File Handling in python

Sometimes, it is not enough to only display the data on the console. The data to be displayed may be very large, and only a limited amount of data can be displayed on the console since the memory is volatile, it is impossible to recover the programmatically generated data again and again.

The file handling plays an important role when the data needs to be stored permanently into the file. A file is a named location on disk to store related information. We can access the stored information (non-volatile) after the program termination.

In Python, files are treated in two modes as text or binary. The file may be in the text or binary format, and each line of a file is ended with the special character like a comma (,) or a newline character. Python executes the code line by line. So, it works in one line and then asks the interpreter to start the new line again. This is a continuous process in Python.

Hence, a file operation can be done in the following order.

- Open a file
- Read or write - Performing operation
- Close the file

Create File in Python

Python is widely used in data analytics and comes with some inbuilt functions to work with files. We can create a file and do different operations, such as write a file and read a file using Python.

Create a Empty Text File

We don't have to import any module to create a new file. We can create a file using the built-in function **open().**

Example

open('file_Path', 'access_mode')

Pass the file name and access mode to the **open()** function to create a file. Access mode specifies the purpose of opening a file.

Below is the list of access modes for creating an a file.

File Mode	Meaning
`w`	Create a new file for writing. If a file already exists, it truncates the file first. Use to create and write content into a new file.
`x`	Open a file only 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`	Create a binary file
`t`	Create and open a file in a text mode
`r`	Open file for the reading purpose.

Example: **Create a new empty text file** named 'sales.txt'

```
# create a empty text file

# in current directory

fp = open('sales.txt', 'w')

fp.close()
```

Use **access mode** `w` if you want to create and write content into a file.

```
# create a empty text file

fp = open('sales_2.txt', 'w')
```

```
fp.write('first line')

fp.close()
```

Note:

- The file is created in the same directory where our program/script is running.
- If you have not specified any specific path(directory location), the file is created in the working directory. It is known as creating a file using the **relative path**. A relative path contains the current directory and then the file name.

Create File in A Specific Directory

To create a file inside a specific directory, we need to open a file using the absolute path. An absolute path contains the entire path to the file or directory that we need to use.

Let's see the example to **create a file for writing** using the absolute path.

```
# create a text file for writing

with open(r'E:\python_examples\reports\profit.txt', 'w') as fp:

    fp.write('This is first line')

    pass
```

Note: Using the **with** statement a **file is closed automatically** it ensures that all the resources that are tied up with the file are released.

Create a File If Not Exists

Sometimes it is essential not to create a new file if a file with the same name already exists in a given path. By default, when you open a file in write mode, it overwrites it if it exists. Else, create the new one.

```
import os

file_path = r'E:\pynative\account\profit.txt'
```

```
if os.path.exists(file_path):
    print('file already exists')
else:
    # create a file
    with open(file_path, 'w') as fp:
        # uncomment if you want empty file
        fp.write('This is first line')
```

Python Write to File

It is pretty standard that large chunks of data need to store in the Files. Python is widely used in data analytics and comes with some inbuilt functions to write data into files.

We can open a file and do different operations on it, such as write new contents into it or modify a file for appending content at the end of a file.

Access Modes for writing a file

Access mode specifying the **purpose of opening a file**.

Whenever we need to write text into a file, we have to open the file in one of the specified access modes. We can open the file basically to read, write or append and sometimes to do multiple operations on a single file.

To write the contents into a file, we have to open the file in write mode. Open a file using the built-in function called `open()`. This function takes two parameters, namely filename, and access mode, and returns the file pointer.

Steps for Writing Data into a File in Python

To write into a file, Please follow these steps:

1. **Find the path of a file**

   We can read a file using both relative path and absolute path. The path is the location of the file on the disk.
   An **absolute path** contains the complete directory list required to locate the file.
   A **relative path** contains the current directory and then the file name.

2. **Open file in write mode**

   Pass file path and access mode `w` to the `open()` function. The access mode opens a file in write mode.
   For example, `fp= open(r'File_Path', 'w')`

3. **Write content into the file**

   Once a file is opened in the `write` mode, write text or content into the file using the write() method.
   For example, `fp.write('new text').`
   The `write()` method will add new text at the beginning of a file. For an existing file, this new content will replace the existing content. If the file is not already present a new file will be created, and content is written into it.

4. **Close file after completing the write operation**

   We need to make sure that the file will be closed properly after completing the file operation. Use `fp.close()` to close a file.

5. **Append content at the end of the file**

   Pass file path and access mode `a` to the `open()` function to open a file in append mode.
   For example, `fp= open(r'File_Path', 'a')`
   Next, use the `write()` method to write content at the end of the file without deleting the existing content

Example: Write to a Text file in Python

```
text = "This is new content"

# writing new content to the file

fp = open("write_demo.txt", 'w')

fp.write(text)

print('Done Writing')

fp.close()

# Open the file for reading the new contents

fp = open("write_demo.txt", 'r')

print(fp.read())

fp.close()
```

File Write Methods

Python provides two different methods to write into a file. We don’t have to import any module for that. Below are the methods.

Method	Description
`write()`	Use to write a string into a file. It only accepts a string as the argument. Else it will raise a `TypeError: write() argument must be str`.
`writelines()`	Use to write a list of strings into a file. It accepts both string and list as the argument.

writelines(): Write a list of lines to a file

We can write multiple lines at once using the writelines() method. We can pass a list of strings that we want to add to the file to it. Use this method when you wanted to write a list into a file.

Example:

```
person_data = ['Name: Emma', '\nAddress: 221 Baker Street', '\nCity: London']
# writing string and list of lines to a file
fp = open("write_demo.txt", "w")
fp.writelines(person_data)
fp.close()

# opening the file in read mode
fp = open("write_demo.txt", "r")
print(fp.read())
fp.close()
```