CMR ENGINEERING COLLEGE
UGC AUTONOMOUS
(Approved by AICTE - New Delhi. Affiliated to JNTUH and Accredited by NAAC & NBA)
Kandlakoya (V), Medchal (M), Medchal - Malkajgiri (D)-501401

DEPARTMENT OF INFORMATION TECHNOLOGY

FOR ACADEMIC YEAR 2023-2024

SUB: PYTHON PROGRAMMING LAB **BRANCH & SEC :** IT-B
BATCH NO: 1 **Max. Marks:** 10

PROJECT WORK

Submitted by

MOHAMMED ARSH KHAN 228R1A1294

Under the esteemed guidance of:

Mrs. C. Madhuri

Assistant Professor

Awarded Marks: **Signature of the Faculty:**

Project :

Project Report: Check if two PDF documents are identical with Python.

Title:

Comparing PDF Documents for Identical Content using Python

Aim:

The aim of this project is to develop a Python script that compares two PDF documents to determine if their content is identical or not.

Description:

This project focuses on creating a Python script that can efficiently compare the content of two PDF documents. By calculating the SHA-1 hash of each file and comparing the hashes, the script can determine if the documents are identical. This process ensures accurate comparison regardless of the size or complexity of the PDF files.

Objectives:

1. Develop a Python script to calculate the SHA-1 hash of PDF files.
2. Implement a method to compare the hashes of two PDF documents.
3. Verify the effectiveness and efficiency of the script in identifying identical PDF files.

Implementation:

The implementation involves using the `hashlib` library to compute the SHA-1 hash of each file. The script reads the files in small chunks to handle large files efficiently. The `hash_file` function takes two filenames as input parameters, computes the hashes for each file, and returns them. Finally, the script compares the hexdigests of the hash objects to determine if the files are identical.

Algorithm:

1. Open the first file (`fileName1`) in binary mode and initialize a SHA-1 hash object (`h1`).

2. Read the file in small chunks and update the hash object with each chunk.

3. Repeat the process for the second file (`fileName2`) and hash object (`h2`).

4. Compare the hexdigests of the hash objects.

5. If the hexdigests are equal, the files are identical; otherwise, they are not.

Code Implementation:

python

```
import hashlib


def hash_file(fileName1, fileName2):
    h1 = hashlib.sha1()
    h2 = hashlib.sha1()

    with open(fileName1, "rb") as file:
```

```
        chunk = 0
        while chunk != b'':
            chunk = file.read(1024)
            h1.update(chunk)

    with open(fileName2, "rb") as file:
        chunk = 0
        while chunk != b'':
            chunk = file.read(1024)
            h2.update(chunk)

    return h1.hexdigest(), h2.hexdigest()

msg1, msg2 = hash_file("pd1.pdf", "pd1.pdf")

if(msg1 != msg2):
    print("These files are not identical")
else:
    print("These files are identical")
```

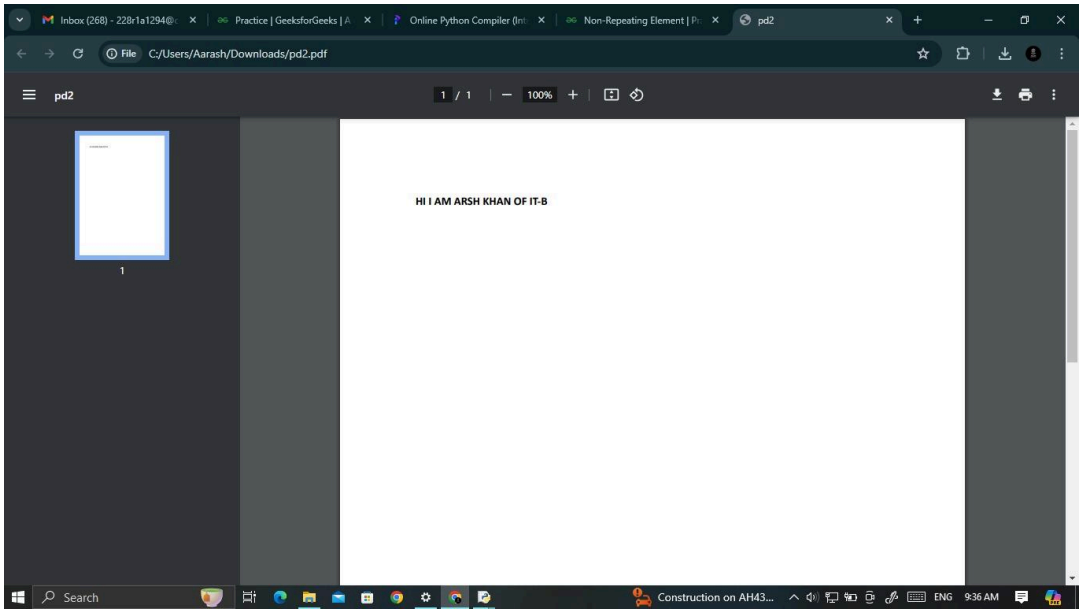
Execution:

To execute the script, save it with a `.py` extension and ensure that the PDF files you want to compare are in the same directory as the script. Run the script in a Python environment, and it will output whether the files are identical or not.

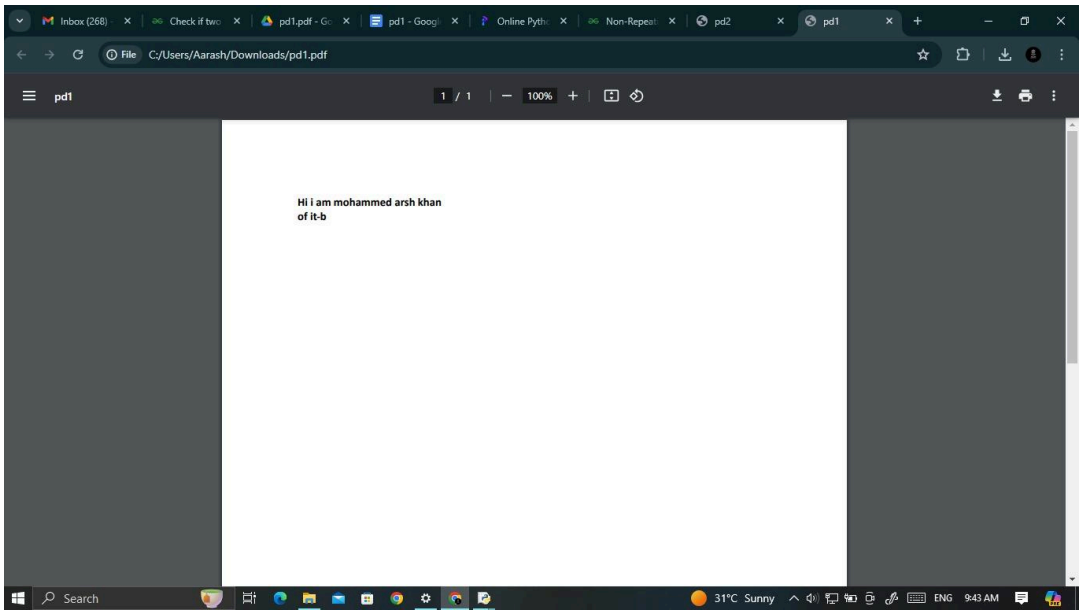
Output:

If the output is "These files are identical," it means the content of the two PDF documents is the same. Otherwise, if the output is "These files are not identical," the content differs between the two documents.

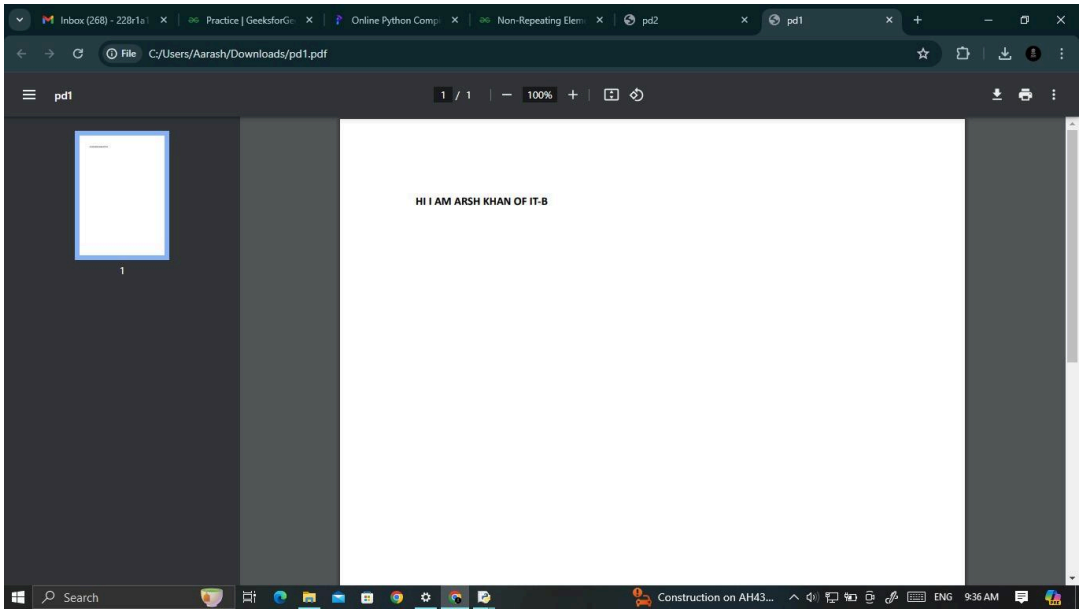
Output screenshots:



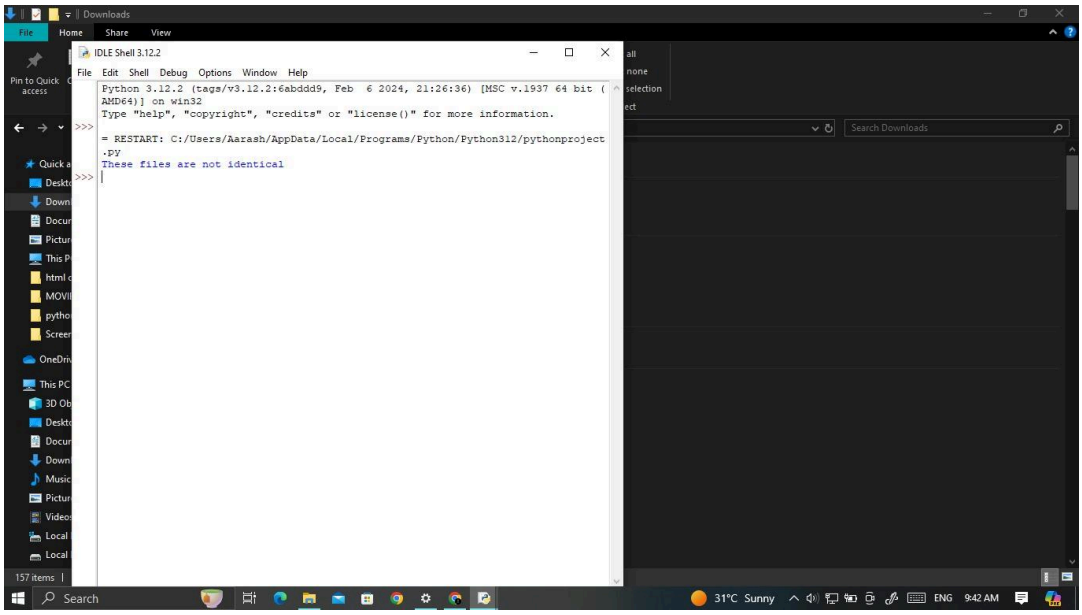
In above figure it is showing our PDF 1



In above figure we can see the PDF 2



In the above screenshot we can see it is our PDF 3



In the above screenshot we can see that when we compare PDF 1 and PDF 2, it shows that these files are not identical.

In the above screenshot we can see that when we compare PDF 1 and PDF 3, it shows that these files are identical.

Conclusion:

The Python script successfully compares two PDF documents by calculating their SHA-1 hashes. It provides a reliable method for determining if the files have identical content, which can be valuable in various applications requiring file integrity verification. This project demonstrates an effective approach to compare files efficiently using Python.Code.