You can easily move the file pointer back to the start using the seek() method. Here's how it works:

**Resetting File Pointer to Start**

file.seek(0)

* seek(0) moves the pointer to the **beginning** of the file.
* You can use this after reading or writing to **re-read** or **overwrite** content from the start.

**Example: Read Twice Without Reopening**

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

print("First read:")

print(file.read()) # Reads entire file

file.seek(0) # Move back to start

print("\nSecond read:")

print(file.read()) # Reads again from beginning

**Bonus: Other seek() Uses**

* file.seek(n) moves to the **n-th byte** in the file.
* file.seek(0, 2) moves to the **end** of the file (useful in 'r+' or 'a+' modes).

Great question, Mojaz — this taps into your curiosity about file handling and pointer control. Let’s break it down step by step:

**What is file.seek(0, 2)?**

The seek() method moves the file pointer to a specific byte position. It takes two arguments:

file.seek(offset, whence)

* offset: Number of bytes to move.
* whence: Reference point:
  + 0 → beginning of file
  + 1 → current position
  + 2 → end of file

So file.seek(0, 2) means:

"Move the pointer **0 bytes** from the **end of the file**" → i.e., go to the very end.

**Why is this useful in 'r+' or 'a+' modes?**

| **Mode** | **Purpose** | **Why seek(0, 2) Helps** |
| --- | --- | --- |
| 'r+' | Read + Write | You can jump to the end to **append** without truncating. |
| 'a+' | Append + Read | File pointer starts at end; seek(0, 2) ensures you're ready to **write at the end**. |

**DRY RUN Example**

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

file.seek(0, 2) # Move to end

file.write("New entry\n") # Appends safely

Without seek(0, 2), you might accidentally read or write from the wrong position — especially in 'r+' mode where the pointer starts at the beginning.