Python
File Integrity Checker

```
import os
import hashlib

def calculate_hashes(file_path):
    md5_hash = hashlib.md5()
    sha256_hash = hashlib.sha256()

    with open(file_path, "rb") as f:
        while True:
            data = f.read(65536)
            if not data:
                break
            md5_hash.update(data)
            sha256_hash.update(data)

    return md5_hash.hexdigest(), sha256_hash.hexdigest()

def check_integrity(directory_path):
    if not os.path.exists(directory_path) or not
os.path.isdir(directory_path):
        print(f"Directory '{directory_path}' does not exist or is not a
valid directory.")
        return

    for root, dirs, files in os.walk(directory_path):
        for file_name in files:
            file_path = os.path.join(root, file_name)
            md5_hash, sha256_hash = calculate_hashes(file_path)
            print(f"File: {file_path}\nMD5 Hash: {md5_hash}\nSHA-256 Hash:
{sha256_hash}\n")

if __name__ == "__main__":
    while True:
```

```
        provide_directory = input("Will you provide a directory path to
check integrity? (yes/no): ").strip().lower()

        if provide_directory == "yes":
            directory_to_check = input("Enter the directory path: ")
            check_integrity(directory_to_check)
            break
        elif provide_directory == "no":
            while True:
                provide_file_path = input("Will you provide a file path
instead? (yes/no): ").strip().lower()
                if provide_file_path == "yes":
                    file_path = input("Enter the file path: ")
                    if os.path.exists(file_path) and
os.path.isfile(file_path):
                        md5_hash, sha256_hash =
calculate_hashes(file_path)
                        print(f"File: {file_path}\nMD5 Hash:
{md5_hash}\nSHA-256 Hash: {sha256_hash}\n")
                        break
                    else:
                        print(f"File '{file_path}' does not exist. Please
enter a valid file path.")
                elif provide_file_path == "no":
                    print("Okay, have a good day!")
                    break
                else:
                    print("Invalid input. Please enter 'yes' or 'no'.")
            break
        else:
            print("Invalid input. Please enter 'yes' or 'no'.")
```

When downloading files from websites today, there's a risk of tampering, whether through malicious individuals exploiting the site, network vulnerabilities, or compromises on personal computers. To address this, some downloads include a checksum value

like SHA-256, acting as a digital fingerprint for the file. By computing this checksum ourselves using Python and libraries like os for system interactions and hashlib for SHA-256 and MD5 computation, we verify that the downloaded file matches the original.

This Python script provides SHA-256 and MD5 checksum values, empowering users to protect against malicious downloads. It's crucial to note that even checksums can be tampered with. For added security, verify checksums on a separate, secure device and network to ensure authenticity.

In total, it's important to remain vigilant against tampering in various forms: the integrity of the checksum displayed on your screen, potential alterations to the file linked from the download button, or tampering that could occur during the actual download process. This proactive approach ensures the safety and authenticity of downloaded files, offering peace of mind in digital interactions by verifying file integrity before use.

File Integrity Checker

Purpose: This Python script is designed to ensure the integrity of files by calculating and displaying MD5 and SHA-256 hashes. These hashes act as unique digital fingerprints that can verify the authenticity and detect any alterations in files.

Components:

1. **calculate_hashes(file_path):**
    - This function computes both MD5 and SHA-256 hashes for a specified file.
    - **Parameters:**
        - `file_path`: The path to the file for which hashes are to be computed.
    - **Returns:**

        - A tuple containing the MD5 and SHA-256 hashes as hexadecimal strings.
2. **check_integrity(directory_path):**
    - This function checks the integrity of files within a specified directory.
    - **Parameters:**
        - `directory_path`: The path to the directory to check.
    - **Behavior:**
        - It iterates through all files in the directory (including subdirectories) using `os.walk()`.
        - For each file, it computes MD5 and SHA-256 hashes using `calculate_hashes()` and prints the results, indicating the file path and corresponding hashes.
3. **Main Execution:**
    - The script starts by prompting the user to provide either a directory path or a file path for integrity verification.
    - Depending on the user's input, it either:
        - Calls `check_integrity()` to check all files within the specified directory.
        - Computes and displays hashes directly for the specified file path using `calculate_hashes()`.
    - The script continues to prompt until valid input ("yes" or "no") is received from the user.

Usage:

- When executed, the script interacts with the user through input prompts to determine the scope of integrity verification (directory or single file).
- It computes MD5 and SHA-256 hashes for:
    - All files within the specified directory.
    - Or a single specified file, ensuring that the hashes match those expected for genuine files.

Pre-requisites:

- Python 3.x installed.
- Availability of the `os` and `hashlib` modules (standard library modules for system interaction and hashing operations).

Conclusion: By utilizing this file integrity checker in Python, users can verify the safety and authenticity of downloaded files, thereby enhancing security and confidence in digital interactions.