# Task 1: FILE INTEGRITY CHECKER

## Instruction:

BUILD A TOOL TO MONITOR CHANGES IN FILES BY CALCULATING AND COMPARING HASH VALUES.

## DELIVERABLE:

A PYTHON SCRIPTUSING LIBRARIES LIKE HASHLIB TO ENSURE FILE INTEGRITY.

### Script:

```
import hashlib
import os
import json
import time

def calculate_file_hash(file_path):
    sha256_hash = hashlib.sha256()
    with open(file_path, "rb") as f:
        for byte_block in iter(lambda: f.read(4096), b""):
            sha256_hash.update(byte_block)
    return sha256_hash.hexdigest()

def load_hash_database(db_file):
    if os.path.exists(db_file):
        with open(db_file, "r") as f:
            return json.load(f)
    return {}
```

```
def save_hash_database(db_file, hash_db):
    with open(db_file, "w") as f:
        json.dump(hash_db, f, indent=4)

def monitor_files(directory, db_file):
    hash_db = load_hash_database(db_file)

    while True:
        for root, _, files in os.walk(directory):
            for file in files:
                file_path = os.path.join(root, file)
                current_hash = calculate_file_hash(file_path)

                if file_path in hash_db:
                    if hash_db[file_path] != current_hash:
                        print(f"File changed: {file_path}")
                        hash_db[file_path] = current_hash
                else:
                    print(f"New file detected: {file_path}")
                    hash_db[file_path] = current_hash

        save_hash_database(db_file, hash_db)
        time.sleep(10)  # Check every 10 seconds

if __name__ == "__main__":
    directory_to_monitor = "path/to/directory"  # Replace with
    hash_database_file = "hash_database.json"

    print(f"Monitoring directory: {directory_to_monitor}")
    print("Press Ctrl+C to stop monitoring")

    try:
        monitor_files(directory_to_monitor, hash_database_file)
    except KeyboardInterrupt:
        print("\nMonitoring stopped")
```

## Description:

This script will calculate SHA-256 hashes of specified files and compare them to previously stored hashes to detect any changes.

## To use this script:

1. Replace `"path/to/directory"` with the actual path of the directory you want to monitor.
2. Run the script. It will start monitoring the specified directory and print messages when files are changed or new files are detected.
3. The script will create a `hash_database.json` file to store the hashes of monitored files.
4. To stop the monitoring, press `Ctrl+C`.

> This implementation ensures file integrity by continuously calculating and comparing hash values, meeting the requirements of the task.