COMP 5710
Group Project

Report

For 4.C of the project we integrated forensics to 5 simple python scripts. These scripts are simple scripts like add, subtract, multiply, divide, and add files. These are super simple scripts to read and understand. We implemented logging with comments to each script for integrating forensics. The scripts are below that were made for this part of the group project.

For each file I wrote it in Notepad and ran them in WindowsPowershell. The commands used are at the end of this report.

The first script is just a simple addition script.

File One:

```
import logging

# Configure logging to also output to the console
logging.basicConfig(filename='FileOne.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

def add(a, b):
    try:
        # Perform the addition
        result = a + b
        # Log the operation
        logging.info(f"Add operation performed: {a} + {b} = {result}. This logs the result of adding two numbers.")
        return result
    except Exception as e:
        # Log the exception
        logging.exception("Error in add method: An exception occurred while attempting to add values.")
        return None

# Example call to the add function
if __name__ == "__main__":
    add(5, 7)  # This should generate a log entry
```

This is the output from the log file that was given.

```
2024-10-27 20:10:45,155 - INFO - Add operation performed: 5 + 7 = 12. This logs the result of adding two numbers.
```

The second file is just a simple subtraction script.

File Two:

```
import logging

# Configure logging to include timestamps and log level
logging.basicConfig(filename='FileTwo.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

def subtract(a, b):
    try:
        # Perform the subtraction
        result = a - b
        # Log the operation with a clear comment
        logging.info(f"Subtract operation performed: {a} - {b} = {result}. This logs the result of
subtracting two numbers.")
        return result
    except Exception as e:
        # Log the exception with a clear context
        logging.exception("Error in subtract method: An exception occurred while attempting to
subtract values.")
        return None

# Example call to the subtract function
if __name__ == "__main__":
    subtract(10, 5)  # This should generate a log entry
```

This is the output after running the python script.

```
2024-10-27 20:13:51,347 - INFO - Subtract operation performed: 10 - 5 = 5. This logs the result of subtracting two numbers.
```

The third file is a simple multiplication script.

File Three:

```
import logging

# Configure logging to include timestamps and log level
logging.basicConfig(filename='LogThree.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

def multiply(a, b):
    try:
        # Perform the multiplication
        result = a * b
        # Log the operation with a clear comment
        logging.info(f"Multiply operation performed: {a} * {b} = {result}. This logs the result of
multiplying two numbers.")
```

```
        return result
    except Exception as e:
        # Log the exception with a clear context
        logging.exception("Error in multiply method: An exception occurred while attempting to
multiply values.")
        return None

# Example call to the multiply function
if __name__ == "__main__":
    multiply(6, 7)  # This should generate a log entry
```

This is the log output given for running this script.

```
2024-10-28 08:16:35,435 - INFO - Multiply operation performed: 6 * 7 = 42. This logs the result of multiplying two numbers.
```

The fourth file is a simple division script.

File Four:

```
import logging

# Configure logging to include timestamps and log level
logging.basicConfig(filename='FileFour.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

def divide(a, b):
    try:
        if b == 0:
            logging.warning("Attempted to divide by zero. Division by zero is not allowed.")
            return None
        # Perform the division
        result = a / b
        # Log the operation with a clear comment
        logging.info(f"Divide operation performed: {a} / {b} = {result}. This logs the result of dividing
two numbers.")
        return result
    except Exception as e:
        # Log the exception with a clear context
        logging.exception("Error in divide method: An exception occurred while attempting to divide
values.")
        return None

# Example call to the divide function
if __name__ == "__main__":
    divide(10, 2)  # This should generate a log entry
    divide(10, 0)  # This will trigger the warning for division by zero
```

These are the comments from logging below.

```
2024-10-28 08:19:23,442 - INFO - Divide operation performed: 10 / 2 = 5.0. This logs the result of dividing two numbers.
2024-10-28 08:19:23,442 - WARNING - Attempted to divide by zero. Division by zero is not allowed.
```

The fifth file is a simple add file script.

File Five:

```
import logging

# Configure logging to include timestamps and log level
logging.basicConfig(filename='FileFive.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

def add_to_file(file_path, content):
    try:
        # Open the file in append mode and write the content
        with open(file_path, 'a') as file:
            file.write(content + '\n')
            # Log the successful addition of content
            logging.info(f"Added content to {file_path}: {content}. This logs the successful write operation.")
    except Exception as e:
        # Log the exception with a clear context
        logging.exception(f"Error adding to file {file_path}: An exception occurred while writing to the file.")

# Example call to the add_to_file function
if __name__ == "__main__":
    add_to_file('example.txt', 'This is a test line.')  # This should generate a log entry
```

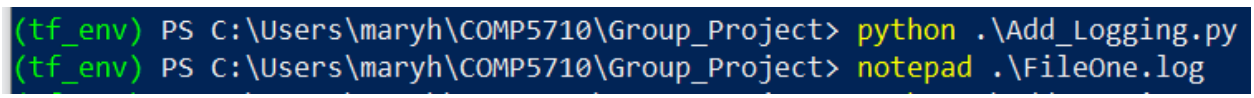
These are the comments given by the logging below.

```
2024-10-28 08:21:36,581 - INFO - Added content to example.txt: This is a test line.. This logs the successful write operation.
```

Below I have given screenshots of the commands I used to run and then open the log file for each script below.

Commands used:

For each file I ran the same commands to run the file and open the log file created when the python file ran correctly.



```
(tf_env) PS C:\Users\maryh\COMP5710\Group_Project> python .\Sub_Logging.py
(tf_env) PS C:\Users\maryh\COMP5710\Group_Project> notepad .\FileTwo.log
```

```
(tf_env) PS C:\Users\maryh\COMP5710\Group_Project> python .\Multi_Logging.py
(tf_env) PS C:\Users\maryh\COMP5710\Group_Project> notepad .\FileThree.log
```

```
(tf_env) PS C:\Users\maryh\COMP5710\Group_Project> python .\Div_Logging.py
(tf_env) PS C:\Users\maryh\COMP5710\Group_Project> notepad .\FileFour.log
```

```
(tf_env) PS C:\Users\maryh\COMP5710\Group_Project> python .\AddFile_Logging.py
(tf_env) PS C:\Users\maryh\COMP5710\Group_Project> notepad .\FileFive.log
```