Report

Directory Compression and Decompression

This report provides an overview of two Python scripts that enable directory compression into a ZIP file and directory decompression from a ZIP file. The first script, "Compressing Files," allows the user to compress files within a directory into a ZIP archive, calculate the compression ratio, and display file sizes. The second script, "Decompressing Files," enables users to extract the contents of a ZIP file into a specified directory.

- Compressing Directory
    - The script prompts the user to enter the path to a directory containing files to be compressed.
    - It verifies whether the input directory exists. If not, it displays an error message and exits.
    - The user is prompted to specify the path and filename for the output ZIP file.
    - The script utilizes the `zipfile` module to compress the directory into a ZIP file while preserving the directory structure. The `ZIP_DEFLATED` compression method is used.
    - Original file sizes within the directory and the compressed ZIP file size are calculated using the `os.path.getsize` function.
    - The compression ratio, defined as the ratio of compressed file size to original file size, is calculated and displayed.
    - A success message indicates the completion of directory compression.
- Decompressing ZIP File
    - The script prompts the user to enter the path to the ZIP file to be decompressed.
    - It checks whether the specified ZIP file exists. If the file is not found, it displays an error message and exits.
    - The user is prompted to specify the path for extracting the ZIP file's contents.
    - The script ensures that the extraction path exists or creates it if necessary.
    - The `zipfile` module is used to decompress the ZIP file, extracting its contents
    - A success message indicates the completion of ZIP file extraction.

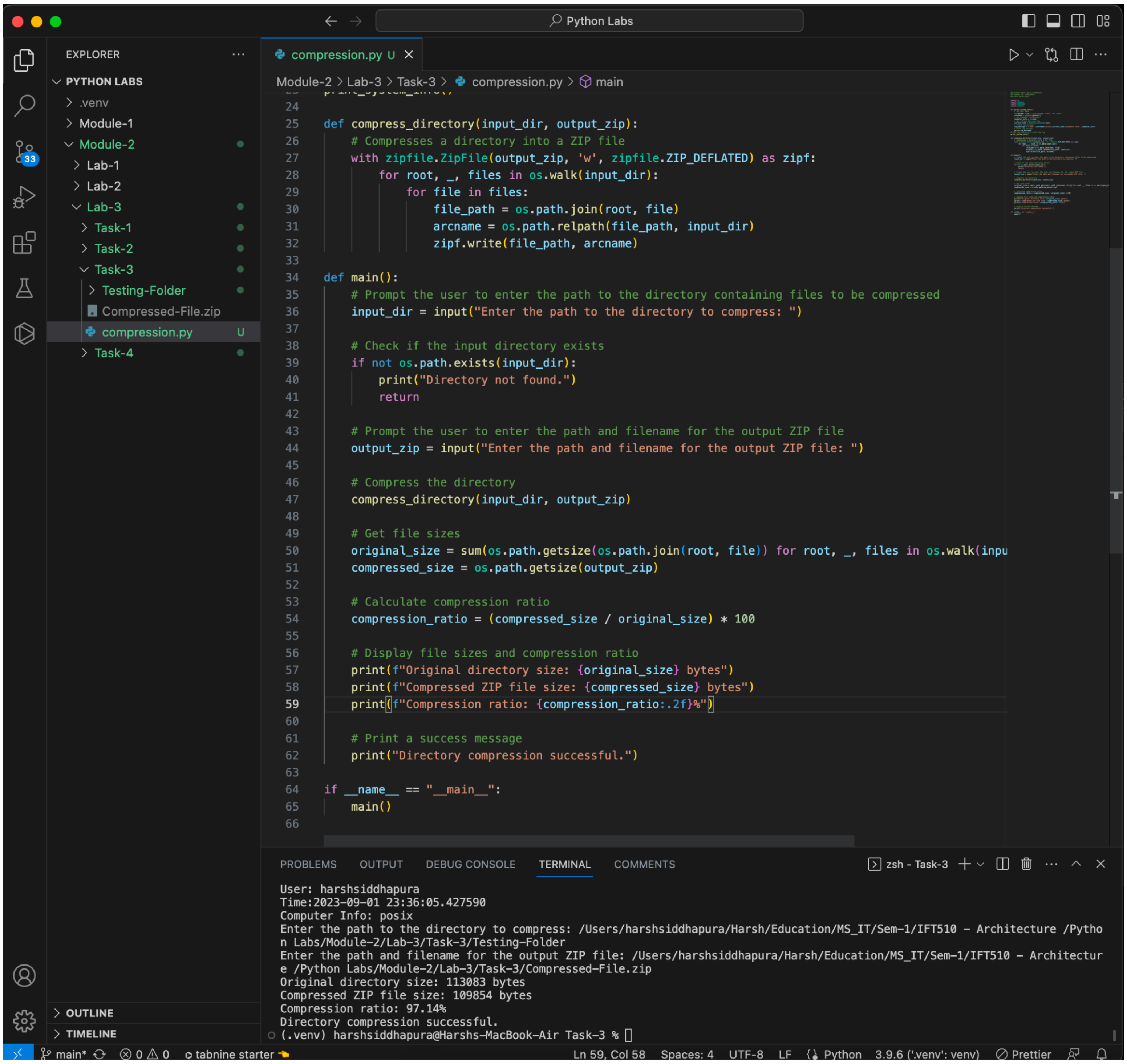
These scripts offer practical functionality for managing files and directories:

- Compressing Files: Useful for bundling multiple files into a single compressed archive, reducing storage space, and simplifying file transfer.
    - Data Backup and Archiving: Users can compress directories containing valuable data, creating efficient backups that consume less storage space.
    - File Transfer: Compressed ZIP files are ideal for sending large sets of files over networks or via email, as they simplify the transfer process and reduce transmission times.
    - Software Packaging: Developers use directory compression to package software and its associated resources into a single distributable ZIP archive.

- Decompressing Files: Essential for extracting the contents of ZIP archives, whether for data recovery, file access, or installation purposes.
    - Data Recovery: Decompression is vital for recovering data from compressed archives, making it possible to access files that may have been backed up in ZIP format.
    - Software Installation: Many software packages are distributed in ZIP archives, and decompression is the first step in the installation process.
    - File Access: ZIP files are commonly used to group related files, and decompression provides a convenient method for accessing these files without extracting the entire archive.

In conclusion, these Python scripts provide a straightforward and user-friendly means of compressing directories into ZIP files and decompressing ZIP files into specified directories. They demonstrate the capabilities of the `zipfile` module and the `os` module for file handling and manipulation. These tools can be invaluable for file management tasks in various applications, including data backup, software distribution, and data extraction.

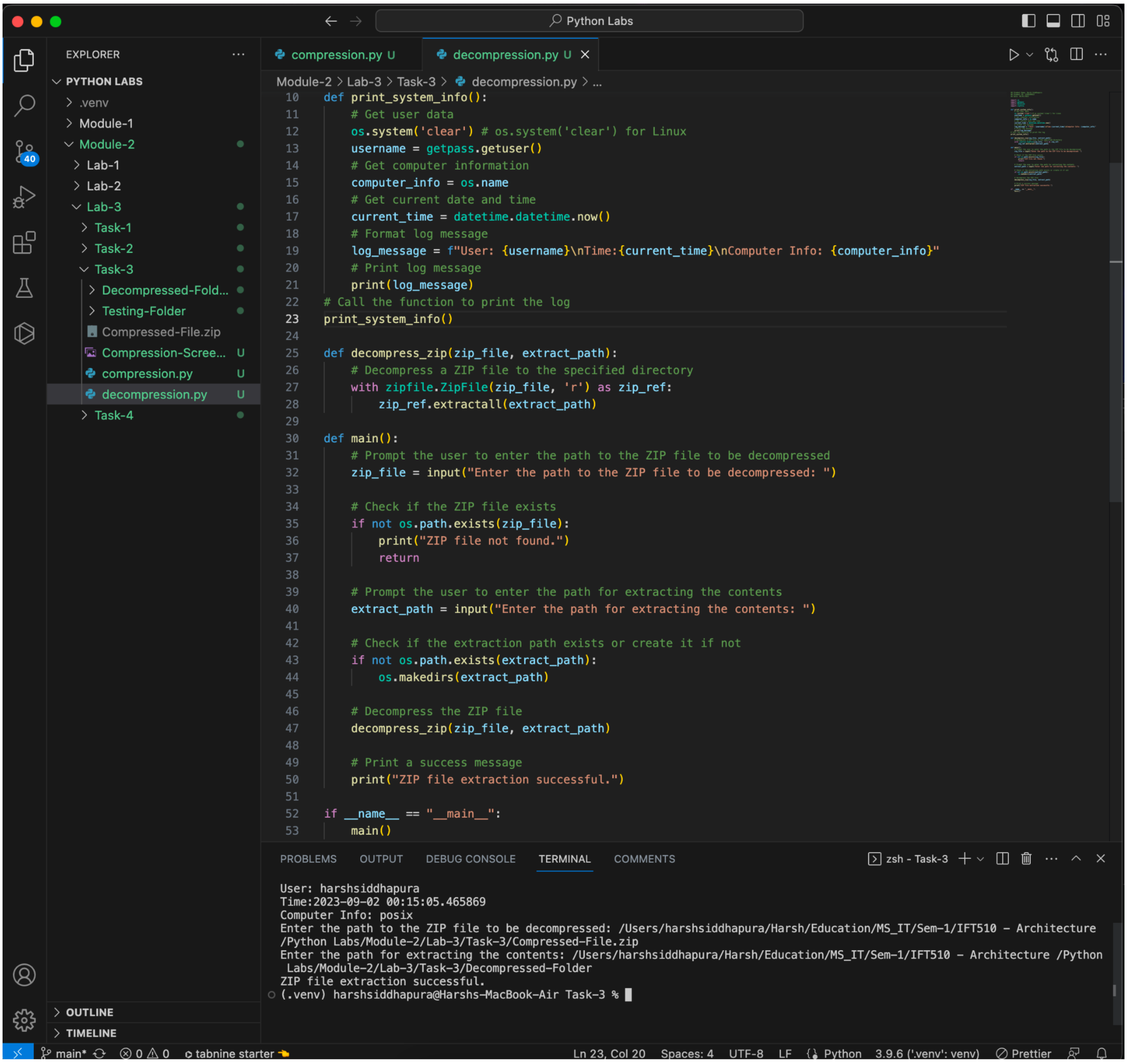
Compression



```python

def compress_directory(input_dir, output_zip):
    # Compresses a directory into a ZIP file
    with zipfile.ZipFile(output_zip, 'w', zipfile.ZIP_DEFLATED) as zipf:
        for root, _, files in os.walk(input_dir):
            for file in files:
                file_path = os.path.join(root, file)
                arcname = os.path.relpath(file_path, input_dir)
                zipf.write(file_path, arcname)

def main():
    # Prompt the user to enter the path to the directory containing files to be compressed
    input_dir = input("Enter the path to the directory to compress: ")

    # Check if the input directory exists
    if not os.path.exists(input_dir):
        print("Directory not found.")
        return

    # Prompt the user to enter the path and filename for the output ZIP file
    output_zip = input("Enter the path and filename for the output ZIP file: ")

    # Compress the directory
    compress_directory(input_dir, output_zip)

    # Get file sizes
    original_size = sum(os.path.getsize(os.path.join(root, file)) for root, _, files in os.walk(inpu
    compressed_size = os.path.getsize(output_zip)

    # Calculate compression ratio
    compression_ratio = (compressed_size / original_size) * 100

    # Display file sizes and compression ratio
    print(f"Original directory size: {original_size} bytes")
    print(f"Compressed ZIP file size: {compressed_size} bytes")
    print(f"Compression ratio: {compression_ratio:.2f}%")

    # Print a success message
    print("Directory compression successful.")

if __name__ == "__main__":
    main()

```

```
User: harshsiddhapura
Time:2023-09-01 23:36:05.427590
Computer Info: posix
Enter the path to the directory to compress: /Users/harshsiddhapura/Harsh/Education/MS_IT/Sem-1/IFT510 - Architecture /Pytho
n Labs/Module-2/Lab-3/Task-3/Testing-Folder
Enter the path and filename for the output ZIP file: /Users/harshsiddhapura/Harsh/Education/MS_IT/Sem-1/IFT510 - Architectur
e /Python Labs/Module-2/Lab-3/Task-3/Compressed-File.zip
Original directory size: 113083 bytes
Compressed ZIP file size: 109854 bytes
Compression ratio: 97.14%
Directory compression successful.
(.venv) harshsiddhapura@Harshs-MacBook-Air Task-3 %
```

Decompression



```python
def print_system_info():
    # Get user data
    os.system('clear') # os.system('clear') for Linux
    username = getpass.getuser()
    # Get computer information
    computer_info = os.name
    # Get current date and time
    current_time = datetime.datetime.now()
    # Format log message
    log_message = f"User: {username}\nTime:{current_time}\nComputer Info: {computer_info}"
    # Print log message
    print(log_message)
# Call the function to print the log
print_system_info()

def decompress_zip(zip_file, extract_path):
    # Decompress a ZIP file to the specified directory
    with zipfile.ZipFile(zip_file, 'r') as zip_ref:
        zip_ref.extractall(extract_path)

def main():
    # Prompt the user to enter the path to the ZIP file to be decompressed
    zip_file = input("Enter the path to the ZIP file to be decompressed: ")

    # Check if the ZIP file exists
    if not os.path.exists(zip_file):
        print("ZIP file not found.")
        return

    # Prompt the user to enter the path for extracting the contents
    extract_path = input("Enter the path for extracting the contents: ")

    # Check if the extraction path exists or create it if not
    if not os.path.exists(extract_path):
        os.makedirs(extract_path)

    # Decompress the ZIP file
    decompress_zip(zip_file, extract_path)

    # Print a success message
    print("ZIP file extraction successful.")

if __name__ == "__main__":
    main()
```

```
User: harshsiddhapura
Time:2023-09-02 00:15:05.465869
Computer Info: posix
Enter the path to the ZIP file to be decompressed: /Users/harshsiddhapura/Harsh/Education/MS_IT/Sem-1/IFT510 - Architecture
/Python Labs/Module-2/Lab-3/Task-3/Compressed-File.zip
Enter the path for extracting the contents: /Users/harshsiddhapura/Harsh/Education/MS_IT/Sem-1/IFT510 - Architecture /Python
 Labs/Module-2/Lab-3/Task-3/Decompressed-Folder
ZIP file extraction successful.
(.venv) harshsiddhapura@Harshs-MacBook-Air Task-3 %
```