Class Exercise:

Threading v1

In this exercise, you will implement a program that performs concurrent file processing using threading and synchronization in Python. The program will read multiple files concurrently, process their contents, and store the results in a shared data structure. To ensure thread safety and prevent conflicts, you will use synchronization techniques, taking into account the Global Interpreter Lock (GIL) limitations.

Instructions:

1. Create a Python class called FileProcessor that has the following attributes and methods:

* Attribute: results (a list) - stores the processed results.
* Method: process\_file(filename) - reads and processes the contents of a file. The processing can be as simple as counting the number of lines in the file and appending the result to the results list.

1. Create a function called worker that takes a FileProcessor object and a filename as parameters. This function will simulate the behavior of a thread processing a file. Inside the function, do the following:

* Acquire a lock to ensure exclusive access to the shared data structure (results list).
* Call the process\_file method of the FileProcessor object to process the file.
* Print a message indicating the thread ID and the filename being processed.
* Release the lock.

1. In the main part of your code, do the following:

* Create an instance of the FileProcessor class.
* Create a list of filenames that you want to process concurrently.
* Create multiple threads (at least 3) that will execute the worker function, passing the FileProcessor object and a filename from the list as parameters.
* Start the threads and wait for them to complete.

1. After all the threads have finished processing, print the contents of the results list to verify that the files were processed correctly.

Instructions:

import threading

class FileProcessor:

# Your code here

    def process\_file(self, filename):

# Your code here

def worker(processor, filename):

# Your code here

def main():

# Your code here

if \_\_name\_\_ == '\_\_main\_\_':

    main()

Expected output:

Thread 1234 processed file: file1.txt

Thread 2345 processed file: file2.txt

Thread 3456 processed file: file3.txt

Processed results:

File: file1.txt, Lines Count: 10

File: file2.txt, Lines Count: 8

File: file3.txt, Lines Count: 15