Python File Open

File handling is an important part of any web application / batch applications. Python has several functions for creating, reading, updating, and deleting files.

## **File Handling**

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

To open a file for reading it is enough to specify the name of the file with complete path from root directory:

f = open("demofile.txt")

The code above is the same as:

f = open("demofile.txt", "rt")

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

Because "r" for read, and "t" for text are the default values, you do not need to specify them. 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)

To open the file, use the built-in open() function.

The open() function returns a file object, which has a read() method for reading the content of the file:

f = open("demofile.txt", "r")  
print(f.read())

By default the read() method returns the whole text, but you can also specify how many characters you want to return:

Return the 5 first characters of the file:

f = open("demofile.txt", "r")  
print(f.read(**5**))

You can return one line by using the readline() method:

f = open("demofile.txt", "r")  
print(f.readline())

By calling readline() two times, you can read the two first lines:

f = open("demofile.txt", "r")  
print(f.readline())  
print(f.readline())

Loop through the file line by line:

f = open("demofile.txt", "r")  
for x in f:  
  print(x)

It is a good practice to always close the file when you are done with it.

f = open("demofile.txt", "r")  
print(f.readline())  
f.close()

You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

## **Write to an Existing File**

To write to an existing file, you must add a parameter to the open() function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

f = open("demofile2.txt", "a")  
f.write("Now the file has more content!")  
f.close()  
  
#open and read the file after the appending:  
f = open("demofile2.txt", "r")  
print(f.read())

Open the file "demofile3.txt" and overwrite the content:

f = open("demofile3.txt", "w")  
f.write("Woops! I have deleted the content!")  
f.close()  
  
#open and read the file after the appending:  
f = open("demofile3.txt", "r")  
print(f.read())

the "w" method will overwrite the entire file.

## **Create a New File**

To create a new file in Python, use the open() method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

Create a file called "myfile.txt":

f = open("myfile.txt", "x")

Result: a new empty file is created!

Create a new file if it does not exist:

f = open("myfile.txt", "w")

## **Delete a File**

To delete a file, you must import the OS module, and run its os.remove() function:

import os  
os.remove("demofile.txt")

## **Check if File exist:**

import os  
if os.path.exists("demofile.txt"):  
  os.remove("demofile.txt")  
else:  
  print("The file does not exist")

To delete an entire folder, use the os.rmdir() method:

Remove the folder "myfolder":

import os  
os.rmdir("myfolder")

You can only remove empty folders.

## **Get the list of Files and directories:**

# importing os module

import os

**# Get the list of all files and directories**

# in the root directory

path = "/"

dir\_list = os.listdir(path)

  print("Files and directories in '", path, "' :")

  # print the list

print(dir\_list)

Reading CSV File:

CSV files are the most common format of exchanging files in Data Science. Most online repositories would rather store files for download in CSV format than an xlsx format. Although they are both files in spreadsheet format, a CSV file is a simplified and lightweight spreadsheet that stores data as texts.

As a data scientist, it is extremely vital to learn how to work with CSV files. That's the aim of this tutorial. By the end of this article, you will learn

* What CSV files are
* What a CSV file looks like
* How to read and write data to a CSV file using the csv library
* How to read and write data to a CSV file using the pandas library

## Reading a CSV file

This function is used to read the data in a CSV file. The csv.reader method parses the data in the CSV file and returns the rows. A for loop can then be used to parse and print all the rows.

#open the file in text mode

with open('email.csv', 'rt') as file:

#store it I n a variable

data = csv.reader(file)

#parse each row and print the result

for each\_row in data:

print(each\_row)

Output:

['Login email', 'Identifier', 'First name', 'Last name']

['laura@example.com', '2070', 'Laura', 'Grey']

['craig@example.com', '4081', 'Craig', 'Johnson']

['mary@example.com', '9346', 'Mary', 'Jenkins']

['jamie@example.com', '5079', 'Jamie', 'Smith']

## How to Read a CSV File as a Dictionary.

To read a CSV file as a dictionary, you use the DictReader() method rather than the reader() method. Let’s see an example.

#open the file in text mode

with open('email.csv', 'rt') as file:

#store it in a variable

data = csv.DictReader(file)

#parse each row and print the result

for each\_row in data:

print(each\_row)

Output:

OrderedDict([('Login email', 'laura@example.com'), ('Identifier', '2070'), ('First name', 'Laura'), ('Last name', 'Grey')])

OrderedDict([('Login email', 'craig@example.com'), ('Identifier', '4081'), ('First name', 'Craig'), ('Last name', 'Johnson')])

OrderedDict([('Login email', 'mary@example.com'), ('Identifier', '9346'), ('First name', 'Mary'), ('Last name', 'Jenkins')])

OrderedDict([('Login email', 'jamie@example.com'), ('Identifier', '5079'), ('First name', 'Jamie'), ('Last name', 'Smith')])

## Writing Data to a CSV file

Just as it is possible to read data in a CSV, you can convert write data to a CSV file. To do this, you use the writer() method. The writerow() method is then used to add data to a row.

To see how this is done, see the example below.

#import necessary libraries

import csv

#open the CSV file

with open('writen\_data.csv', mode='w') as file:

data = csv.writer(file)

#write data to the file, row by row

data.writerow(['Login email', 'Identifier', 'First name', 'Last name'])

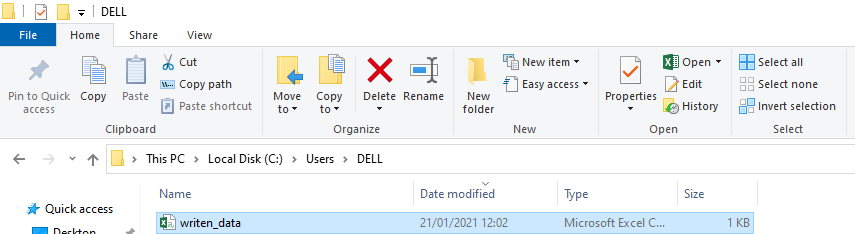
data.writerow(['laura@example.com', '2070', 'Laura', 'Grey'])

data.writerow(['craig@example.com', '4081', 'Craig', 'Johnson'])

data.writerow(['mary@example.com', '9346', 'Mary', 'Jenkins'])

data.writerow(['jamie@example.com', '5079', 'Jamie', 'Smith'])

Output:



Here’s what the file looks like