**Day 3 – Lab Document**

**Track**: Python for Data Engineering (Part 2)  
**Topics Covered**: Regex, Lambdas, JSON/XML Handling, REST APIs, Async I/O, Structured Logging  
**Lab Count**: 3 Labs  
**Duration**: 3+ hours (spread across multiple slots with breaks and lectures)

**Lab 1: Regex, Filtering, and Lambda Operations**

**Objective**

Understand and apply regular expressions and lambda functions to clean and transform textual data such as emails and phone numbers.

**Background**

In real-world data pipelines, input data often contains inconsistencies such as malformed emails or phone numbers. This lab simulates typical data cleansing steps using regex and inline lambda functions.

**Setup**

* Use a Python virtual environment.
* Install required modules (if not already available):

bash

pip install regex

**Tasks**

1. Create a list of customer dictionaries with fields: name, email, and phone.
2. Use **regex** to:
   * Validate email addresses with pattern:  
     ^[\w\.-]+@[\w\.-]+\.\w{2,4}$
   * Extract valid Indian mobile numbers (10 digits starting with 7/8/9).
3. Use **lambda functions** and **list comprehensions** to:
   * Convert customer names to title case.
   * Filter customers with email domains like @example.com.
4. Count and display:
   * Number of valid and invalid emails.
   * Number of valid and invalid phone numbers.

**Expected Output**

* Cleaned list of customer records.
* Summary printout with counts.
* Console output of valid/invalid entries.

**Deliverable**

* Python script: lab1\_regex\_lambdas.py

**Lab 2: API Integration and JSON/XML Transformation**

**Objective**

Interact with public REST APIs using Python, parse both JSON and XML responses, and store them in structured formats.

**Background**

Working with external APIs and transforming the received data into usable formats like JSON or DataFrames is critical in building ingestion and transformation pipelines.

**Setup**

* Use requests for API calls.
* Use json for JSON manipulation.
* Use xml.etree.ElementTree for XML parsing.

**Tasks**

**Part A: JSON API**

1. Use the public API https://jsonplaceholder.typicode.com/posts
2. Fetch the first 20 post records.
3. Parse and extract the following fields:
   * userId, title, body
4. Store the result as a list of dictionaries and write to posts.json.

**Part B: XML to JSON**

1. Create a sample XML blog feed file or use an online one.
2. Parse the XML using ElementTree.
3. Convert it to a dictionary format with key fields like title, author, pubDate.
4. Write the structured output to a file named posts.xml.json.

**Expected Output**

* JSON file with 20 records from REST API.
* Transformed JSON from XML input.
* Clean, formatted, and human-readable file contents.

**Deliverables**

* lab2\_api\_to\_json.py
* posts.json
* posts.xml.json

**Lab 3: Async API Fetching and Structured Logging**

**Objective**

Perform asynchronous REST API calls using aiohttp or httpx, and log results with structured metadata using Python’s logging and json modules.

**Background**

Modern data pipelines often require high-concurrency API ingestion. Asynchronous programming significantly improves performance when fetching from multiple endpoints. Structured logging ensures operational visibility.

**Setup**

* Install dependencies:

bash

pip install aiohttp

**Tasks**

1. Fetch the following endpoints asynchronously:
   * /posts
   * /comments
   * /users  
     All from base URL: https://jsonplaceholder.typicode.com
2. Use asyncio.gather() to run all fetches concurrently.
3. For each request, log the following:
   * Timestamp
   * API endpoint
   * HTTP status code
   * Number of records fetched
   * Error message if any
4. Use the logging module configured to output in JSON Lines format (.jsonl).
5. Write logs to api\_log.jsonl.

**Expected Output**

* API responses fetched in parallel.
* Log file with line-by-line JSON log entries including status and metadata.
* Proper error handling in case of connection or response failures.

**Deliverables**

* lab3\_async\_logging.py
* api\_log.jsonl

**Checklist**

| **File Name** | **Description** |
| --- | --- |
| lab1\_regex\_lambdas.py | Regex and lambda-based data cleanup script |
| lab2\_api\_to\_json.py | Script for REST API and XML parsing |
| posts.json | Output from JSON API |
| posts.xml.json | Output from XML-to-JSON transformation |
| lab3\_async\_logging.py | Async API calls and structured logging |
| api\_log.jsonl | Structured logs from API fetch run |