Come calcolare il checksum MD5 di un file con Python

Il checksum MD5 è una firma digitale che viene generata per verificare l'integrità di un file. Viene spesso usato per assicurarsi che un file non sia stato alterato o corrotto durante il trasferimento. Python fornisce strumenti semplici ed efficaci per calcolare il checksum MD5 di un file attraverso il modulo `hashlib`. Di seguito, esploreremo come utilizzare questo modulo per calcolare il checksum MD5 di un file in pochi passi.

Per evitare di sovraccaricare la memoria, specialmente con file di grandi dimensioni, è consigliabile leggere il file a blocchi. In Python, possiamo farlo con un ciclo che legge il file in blocchi di una dimensione definita, ad esempio 4096 byte.

Dopo aver letto il file, possiamo aggiornare l'oggetto hash per ogni blocco di dati letto. Questo ci permette di ottenere l'MD5 del file completo senza caricare l'intero contenuto in memoria.

```
import hashlib

def calculate_md5(file_path):
    # Creiamo un oggetto md5
    md5_hash = hashlib.md5()

    # Apriamo il file in modalità lettura binaria
    with open(file_path, "rb") as f:
        # Leggiamo il file a blocchi
        for block in iter(lambda: f.read(4096), b""):
            md5_hash.update(block)
```

```
    # Restituiamo l'hex digest, ovvero il checksum
MD5 come stringa esadecimale
    return md5_hash.hexdigest()

# Utilizzo della funzione
fpath = "percorso/del/tuo/file"
checksum = calculate_md5(fpath)
print("Checksum MD5:", checksum)
```

Spiegazione:

1. **Creazione dell'oggetto MD5**: `hashlib.md5()` crea un oggetto hash utilizzando l'algoritmo MD5.
2. **Apertura del File**: Apriamo il file in modalità lettura binaria (`"rb"`) per garantire che il contenuto del file sia letto come sequenza di byte, indipendentemente dal tipo di file.
3. **Lettura a Blocchi**: Utilizzando `iter(lambda: f.read(4096), b"")`, leggiamo il file in blocchi di 4096 byte alla volta. Questo approccio è particolarmente utile per file di grandi dimensioni, poiché riduce l'utilizzo della memoria.
4. **Aggiornamento dell'Hash**: Per ogni blocco letto, aggiorniamo l'oggetto hash tramite `hash_md5.update(blocco)`.
5. **Calcolo e Output dell'Hash MD5**: Dopo aver letto tutto il file, otteniamo il checksum MD5 usando `hash_md5.hexdigest()`, che restituisce l'hash come stringa esadecimale.

Conclusione

L'algoritmo MD5 è veloce ed efficace per il controllo dell'integrità dei file, ma non è ideale per applicazioni che richiedono un alto livello di sicurezza,

poiché non è considerato a prova di collisione. In applicazioni critiche dal punto di vista della sicurezza, è meglio usare algoritmi più robusti come SHA-256. Tuttavia, per controlli di integrità basilari e per individuare modifiche accidentali nei file, l'MD5 rimane una scelta pratica e ampiamente utilizzata.