```
#!/usr/bin/env python3

# ==============================
#  MD5 Hash Match Finder
#  Author: Scott Russo
#  Description:
#  This script goes through every file in a folder called 'suspicious_files'
#  and checks which file matches a specific MD5 hash (a digital fingerprint).
#  This helps us identify if a known malicious file is present.
# ==============================

# Step 1: Import two built-in Python modules.
import hashlib  # Lets us create MD5 hashes (digital fingerprints)
import os       # Lets us work with folders and file paths

# Step 2: Set the MD5 hash we're looking for.
# This is the known hash we're trying to match. If a file has this hash,
# it means it's the file we're looking for (possibly malware or a known sample).
target_hash = "29638606a7ac6ebea40b9be9358b9943"

# Step 3: Set the name of the folder that holds the files to check.
# This should be a real folder in the same location as this script.
folder = "suspicious_files"

# Step 4: Create a function to calculate the MD5 hash of any file.
def calculate_md5(filepath):
    """
    This function opens a file and reads it in small chunks.
    It builds an MD5 hash from the file's contents.
    Returns the final hash as a string.
    """
    md5 = hashlib.md5()  # Start a new, empty MD5 hash object

    try:
        # Open the file in binary mode (read bytes, not text)
        with open(filepath, "rb") as f:
            # Read 4 KB of the file at a time until it's done
            for chunk in iter(lambda: f.read(4096), b""):
                md5.update(chunk)  # Add each chunk to the MD5 object

        # Return the full MD5 hash in readable form (hexadecimal)
        return md5.hexdigest()

    except Exception as e:
```

```
        # If something goes wrong (like file access issues), return nothing
        return None

# Step 5: Check if the suspicious_files folder exists.
# If it doesn't exist, we stop the program early and show an error.
if not os.path.isdir(folder):
    print(f"Folder not found: {folder}")
    exit()  # Exit the script early

# Step 6: Print a message to let the user know scanning is starting.
print(f"\nScanning '{folder}' for file matching MD5: {target_hash}\n")

# Step 7: Set a flag to track if we find a matching file
match_found = False

# Step 8: Go through every item (file or folder) in the suspicious_files folder
for filename in os.listdir(folder):
    # Build the full file path (folder + filename)
    filepath = os.path.join(folder, filename)

    # Only process real files — skip folders or weird items
    if os.path.isfile(filepath):
        # Get the file's actual MD5 hash
        file_hash = calculate_md5(filepath)

        # If the hash matches the one we're looking for
        if file_hash == target_hash:
            print("Match found!")
            print(f"File: {filename}")
            print(f"MD5:  {file_hash}")
            match_found = True  # We found it, no need to keep looking
            break  # Stop the loop early — job done

# Step 9: If we checked everything and didn't find a match
if not match_found:
    print("No matching file found.")
```

```
┌──(scott㉿kali)-[~]
└─$ cd Downloads

┌──(scott㉿kali)-[~/Downloads]
└─$ cd Python_Exercise

┌──(scott㉿kali)-[~/Downloads/Python_Exercise]
└─$ nano hash_machine.py

┌──(scott㉿kali)-[~/Downloads/Python_Exercise]
└─$ python3 hash_machine.py

Scanning 'suspicious_files' for file matching MD5: 29638606a7ac6ebea40b9be935
8b9943

Match found!
File: DhYU3g9P
MD5:  29638606a7ac6ebea40b9be9358b9943

┌──(scott㉿kali)-[~/Downloads/Python_Exercise]
└─$
```