shutil.move(source, destination)

    print('Moved:', file)

# **Extract text from PDF File using Python**

# pip install PyPDF2

# importing required modules

import PyPDF2

# creating a pdf file object

pdfFileObj = open(r'D:\Python Notes FM\6 - Lists.pdf', 'rb')

# creating a pdf reader object

pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

# printing number of pages in pdf file

print(pdfReader.numPages)

# creating a page object

pageObj = pdfReader.getPage(0)

# extracting text from page

print(pageObj.extractText())

# closing the pdf file object

pdfFileObj.close()

# **Reading an excel file using Python**

# pip install xlrd

# Reading an excel file using Python

import xlrd

# Give the location of the file

loc = ("path of file")

# To open Workbook

wb = xlrd.open\_workbook(loc)

sheet = wb.sheet\_by\_index(0)

# For row 0 and column 0

print(sheet.cell\_value(0, 0))

# **Process Word Document**

# pip install docx

import docx

def readtxt(filename):

    doc = docx.Document(filename)

    fullText = []

    for para in doc.paragraphs:

        fullText.append(para.text)

    return '\n'.join(fullText)

print (readtxt('path\data.docx'))