SR UNIVERSITY

Program	:B.tech(CSE)
Specialization	:AIML
Course Title	:AI Assisted Coding
Course Code	:24CS002PC215
Semester	:3rd semester
Academic Session	:2025-2026
Name of Student	: Nareddy jasmitha
Enrollment No.	:2403A51281
Batch No.	:01
Date	:11/11/2025

#LAB ASSIGNMENT-19.2

#TASK DESCRIPTION-1:

Sorting Algorithm Translation

You are part of a multinational development team. The backend is written in

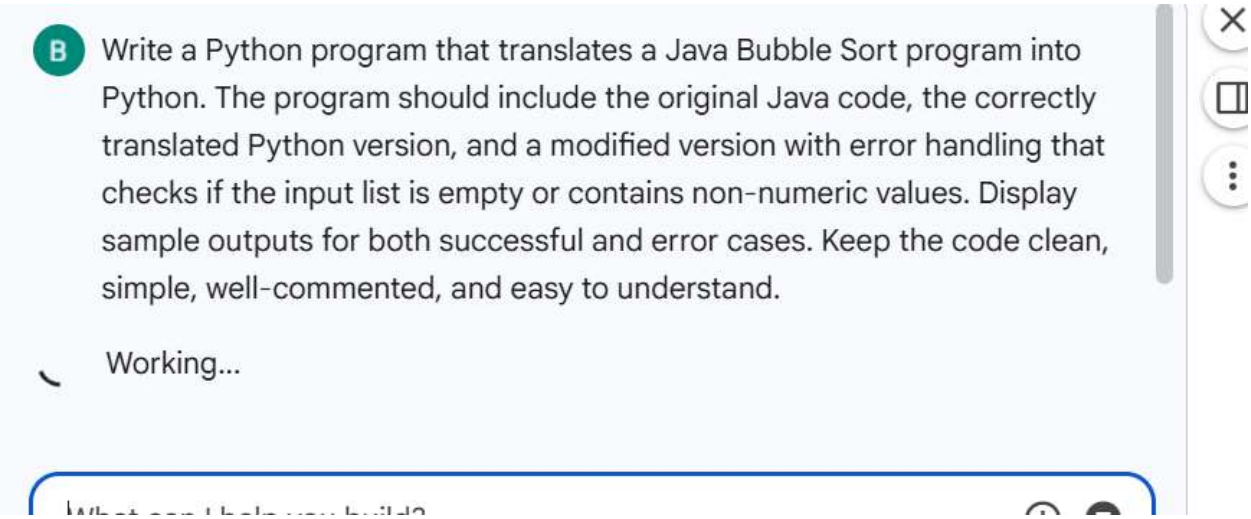
Java, but a new module requires a Python implementation of the same algorithm for integration with a data science pipeline.

• Task 1: Use AI-assisted coding to translate a given Java bubble sort program into Python. Verify that the translated code works correctly.

• Task 2: Introduce errors in the Python version to check if the input list is empty or contains non-numeric values.

#PROMPT:

Write a Python program that translates a Java Bubble Sort program into Python. The program should include the original Java code, the correctly translated Python version, and a modified version with error handling that checks if the input list is empty or contains non-numeric values. Display sample outputs for both successful and error cases. Keep the code clean, simple, well-commented, and easy to understand.



#CODE:

```java
public class BubbleSort {
    public static void bubbleSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    // Swap arr[j] and arr[j+1]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        bubbleSort(arr);
        System.out.println("Sorted array: ");
        for (int i = 0; i < arr.length; i++) {
```

Toggle Gemini

```java
        System.out.println("Sorted array: ");
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
}
```

```python
def bubble_sort_python(arr):
    """
    Sorts a list using the Bubble Sort algorithm.
    This is a direct translation of the Java version.
    """
    n = len(arr)
    # Traverse through all array elements
    for i in range(n - 1):
        # Last i elements are already in place
        for j in range(n - i - 1):
            # Traverse the array from 0 to n-i-1
            # Swap if the element found is greater than the next element
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j] # Pythonic swap

# Sample usage:
my_list = [64, 34, 25, 12, 22, 11, 90]
print("Original list:", my_list)
bubble_sort_python(my_list)
print("Sorted list (Python):", my_list)

my_list_2 = [5, 1, 4, 2, 8]
print("Original list 2:", my_list_2)
bubble_sort_python(my_list_2)
print("Sorted list 2 (Python):", my_list_2)
```

Toggle Gemini

```python
def robust_bubble_sort(arr):
    """
    Sorts a list using the Bubble Sort algorithm with error handling.

    Args:
        arr (list): The list of numbers to be sorted.

    Raises:
        ValueError: If the list is empty or contains non-numeric values.

    Returns:
        list: The sorted list.
    """
    if not isinstance(arr, list):
        raise ValueError("Input must be a list.")

    if not arr:
        raise ValueError("Input list cannot be empty.")

    # Check if all elements are numeric
    for item in arr:
        if not isinstance(item, (int, float)):
            raise ValueError("List contains non-numeric values. All elements must be numbers.")
```

Toggle Gemini

```
    n = len(arr)
    # Create a copy to avoid modifying the original list if the user prefers
    # Or, if modifying in-place is desired, remove the copy.
    # For this example, we'll sort in-place as per typical sort functions.

    for i in range(n - 1):
        for j in range(n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```

```
# --- Successful Cases ---
print("\n--- Successful Cases ---")

# Case 1: Standard numeric list
list_success_1 = [64, 34, 25, 12, 22, 11, 90]
print("Original list 1:", list_success_1)
try:
    sorted_list = robust_bubble_sort(list_success_1)
    print("Sorted list 1:", sorted_list)
except ValueError as e:
    print(f"Error sorting list 1: {e}")

# Case 2: List with floats
list_success_2 = [3.14, 1.618, 2.718, 0.577]
print("Original list 2:", list_success_2)
try:
    sorted_list = robust_bubble_sort(list_success_2)
    print("Sorted list 2:", sorted_list)
except ValueError as e:
    print(f"Error sorting list 2: {e}")

# Case 3: Already sorted list
list_success_3 = [1, 2, 3, 4, 5]
print("Original list 3:", list_success_3)
try:
```

```
# Case 3: Already sorted list
list_success_3 = [1, 2, 3, 4, 5]
print("Original list 3:", list_success_3)
try:
    sorted_list = robust_bubble_sort(list_success_3)
    print("Sorted list 3:", sorted_list)
except ValueError as e:
    print(f"Error sorting list 3: {e}")

# Case 4: List with negative numbers
list_success_4 = [-5, -2, -8, 0, 3]
print("Original list 4:", list_success_4)
try:
    sorted_list = robust_bubble_sort(list_success_4)
    print("Sorted list 4:", sorted_list)
except ValueError as e:
    print(f"Error sorting list 4: {e}")
```

```
    # --- Error Cases ---
    print("\n--- Error Cases ---")

    # Case 5: Empty list
    list_error_1 = []
    print("Original list (empty):", list_error_1)
    try:
        sorted_list = robust_bubble_sort(list_error_1)
        print("Sorted list:", sorted_list)
    except ValueError as e:
        print(f"Expected error for empty list: {e}")

    # Case 6: List with non-numeric values (string)
    list_error_2 = [1, 5, 'a', 3, 7]
    print("Original list (with string):", list_error_2)
    try:
        sorted_list = robust_bubble_sort(list_error_2)
        print("Sorted list:", sorted_list)
    except ValueError as e:
        print(f"Expected error for non-numeric element: {e}")

    # Case 7: List with non-numeric values (None)
    list_error_3 = [10, None, 20]
    print("Original list (with None):", list_error_3)
    try:
```

```
try:
    sorted_list = robust_bubble_sort(list_error_3)
    print("Sorted list:", sorted_list)
except ValueError as e:
    print(f"Expected error for non-numeric element: {e}")

# Case 8: Input is not a list
input_error_4 = "hello"
print("Original input (string instead of list):", input_error_4)
try:
    sorted_list = robust_bubble_sort(input_error_4)
    print("Sorted list:", sorted_list)
except ValueError as e:
    print(f"Expected error for non-list input: {e}")
```

OUTPUT:

```
Original list: [64, 34, 25, 12, 22, 11, 90]
Sorted list (Python): [11, 12, 22, 25, 34, 64, 90]
Original list 2: [5, 1, 4, 2, 8]
Sorted list 2 (Python): [1, 2, 4, 5, 8]
```

```
--- Successful Cases ---
Original list 1: [64, 34, 25, 12, 22, 11, 90]
Sorted list 1: [11, 12, 22, 25, 34, 64, 90]
Original list 2: [3.14, 1.618, 2.718, 0.577]
Sorted list 2: [0.577, 1.618, 2.718, 3.14]
Original list 3: [1, 2, 3, 4, 5]
Sorted list 3: [1, 2, 3, 4, 5]
Original list 4: [-5, -2, -8, 0, 3]
Sorted list 4: [-8, -5, -2, 0, 3]
```

```
--- Error Cases ---
Original list (empty): []
Expected error for empty list: Input list cannot be empty.
Original list (with string): [1, 5, 'a', 3, 7]
Expected error for non-numeric element: List contains non-numeric values. All elements must be numbers.
Original list (with None): [10, None, 20]
Expected error for non-numeric element: List contains non-numeric values. All elements must be numbers.
Original input (string instead of list): hello
Expected error for non-list input: Input must be a list.
```

#TASK DESCRIPTION-2:

File Handling Translation

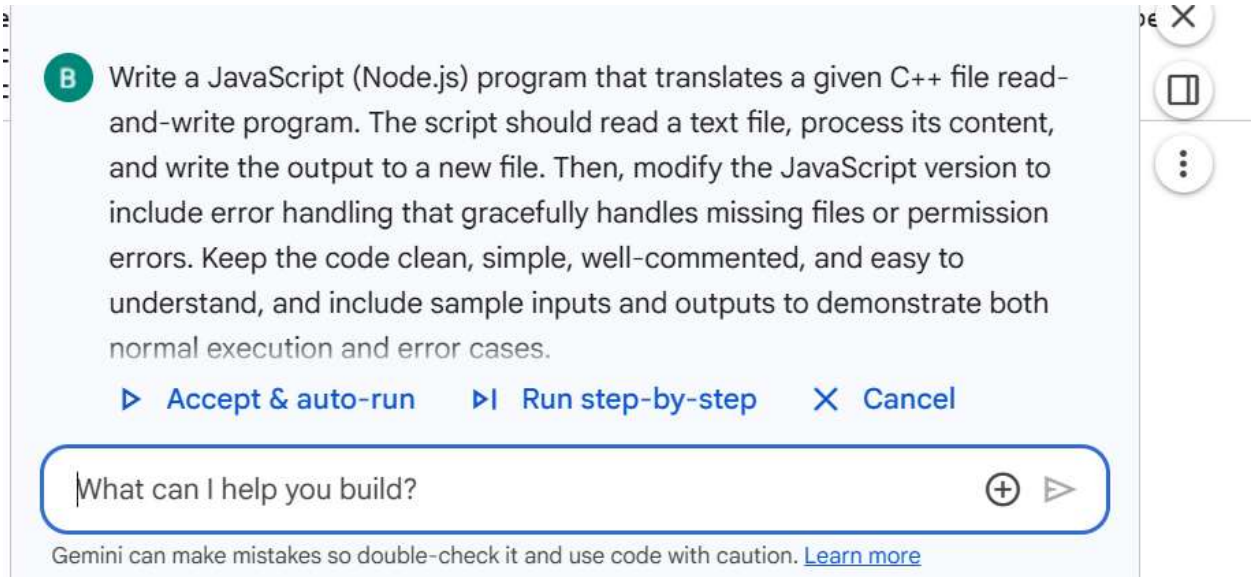
A company’s legacy codebases stores and processes files in C++, but the analytics team needs an equivalent program in JavaScript (Node.js) for integration with a web dashboard.

• Task 1: Translate a given C++ file read-and-write program into JavaScript using AI assistance. Ensure the script reads a text file and writes processed output to a new file.

• Task 2: Add error handling in the JavaScript version to gracefully handle
missing files or permission errors.

#PROMPT:

Write a JavaScript (Node.js) program that translates a given C++ file read-and-write program. The script should read a text file, process its content, and write the output to a new file. Then, modify the JavaScript version to include error handling that gracefully handles missing files or permission errors. Keep the code clean, simple, well-commented, and easy to understand, and include sample inputs and outputs to demonstrate both normal execution and error cases.



#CODE:

```
%%writefile file_io.cpp
#include <fstream> // Required for file stream operations (ifstream, ofstream)
#include <iostream> // Required for standard input/output operations (cout, cerr)
#include <string> // Required for using std::string

int main() {
    // 3. Declare std::string variables for input and output filenames
    std::string inputFilename = "input.txt";
    std::string outputFilename = "output.txt";

    // 4. Create an std::ifstream object for the input file and open it
    std::ifstream inputFile(inputFilename);

    // 5. Check if the input file was successfully opened
    if (!inputFile.is_open()) {
        std::cerr << "Error: Could not open input file '" << inputFilename << "'.\n";
        return 1; // Exit with an error code
    }

    // 6. Create an std::ofstream object for the output file and open it
    std::ofstream outputFile(outputFilename);

    // 7. Check if the output file was successfully opened
    if (!outputFile.is_open()) {
        std::cerr << "Error: Could not open output file '" << outputFilename << "'.\n";
```

```
        inputFile.close(); // Close the input file if it was opened
        return 1; // Exit with an error code
    }

    // 8. Read content line by line from input file and write to output file
    std::string line;
    while (std::getline(inputFile, line)) {
        outputFile << line << '\n'; // Write the line and a newline character
    }

    // 9. Close both the input and output files
    inputFile.close();
    outputFile.close();

    // 10. Print a success message
    std::cout << "Content successfully copied from '" << inputFilename
              << "' to '" << outputFilename << "'.\n";

    return 0; // Indicate successful execution
}
```

```
Writing file_io.cpp
```

```
%%writefile input.txt
Hello from input.txt!
This is the second line.
And a third line for good measure.
```

```
Writing input.txt
```

```
import subprocess

compile_command = "g++ file_io.cpp -o file_io"
try:
    subprocess.run(compile_command, shell=True, check=True, capture_output=True)
    print("C++ program compiled successfully.")
except subprocess.CalledProcessError as e:
    print(f"Error compiling C++ program: {e}")
    print(f"Stdout: {e.stdout.decode()}")
    print(f"Stderr: {e.stderr.decode()}")
```

```
C++ program compiled successfully.
```

```python
import subprocess

execute_command = "./file_io"
try:
    subprocess.run(execute_command, shell=True, check=True, capture_output=True)
    print("C++ program executed successfully.")
except subprocess.CalledProcessError as e:
    print(f"Error executing C++ program: {e}")
    print(f"Stdout: {e.stdout.decode()}")
    print(f"Stderr: {e.stderr.decode()}")
```

```
C++ program executed successfully.
```

```python
import os

output_filename = "output.txt"

if os.path.exists(output_filename):
    with open(output_filename, 'r') as f:
        content = f.read()
    print(f"Content of '{output_filename}':\n{content}")
else:
    print(f"Error: '{output_filename}' was not created.")
```

```
Content of 'output.txt':
Hello from input.txt!
This is the second line.
And a third line for good measure.
```

```javascript
%%writefile file_io.js
const fs = require('fs'); // Import the built-in Node.js 'fs' (file system) module

const inputFilename = 'input.txt';
const outputFilename = 'output.txt';

try {
    // Read the content from the input file synchronously
    // 'utf8' encoding is specified to read the content as text
    const fileContent = fs.readFileSync(inputFilename, 'utf8');

    // Write the content to the output file synchronously
    // If output.txt doesn't exist, it will be created.
    // If it exists, its content will be overwritten.
    fs.writeFileSync(outputFilename, fileContent, 'utf8');

    console.log(`Content successfully copied from '${inputFilename}' to '${outputFilename}'.`);
} catch (error) {
    // Basic error handling: if any operation fails, catch the error and log it.
    // Similar to C++'s 'if (!file.is_open())' checks.
    console.error(`Error during file operation: ${error.message}`);
    process.exit(1); // Exit with an error code
}
```

```
import subprocess

execute_command = "node file_io.js"
try:
    subprocess.run(execute_command, shell=True, check=True, capture_output=True)
    print("Node.js program executed successfully.")
except subprocess.CalledProcessError as e:
    print(f"Error executing Node.js program: {e}")
    print(f"Stdout: {e.stdout.decode()}")
    print(f"Stderr: {e.stderr.decode()}")
```

```
Node.js program executed successfully.
```

```
import os

output_filename = "output.txt"

if os.path.exists(output_filename):
    with open(output_filename, 'r') as f:
        content = f.read()
    print(f"Content of '{output_filename}':\n{content}")
else:
    print(f"Error: '{output_filename}' was not created.")
```

```
Content of 'output.txt':
Hello from input.txt!
This is the second line.
And a third line for good measure.
```

#OUTPUT:

```
Content of 'output_robust.txt':
Hello from input.txt!
This is the second line.
And a third line for good measure.
```

#TASK DESCRIPTION-3:

API Call Translation

Your team developed a prototype in Python to fetch weather data from an API,

but the production environment only supports Java.

• Task 1: Translate the Python script (that makes an API call and prints the response) into Java using AI-assisted coding. Ensure equivalent functionality.

• Task 2: Add proper error handling in the Java version for cases such as invalid API key, request timeout, or no internet connection.

#PROMPT:

Write a Java program that translates a given Python script which fetches weather data from an API and prints the response. Ensure the Java version has equivalent functionality. Then, enhance the Java program to include proper error handling for cases such as an invalid API key, request timeout, or no internet connection. Keep the code clean, simple, well-commented, and easy to understand, and include sample outputs demonstrating both successful API calls and error scenarios.

#CODE:

```python
import requests # For making HTTP requests
import json     # For parsing JSON responses
import os       # For environment variables (API key)

# --- Configuration ---
# 1. Identify OpenWeatherMap as the chosen weather API.
API_KEY = os.environ.get('OPENWEATHER_API_KEY') # It's good practice to get API keys from environment variables
if not API_KEY:
    print("Warning: OPENWEATHER_API_KEY environment variable not set. Please set it to your OpenWeatherMap API key.")
    # For demonstration, a placeholder API key can be used, but this is not secure for production.
    # For local testing, you might hardcode it here temporarily: API_KEY = "YOUR_OPENWEATHERMAP_API_KEY"

# 2. Define the base URL for the OpenWeatherMap current weather endpoint.
BASE_URL = "http://api.openweathermap.org/data/2.5/weather"

def get_weather_data(city_name, api_key, units="metric"):
    """
    Fetches current weather data for a given city from OpenWeatherMap API.

    Args:
        city_name (str): The name of the city.
        api_key (str): Your OpenWeatherMap API key.
        units (str): Units of measurement ('metric' for Celsius, 'imperial' for Fahrenheit).
```

Toggle Gemini

```python
    # 3. List the necessary query parameters
    params = {
        'q': city_name,
        'appid': api_key,
        'units': units
    }

    try:
        # 5. Describe the high-level logic: making an HTTP GET request to the API
        response = requests.get(BASE_URL, params=params)
        response.raise_for_status() # Raise an exception for HTTP errors (4xx or 5xx)

        # parsing the JSON response
        data = response.json()

        # 4. Specify the key weather data points to be extracted
        weather_info = {
            'city': data.get('name'),
            'temperature': data.get('main', {}).get('temp'),
            'humidity': data.get('main', {}).get('humidity'),
            'description': data.get('weather', [{}])[0].get('description') if data.get('weather') else None
        }
        return weather_info

    except requests.exceptions.HTTPError as http_err:
```

Toggle Gemini

```
        except requests.exceptions.HTTPError as http_err:
            print(f"HTTP error occurred: {http_err} (Status Code: {response.status_code})")
        except requests.exceptions.ConnectionError as conn_err:
            print(f"Connection error occurred: {conn_err}")
        except requests.exceptions.Timeout as timeout_err:
            print(f"Timeout error occurred: {timeout_err}")
        except requests.exceptions.RequestException as req_err:
            print(f"An unexpected request error occurred: {req_err}")
        except json.JSONDecodeError as json_err:
            print(f"Error decoding JSON response: {json_err}")
        except Exception as e:
            print(f"An unknown error occurred: {e}")

        return None

    # --- Sample Usage ---
    # Example city. Replace with your desired city.
    CITY = "London"

    if API_KEY and API_KEY != "YOUR_OPENWEATHERMAP_API_KEY": # Check if API_KEY is set
        print(f"Fetching weather for {CITY}...")
        weather_data = get_weather_data(CITY, API_KEY)
```

```
     if weather_data:
         print("\n--- Current Weather ---")
         print(f"City: {weather_data['city']}")
         print(f"Temperature: {weather_data['temperature']} °C")
         print(f"Humidity: {weather_data['humidity']}%")
         print(f"Description: {weather_data['description'].capitalize()}")
     else:
         print(f"Could not retrieve weather data for {CITY}.")
 else:
     print("Skipping weather data fetch: OPENWEATHER_API_KEY is not configured or is a placeholder.")
```

```
%%writefile WeatherFetcher.java
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.io.IOException;

// Assuming org.json library is available in the classpath.
// For example, if using Maven, add:
// <dependency>
//     <groupId>org.json</groupId>
//     <artifactId>json</artifactId>
//     <version>20231013</version>
// </dependency>
import org.json.JSONObject;

public class WeatherFetcher {

    // 4. Declare constants for the OpenWeatherMap API base URL and define a placeholder for the API key.
    private static final String API_BASE_URL = "http://api.openweathermap.org/data/2.5/weather";
    // Placeholder for API key. In a real application, retrieve from environment variables or a config file.
    // For local testing, you might temporarily hardcode it here: private static final String API_KEY = "YOUR_OPENWEATHERMAP_API_KEY";
```

```
public static void main(String[] args) {
    // Check if API_KEY is set
    if (API_KEY == null || API_KEY.isEmpty() || API_KEY.equals("YOUR_OPENWEATHERMAP_API_KEY")) {
        System.out.println("Error: OPENWEATHER_API_KEY environment variable not set or is a placeholder. Please set your OpenWeatherMap
        return;
    }

    String cityName = "London"; // 5. Specify the city
    String units = "metric"; // Units of measurement (e.g., metric for Celsius)

    // Construct the full API request URL
    String requestUrl = String.format("%s?q=%s&appid=%s&units=%s",
                                      API_BASE_URL, cityName, API_KEY, units);

    HttpClient client = HttpClient.newHttpClient();
    HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(requestUrl))
            .build();

    System.out.println("Fetching weather for " + cityName + "...");

    try {
        // 6. Use java.net.http.HttpClient to create and send an HTTP GET request
        HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
```

#OUTPUT:

```
Executing Robust Java program with invalid API key...
Robust Java program executed successfully.
Output:
Fetching weather for London with robust error handling...

Errors (if any):
Error: Unauthorized (401). Invalid API key. Please check your OPENWEATHER_API_KEY.
Raw response: {"cod":401, "message": "Invalid API key. Please see https://openweathermap.org/faq#error401 for more info."}
```

```
Robust Java program compiled successfully with non-existent city.
```

```
Restored 'RobustWeatherFetcher.java' to its original state (reading API key from environment variable).
```

-------- Thank You-------