### Assignment Title: Asynchronous Data Fetch and Save

#### Objective:

Develop a Python application that fetches data from a public API asynchronously and saves the fetched data into a file using multi-threading. Make sure to check the sample code at the end and start working from that sample code and add your own modifications.

#### Tools and Libraries:

* **Python 3.8+**
* **asyncio library**: For asynchronous programming.
* **aiohttp library**: For making asynchronous HTTP requests.
* **threading library**: For writing data to the file in a separate thread.

#### Requirements:

1. **API Interaction**:
   * Use the aiohttp client to fetch data from the [JSONPlaceholder](https://jsonplaceholder.typicode.com/) API, specifically the /posts endpoint.
2. **Data Processing**:
   * Process the data fetched from the API to format or filter as required (e.g., select specific fields).
3. **Asynchronous Execution**:
   * Implement the API fetching logic asynchronously using asyncio.
4. **Multi-threaded File Writing**:
   * Use a separate thread to save the processed data to a file, ensuring that file writing does not block the fetching of data.
5. **Error Handling**:
   * Implement robust error handling to manage potential issues like network errors or file I/O errors.

#### Step-by-Step Instructions:

1. **Setup the Project**:
   * Install Python 3.8 or higher.
   * Install required packages: pip install aiohttp.
2. **Understanding Asynchronous Programming**:
   * Read about asyncio and asynchronous programming in Python from the [official asyncio documentation](https://docs.python.org/3/library/asyncio.html).
   * Explore aiohttp for making asynchronous HTTP requests through their [official documentation](https://docs.aiohttp.org/).
3. **Coding the Solution**:
   * Write a coroutine fetch\_data that uses aiohttp to asynchronously fetch data from the API.
   * Write a function write\_data that will be run in a separate thread to write data to a file.
   * Use asyncio to manage asynchronous tasks and threading to handle file writing.
4. **Testing and Validation**:
   * Test the program in different scenarios (e.g., with network issues, with large data sets).
   * Ensure the program does not block the main thread while writing to the file.
5. **Submission Guidelines**:
   * Submit a single Python script (your file name should be called “Studentnumber-assignment-x.py – where x is the number of the assignment; for example, if your student id is c01234567 and the assignment is assignment 1, the filename should be “c01234567-assignment-1.py”
   * Use comments to explain your code no extra file should be submitted.

#### Resources for Learning:

* [AsyncIO Official Documentation](https://docs.python.org/3/library/asyncio.html)
* [AIOHTTP Documentation](https://docs.aiohttp.org/)
* [Threading in Python](https://docs.python.org/3/library/threading.html)
* [Python Async/Await Tutorial](https://realpython.com/async-io-python/)

# Sample Code to be completed by you!

import asyncio

import aiohttp # Ensure aiohttp is installed using pip

# Asynchronous function to fetch data from the API

async def fetch\_data(url):

print("Fetching data from API...")

async with aiohttp.ClientSession() as session: # Managing the session

async with session.get(url) as response: # Asynchronous GET request

assert response.status == 200

data = await response.json() # Asynchronously read the response as JSON

return data

# Placeholder function for data processing

def process\_data(data):

print("Processing data...")

# Placeholder: Process data as needed before saving

return data

# Placeholder function for writing data to a file using threading

def write\_data\_to\_file(data):

print("Writing data to file...")

# TODO: Implement file writing in a separate thread

# This function should be executed in a separate thread to avoid blocking the asyncio event loop

async def main():

url = "https://jsonplaceholder.typicode.com/posts"

data = await fetch\_data(url)

processed\_data = process\_data(data)

# TODO: Call write\_data\_to\_file in a new thread

# Example: threading.Thread(target=write\_data\_to\_file, args=(processed\_data,)).start()

if \_\_name\_\_ == "\_\_main\_\_":

# Running the main function using asyncio

asyncio.run(main())