18-02-2025

**File Handling:**

* File handling in Python allows you to read, write, update and manipulate files. Python provides built-in functions to work with files using the open() function.
* The open() function takes two parameters; filename and mode.
* There are four different modes for opening a file.
* “r”-Read. Opens a file for reading, error occurs 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.
* “x”-create. Create the specified file, returns an error if the file exists.
* In additional, we can specify if the file should be handled as binary or text mode.
* “t”-Text, text mode.
* “b”-Binary, Binary mode (ex: images)

**Why is File Handling Used?**

File handling is used for:

1. **Data Storage:** Storing user data, logs, reports, and other information persistently.
2. **Data Sharing:** Exchanging data between different applications or systems.
3. **Configuration Management:** Storing application settings in files like JSON or XML
4. **Logging and Debugging:** Recording logs for troubleshooting and performance tracking.

**Need for File Handling**

1. **Permanent Data Storage:** Data remains even after program execution ends.
2. **Data Management:** Helps in handling large volumes of data efficiently.
3. **User Input/Output:** Allows programs to accept input from or provide output to files instead of relying only on console interactions.
4. **Interoperability:** Files help in data exchange between different software systems.

**Syntax:**

* To open a file for reading it is needed to specify the name of the file.
* To open the file, use the built-in open() function. The open() function returns a file object, which has read() method for reading the content of the file.
* Ex: f = open("demo.txt", "r")

print(f.read())

output:

Hello! Welcome to demofile.txt

This file is for testing purposes.

Good Luck!

* If the file is located in a different location, you can also specify the file path, like this
* Ex: f=open("/content/demo.txt", "r")

print(f.read())

**Read only required parts of the file:**

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

Ex: f=open("demo.txt", "r")

print(f.read(5))

output: Hello

**Read lines:**

* We can return one line by using the readline() method.
* Ex: f=open("demo.txt", "r")

print(f.readline())

output: Hello! Welcome to demofile.txt

* By calling readline() two times, we can read the first two line from the file.
* By looping through the lines of the file, we can read the whole file, line by line.
* Ex: f=open("demo.txt", "r")

for i in f:

print(i)

output: Hello! Welcome to demofile.txt  
 This file is for testing purposes.  
 Good Luck!

**Close files:**

Close the file when you are finished with it. 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.

Ex: f=open("demo.txt", "r")

print(f.readline())

f.close()

output: Hello! Welcome to demofile.txt

**write:**

* 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

* Ex: open the file “demo.txt” and append content to the file

f = open("demo.txt", "a")

f.write("Now the file has more content!")

f.close()

#open and read the file after the

appending:

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

print(f.read())

output: Hello! Welcome to demofile.txt  
 This file is for testing purposes.  
 Good Luck!Now the file has more content!

* Open the file “demo.txt” and overwrite the content. The “w” method will overwrite the entire file.
* Ex: f = open("demo.txt", "w")

f.write("Woops! I have deleted the content!")

f.close()

#open and read the file after the overwriting:

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

print(f.read())

output:

Woops! I have deleted the content!

**Create a new file:**

* To create a new file in python, we use the open() method, with one of the following parameters:
* "x" - Create - will create a file, returns an error if the file exists.
* "a" - Append - will create a file if the specified file does not exists.
* "w" - Write - will create a file if the specified file does not exists.
* Ex: Create a file called "myfile.txt":

f = open("myfile.txt", "x")

result: a new empty file is created.

**Delete a file:**

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

Ex: import os

Os.remove(“demo.txt”)

After that when you execute the below code ,you will get that no such file or directory are present.

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

print(f.read())

output: No such file or directory: 'demo.txt'

**Check if file exist:**

To avoid getting an error, you might want to check if the file exists before you try to delete it.

Ex: check if file exists, then delete it.

import os

if os.path.exists("demo.txt"):

  os.remove("demo.txt")

else:

  print("The file does not exist")

output:

The file does not exist

**Delete Folder:**

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

import os  
os.rmdir("myfolder")

**Advantages of File Handling:**

**Easy Data Sharing** – Files can be shared between different programs, systems, or users without requiring an active database.

**Security & Access Control** – Files can be secured with permissions, encryption, and user authentication to prevent unauthorized access.

**Large Data Handling** – Files allow handling of large amounts of data that might not fit to RAM

**Flexibility in Data Storage** – Can store different types of data, such as text, binary, images, audio, and more.

**Cross-Platform Compatibility** – Files can be transferred and accessed across different operating systems.

**Disadvantages of File handling:**

**Slower Data Access** – Searching, updating, or modifying data in files can be slower compared to databases, especially with large files.

**Lack of Concurrent Access** – File systems don’t efficiently support multiple users accessing and modifying files simultaneously.

**Redundancy & Duplication** – Data duplication may occur as there is no built-in mechanism for avoiding redundancy like in databases.