1IEE14 - Laboratorio 6

Instrucciones para el laboratorio:

- Materiales permitidos: Wiki del curso, apuntes de clase, consultar foros, tutoriales o documentación de Python online.
- Está prohibido el uso de cualquier modelo de lenguaje como ChatGPT o usar Github Copilot. A cualquier alumno que se le detecte que ha consultado un modelo de lenguaje se le pondrá nota 0(cero) en el laboratorio.
- Usted debe subir a Paideia un solo archivo comprimido (.zip o .rar) con el nombre L6_CODIGOPUCP.zip o L6_CODIGOPUCP.rar el cual contendrá sus archivos de Python para cada pregunta. En caso se le pida responder de manera teórica o incluir capturas de pantalla, deberá adjuntar un documento PDF con sus respuestas.
- El horario máximo permitido para subir el archivo es a las 10:00:00 pm. Pasada esa hora, habrá una penalidad de 2 puntos por cada minuto extra que se demore en entregar su archivo.

Pregunta 1: Control Concurrente de Sensores (5 pts.)

Simular 10 sensores que generan datos simultáneamente y calcular estadísticas de esos datos.

a) (2 pts.) Escriba un programa que lance 10 hilos. Cada hilo simula un sensor generando 10 000 valores **float** aleatorios entre 0.0 y 100.0 y los almacena en una lista compartida de forma segura (utilizar **threading.Lock**).

b) (2 pts.) Usando dos hilos adicionales, calcule en paralelo la media y la desviación estándar de la lista valores, almacenando resultados en variables compartidas (sugerencia: utilizar la librería **statistics**).

c) (1 pt.) Imprima en la consola y guarde en resultados.txt los valores de media y desv con 4 decimales.

Salida esperada:

```
joant@LAPTOP-KAHN9ACA:~/lab6/preg1$ python3 preg1.py
Media: 50.0234
Desviación: 28.8328
```

Pregunta 2: Eventos Asincrónicos con Registros Temporales (5 pts)

Registrar eventos aleatorios de forma no bloqueante y persistir sus datos.

a) (2 pts) Implemente una función **async def genera_evento(id)** que espere un tiempo aleatorio entre 1 y 4 segundos, luego devuelva una tupla (id, delay, timestamp) donde **timestamp** es la hora actual usando **datetime.datetime.now().isoformat()**.

b) (2 pts) Lance concurrentemente 5 eventos con **asyncio.gather**, recoja los resultados y escríbalos en eventos.csv con columnas: evento,delay,timestamp.

Salida esperada:

eventos.csv

```
1  evento,delay,timestamp
2  0,1,2025-05-15T14:53:29.628005
3  1,1,2025-05-15T14:53:29.628032
4  2,2,2025-05-15T14:53:30.641078
5  3,4,2025-05-15T14:53:32.641638
6  4,4,2025-05-15T14:53:32.641674
7
```

c) (1 pt) Después de asyncio.run, imprima el tiempo total de ejecución usando **time.perf_counter()** en la consola.

Salida esperada:

```
joant@LAPTOP-KAHN9ACA:~/lab6/preg2$ python3 preg2.py
Tiempo total de registro: 4.00s
```

Pregunta 3: Procesamiento Concurrente de Archivos de Sensores (5 pts)

En una aplicación de electrónica, disponer de múltiples archivos CSV con datos de sensores es común. Se proveen cinco archivos: sensor0.csv a sensor4.csv, cada uno con miles de lecturas numéricas (una lectura por línea).

a) (2 pts) Con **threading**, cree un hilo por cada archivo que calcule la suma de todas las lecturas en su archivo. Al finalizar, imprima la suma por archivo y la suma total de todas las lecturas Calcule e imprima el tiempo de ejecución (sugerencia: utilizar **time.perf_counter()**).

Salida esperada:

```
joant@LAPTOP-KAHN9ACA:~/lab6/preg3$ python3 preg3a.py
sensor1.csv: 249636.28879999966
sensor0.csv: 249300.33350000007
sensor3.csv: 245420.57599999962
sensor4.csv: 251946.35599999924
sensor2.csv: 248484.75630000036
Total de todas las lecturas: 1244788.310599999
Threading tiempo: 0.0056s
```

b) (2 pts) Con **asyncio**, use **asyncio.to_thread** para paralelizar la misma tarea de procesamiento de archivos. Al finalizar, imprima los mismos resultados. Calcule e imprima el tiempo de ejecución.

Salida esperada:

```
joant@LAPTOP-KAHN9ACA:~/lab6/preg3$ python3 preg3b.py
sensor0.csv: 249300.33350000007
sensor1.csv: 249636.28879999966
sensor2.csv: 248484.75630000036
sensor3.csv: 245420.57599999962
sensor4.csv: 251946.35599999924
Total de todas las lecturas: 1244788.310599999
Asyncio tiempo: 0.0054s
```

c) (1 pt) Guarde los resultados en un archivo JSON **sensors_summary.json** con el siguiente formato:

sensors_summary.json

```
{
  "sensor0.csv": 249300.33350000007,
  "sensor1.csv": 249636.28879999966,
  "sensor2.csv": 248484.75630000036,
  "sensor3.csv": 245420.57599999962,
  "sensor4.csv": 251946.35599999924,
  "total": 1244788.310599999
}
```