# Open Source Software Lab

**Lab Test 1**

**Monday – 3 to 5 PM**

Time Duration: 50 Minutes Maximum Marks: 20Marks

Note:

- No extra time will be provided for form submissions. Any responses submitted after the deadline will not be accepted.
- Please create a Word document with your answers, along with screenshots of the output. Upload a word file on Google Classroom which contains the following:
  - Link to your GitHub account
  - Codes for questions 1 along with the URL of the repository
- Save your file using the following format: (Batch_Enrollment_StudentName_Evaluation_1.docx)

**Odd**

**Q1. [CO 2, 20 Marks]** Analyse air quality fluctuations in a high-traffic urban area over a 24-hour period.

1. Generate a NumPy array called pollution_data to simulate PM2.5 (particulate matter) levels for 1440 minutes (one entry per minute). Add random noise to mimic sensor inconsistencies.
2. Apply a low-pass filter from scipy.signal to reduce the high-frequency noise in the PM2.5 data. Write the Python code to accomplish this.
3. Write a function to compute and display the average PM2.5 levels for each hour using NumPy. Print the hourly averages.
4. Plot the original noisy data and the filtered data on the same graph using matplotlib. Use different line styles to distinguish between the two. On the same plot, mark any hours where the average PM2.5 level exceeded a dangerous threshold (e.g., 150). Use a separate color and marker for these points to indicate hazardous pollution levels.
5. Write code to detect and plot any time intervals where PM2.5 levels exceeded 200 for more than 10 consecutive minutes. Highlight these peaks on the graph using distinct markers.

**Even**

**Q1. [CO 2, 20 Marks]** You are tasked with monitoring vehicle traffic on a major highway. Your goal is to analyze traffic data collected every minute over a 24-hour period, simulate real-world noise, and detect congestion.

1. Generate a NumPy array called vehicle_count to simulate traffic flow for 1440 minutes, where each entry represents the number of vehicles passing in that minute. Add random noise to represent real-world sensor inaccuracies.
2. Write Python code to apply a low-pass filter from scipy.signal to reduce noise in the data.
3. Compute and print the average number of vehicles passing each hour using NumPy. Store these averages in a separate array.
4. Create a plot that displays the noisy data, the smoothed data, and the hourly averages. Use distinct markers to highlight the hourly averages.
5. Write a function to detect periods of heavy traffic where the vehicle count exceeds 120 vehicles per minute for longer than 15 consecutive minutes. Highlight these congestion intervals on the graph using a different color or markers.