File Handling

- **File handling:** It is a phenomenon of performing the functionalities inside the file system (such as creation deletion, retrivel, updating data and managing)
- **Stream:** stream is a channel that is created between the python and file for the data transaction to create a stream we have one way by using open function
- open(file_path_with_extension,mode): This is a function used to create a stream for the communication with a file from a python program and creating the file pointer and opening the file. After that it returns the file pointer address

Syntax

fopen = open(file_path_with_extension,mode)

Operations in File Handling

- **Opening a File**: Prepare a file for reading or writing.
    - **Modes:**
        - ‘r’: read(default mode)
        - ‘w’: write (create a new file or truncates an existing one)
        - ‘a’: append(write the end of the file)
        - ‘b’: binary mode(used with other modes like ‘rb’ or ‘wb’)
        - ‘x’: Exclusive creation(fails if the file already exist

Reading a file

- In python, file handling in read mode is done using the “r” mode. This mode is used to open a file for reading its content.
- Key features of “r” mode:

1)File must Exist: The file must exist otherwise, a FileNotFoundError is raised.

2)Read-only Mode: the file is opened only in read-only mode so you cannot write or modify its content

3)Default Mode: If no mode is specified while opening a file, “r” mode is used to opened

Syntax:

```
file = open("filename.txt", "r")
```

Reading Methods:

- **Read():**
- **Purpose**: Reads the entire content of the file as a single string.
- **Use Case**: When you want to load the whole file content at once into memory.
- Example:

```
with open("example.txt", "r") as file:
    content = file.read()
    print(content)
```

- **readline()**
- **Purpose**: Reads one line at a time from the file.
- **Use Case**: When you want to process the file line by line (e.g., for log files or structured data).
- Reads and returns a single line, including the newline character
- If the end of the file is reached,it returns an empty string (‘ ’)
- Example:

```
with open("example.txt", "r") as file:
first_line = file.readline()
print(first_line)
```

- **readlines():**
- **Purpose**: Reads all lines from the file and returns them as a list of strings.
- **Use Case**: When you want to work with all the lines in the file at once but still keep them as individual lines.
- Returns a list, where each item is a string representing a line in the file.
- Each line includes its newline character (\n) at the end

Example:

```
with open("example.txt", "r") as file:
    lines = file.readlines()
    print(lines)
```

Write method

- The write() method writes a specified string to file. It does not automatically add a newline character(\n); you must include it explicitly if needed
- **Syntax: file.write(string)**
- Write accepts only strings if you want to write non-string data(ex:integers or lists), you need to convert it to string using str()
- When used with write modes like ‘w’, write() overwrites the content of the file if it exists
- Each call to write appends data continuously unless you include a new line character (\n)
- **Writelines():**This function used to write the string data in to file object

- Tell():we use the tell() function to return the current position of the cursor (file pointer) from the beginning of the file
- Seek(): we use seek() function to move the cursor(file pointer) to a specific position

Append Mode:

In Python, **append mode** in file handling is used to add data to an existing file without overwriting its current contents. This is achieved using the ‘a’ mode

Opens the file for writing.

If the file does not exist, it creates a new file.

- Data is always written at the end of the file; existing content is not modified or deleted.
- Syntax:

```
With open(file_path, ’a’) as file
file.write(string)
```

Advance modes

- r+ Mode(Read and Write)
- **Description**: Opens the file for both reading and writing. The file must already exist, or it raises a file not found error
- **Cursor**: Starts at the beginning of the file.
- **Overwrite Behavior**: Overwrites content from the current cursor position.
- example:
- # r+ mode example

```
with open("example.txt", "r+") as file:
  print("Before writing:", file.read())
  file.seek(0)  #cursor position to add the content from starting
  file.write("New content written in r+ mode.\n")
  file.seek(0) #cursor position to read the content from file
  print("After writing:", file.read())
```

- a+ mode(append and read)
- **Description**: Opens the file for both reading and appending. If the file does not exist, it is created.
- **Cursor**: Starts at the end of the file for writing.
- **Overwrite Behavior**: Does not overwrite; only appends to the file.
- Example:
- # a+ mode example

```
with open("example.txt", "a+") as file:
    file.write("Appending new content in a+ mode.\n") # Write new
        content at the end of the file
    file.seek(0) # Move the cursor to the start to read the content
    print("File content after appending:", file.read())
```

- **W+ Mode(write and read):**
- **Description**: Opens the file for both writing and reading. If the file does not exist, it is created. If the file exists, its content is **truncated** (erased).
- **Cursor**: Starts at the beginning of the file.
- **Overwrite Behavior**: Overwrites the entire file (truncates).
- **Example:**

```
with open("example.txt", "w+") as file:  # Write new content (overwriting the file)
    file.write("Content written in w+ mode.\n")
# Move the cursor to the beginning to read the content
file.seek(0)
print("File content after writing:", file.read())
```

File Modes with Binary:

- rb: Read a file in binary mode.
- Wb: Write to a file in binary mode (overwrites the file if it exists).
- ab: Append to a file in binary mode.

Data Handling:

In binary mode, data is read or written as **bytes objects** rather than strings.

You cannot directly perform string operations; instead, you may need to encode/decode data to/from strings.

Reading a File in Binary Mode:

```
with open("example_image.jpg", "rb") as file:
    binary_data = file.read()
    print(type(binary_data))  # Output: <class 'bytes'>
    print(binary_data[:10])   # Display first 10 bytes of the file
```

- **Writing a File in Binary Mode:**

```
binary_content = b"This is some binary data."
with open("example.bin", "wb") as file:
    file.write(binary_content)
```

Appending Binary data

```
additional_data = b"\nAppended binary data."
with open("example.bin", "ab") as file:
   file.write(additional_data)
```

Copying an image file:

```
with open("source_image.jpg", "rb") as source_file:
   with open("copy_image.jpg", "wb") as destination_file:
      destination_file.write(source_file.read())
```

- **Use Cases of Binary Mode:**
- Images: Read/Write .jpg,.png,etc
- Videos: Handle .mp4,.avi,etc
- Audio files:Process .mp3,.wav, etc