**Types of python files:**

**Text Files :** These store data in plain text format. Examples include .txt files.

**Binary Files :** These store data in binary format, which is not human-readable. Examples include images, videos and executable files.

The four primary functions used for file handling in Python are:

* open() : Opens a file and returns a file object.
* read() : Reads data from a file.
* write() : Writes data to a file.
* close() : Closes the file, releasing its resources.

**File Operations in Python:**

**1.Mode:+** (Update Mode) −

**Description:**Opens the file for updating (reading and writing).

**2.Mode: r**  - Read-only Mode

**Description:** Read-only Mode is a state where a file, database, or system can be viewed but not modified. It ensures that data remains unchanged and prevents accidental edits or deletions.File must exist; otherwise, it raises an error.

**Example:**

file = open("geeks.txt", "r")

content = file.read()

print(content)

file.close())

**3.Mode: rb** - Read-only in binary Mode.

**Description:** Read-only in binary Mode" refers to opening a file in read Mode (r) while treating its contents as binary (b) rather than text. This is done using the Mode rb in Python's open() function.

File must exist; otherwise, it raises an error.

**Example:**

file = open("geeks.txt", "rb")

content = file.read()

print(content)

file.close()

**4.Mode: r+** -Read and write Mode.

**Description:** Read and write Mode refers to opening a file in a way that allows both reading from and writing to it. The Mode for this operation is "r+". This means:

* The file must already exist; otherwise, an error occurs.
* You can read the file contents.
* You can write new content to the file, modifying existing data.

**Example:**

# Open file in read and write Mode

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

# Read and print the existing content

content = file.read()

print("Before writing:", content)

# Move the cursor to the beginning and write new content

file.seek(0)

file.write("Hello, this is new content!")

# Open again to check changes

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

print("After writing:", file.read())

**5.Mode: rb+** -Read and write in binary Mode.

**Description:**Reading and writing in binary Mode means handling files as a stream of bytes instead of text. This is useful for non-text files like images, videos, PDFs, etc.

**Example:**

# Open an image file in binary Mode and copy it

with open("source.jpg", "rb") as src, open("copy.jpg", "wb") as dest:

dest.write(src.read())

**6.Mode: w**  -Write Mode.

**Description:** Write Mode ("w") is used to write data to a file. If the file already exists, it overwrites the existing content. If the file does not exist, it creates a new one.

**Example:**

# Open a file in write Mode

file = open("example.txt", "w")

# Write some text to the file

file.write("This is a test file.")

# Close the file

file.close()

**7.Mode: wb**  -Write in binary Mode.

**Description:** Writing in binary Mode means writing data to a file in binary format instead of text format. This is useful when working with non-text files like images, audio files.

**Example:**

# Writing in binary Mode

with open("example.bin", "wb") as file:

file.write(b"Hello, Binary World!") # Writing binary data

# Reading in binary Mode

with open("example.bin", "rb") as file:

data = file.read()

print(data) # Output will be in binary format

**8.Mode: w+** -Write and read Mode.

**Description:** Opens the file for both writing and reading. Creates a new file or truncates the existing file.

**Example:**

# Writing to the file

with open("example.txt", "w") as file:

file.write("Learning file handling in Python.")

# Reading from the file

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

content = file.read()

print("Read from file:", content)

**9.Mode: wb+**  -Write and read in binary Mode.

**Description:** Binary Mode means reading or writing files as raw bytes (b Mode). This is useful for non-text files like images, videos, or any file that isn’t plain text.

**Example:**

# Open file in binary write Mode

with open("example.bin", "wb") as file:

data = b"Hello, Binary World!" # Binary data (prefix `b` means bytes)

file.write(data)

# Open file in binary read Mode

with open("example.bin", "rb") as file:

content = file.read()

print(content) # Output: b'Hello, Binary World!'

**10.Mode: a**  -Append Mode.

**Description:** Append Mode ("a") in Python is used to add new data to the end of a file without deleting its existing content.

**Example:**

# Open the file in append Mode ('a')

with open("example.txt", "a") as file:

file.write("This is a new line.\n")

print("Text appended successfully!")

**11.Mode: ab**  -Append in binary Mode.

**Description:** Appending in binary Mode means adding data to an existing file without altering its previous content, while handling the file as binary data (not text). This is useful when working with non-text files like images, audio, or video.

**Example:**

with open("example.bin", "wb") as file:

file.write(b"Hello") # Writing initial binary data

with open("example.bin", "ab") as file:

file.write(b" World") # Appending binary data

with open("example.bin", "rb") as file:

content = file.read()

print(content) # Output: b'Hello World'

**12.Mode: a+** -Append and read Mode.

**Description:** The append ('a') and read ('r') Modes in Python's file handling are used for different purposes:

* Append Mode ('a'): Opens the file for writing but does not erase existing content. Instead, new content is added at the end of the file.
* Read Mode ('r'): Opens the file for reading. You cannot write to the file in this Mode.

**Example:**

# Open a file in write Mode to create it and add some text

with open("example.txt", "w") as file:

file.write("Hello, this is the first line.\n")

# Open file in append Mode

with open("example.txt", "a") as file:

file.write("This is an appended line.\n")

# Open file in read Mode

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

content = file.read()

print(content) # This will display both the original and appended lines

**Output:**

Hello, this is the first line.

This is an appended line.

**13.Mode: ab+** -Append and read in binary Mode.

**Description:** Appending and reading in binary Mode means that you open a file to both add new data at the end and read its existing content in binary format (which is useful for non-text files like images, audio, or even encoded data).

**Example:**

# Open a file in append and read binary Mode

with open("sample.bin", "ab+") as file:

# Writing (Appending) binary data

file.write(b"\x41\x42\x43\x44") # Writing 'ABCD' in binary format

# Move the file pointer to the beginning before reading

file.seek(0)

# Reading the binary data

content = file.read()

print("Binary content:", content)

**14.Mode: x** -Exclusive creation Mode

**Description:** When opening a file using "x" Mode (exclusive creation Mode), it creates a new file but fails if the file already exists. This helps prevent accidental overwrites.

**Example:**

try:

with open("sample.txt", "x") as file:

file.write("This is a new file.")

print("File created successfully!")

except FileExistsError:

print("File already exists!")

**15.Mode: xb** -Exclusive creation in binary Mode.

**Description:** Exclusive creation in binary Mode refers to opening a file for writing in binary format while ensuring that the file is created only if it does not already exist. This is done using the Mode "xb" in Python.

**Example:**

# Open a file in exclusive binary Mode

with open("example.bin", "xb") as file:

file.write(b"Hello, this is binary data!")

**16.Mode: x+** -Exclusive creation with read and write Mode.

**Description:** Exclusive creation with read and write Mode refers to opening a file in a way that ensures the file is created exclusively (i.e., it must not already exist) and allows both reading and writing. In Python, this is done using the "x+" Mode.

**Example:**

try:

with open("example.txt", "x+") as file:

file.write("Hello, this is an exclusive file!")

file.seek(0) # Move cursor to the beginning of the file

content = file.read()

print("File Content:", content)

except FileExistsError:

print("File already exists!")

**17.Mode: xb+**  -Exclusive creation with read and write in binary Mode.

**Description:** Exclusive creation with read and write in binary Mode means opening a file for both reading and writing in binary format, but only if the file does not already exist. If the file already exists, the operation will fail.

"x" → Exclusive creation (Fails if the file exists)

"b" → Binary Mode (Reads/Writes data in bytes)

"+" → Read and write Mode

**Example:**

try:

with open("example.bin", "xb+") as file:

file.write(b"Hello, Binary World!") # Writing bytes

file.seek(0) # Move to the beginning of the file

content = file.read() # Read the written data

print(content) # Output: b'Hello, Binary World!'

except FileExistsError:

print("File already exists!")

**18.Mode: tell()**

**Description :**In Python file handling, tell() is a method of file objects that returns the current position of the file pointer (cursor) within the file. It returns an integer representing the byte offset from the beginning of the file where the next read or write operation will occur.

**Example:**

# Open a file in read Mode

file = open('example.txt', 'r')

# Read the first 10 characters

content = file.read(10)

print(content)

# Check the current position of the file pointer

position = file.tell()

print("Current position:", position)

# Close the file

file.close()

**In this example:**

1. file.read(10) reads the first 10 characters from the file.
2. file.tell() returns the current position of the file pointer after reading.

## **Opening a File in Python**

To open a file we can use open() function, which requires file path and Mode as arguments.

**Example:**

# Open the file and read its contents

with open('geeks.txt', 'r') as file:

## **Closing a File**

Closing a file is essential to ensure that all resources used by the file are properly released. file.close() method closes the file and ensures that any changes made to the file are saved.

**Example:**

file = open("geeks.txt", "r")

# Perform file operations

file.close()

## **Handling Exceptions When Closing a File**

It’s important to handle exceptions to ensure that files are closed properly, even if an error occurs during file operations.

**Example:**

try:

file = open("geeks.txt", "r")

content = file.read()

print(content)

finally:

file.close()

## **Using ‘with’ Statement**

with statement is used for resource management. It ensures that file is properly closed after its suite finishes, even if an exception is raised. with open() as method automatically handles closing the file once the block of code is exited, even if an error occurs.

**Example:**

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

content = file.read()

print(content)

Output:

Hello, World!

Appended text.