

LABASSIGNMENT-19.2

NAME:B.SRINITHI

RoLLNO:2403A51413

COURSE:AIASSISTEDCODING

BATCH:01(AIML)

QUESTIONS:

Page < 2	
	<div>Lab Question 1: Sorting Algorithm Translation<p>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.</p><ul style="list-style-type: none">• 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.</div> <div>Lab Question 2: File Handling Translation<p>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.</p><ul style="list-style-type: none">• 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.</div> <div>Lab Question 3: API Call Translation<p>Your team developed a prototype in Python to fetch weather data from an API, but the production environment only supports Java.</p><ul style="list-style-type: none">• 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.</div>

LABQUESTION1:

PROMPT:

You are part of a multinational development team. The backend system uses Java, but a new module requires a Python implementation for integration with a data science pipeline.

Task1: Use AI-assisted coding to translate the following Java Bubble Sort program into Python and verify that it works correctly.

Task 2: Modify the Python version to include error handling that checks if the input list is empty or contains non-numeric values. If so, print an appropriate error message instead of sorting.

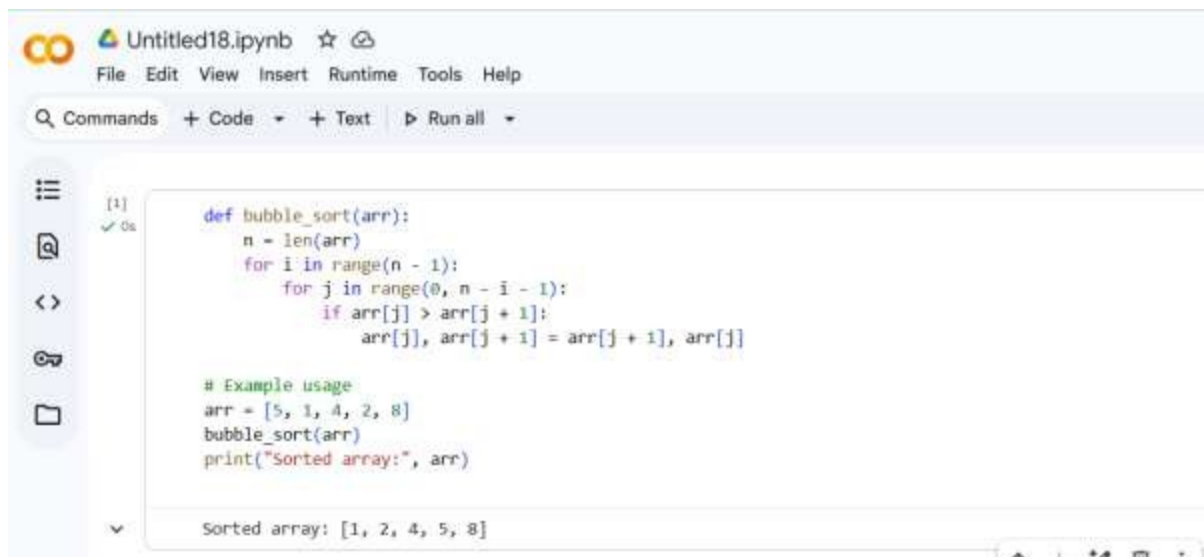
CODE:

```
public class BubbleSort {  
    public static void main(String[] args) {  
        int[] arr = {5, 1, 4, 2, 8};  
        bubbleSort(arr);  
        System.out.println("Sorted array:");  
        for (int num : arr) {  
            System.out.print(num + " ");  
        }  
    }  
}  
  
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]) {
            int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j+1] = temp;
        }
    }
}
}
}
}

```



The screenshot shows a Jupyter Notebook titled 'Untitled18.ipynb'. The interface includes a top menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu is a toolbar with 'Commands', '+ Code', '+ Text', and 'Run all'. On the left side, there is a sidebar with icons for file management and a list of cells. The main area displays a code cell with the following Python code:

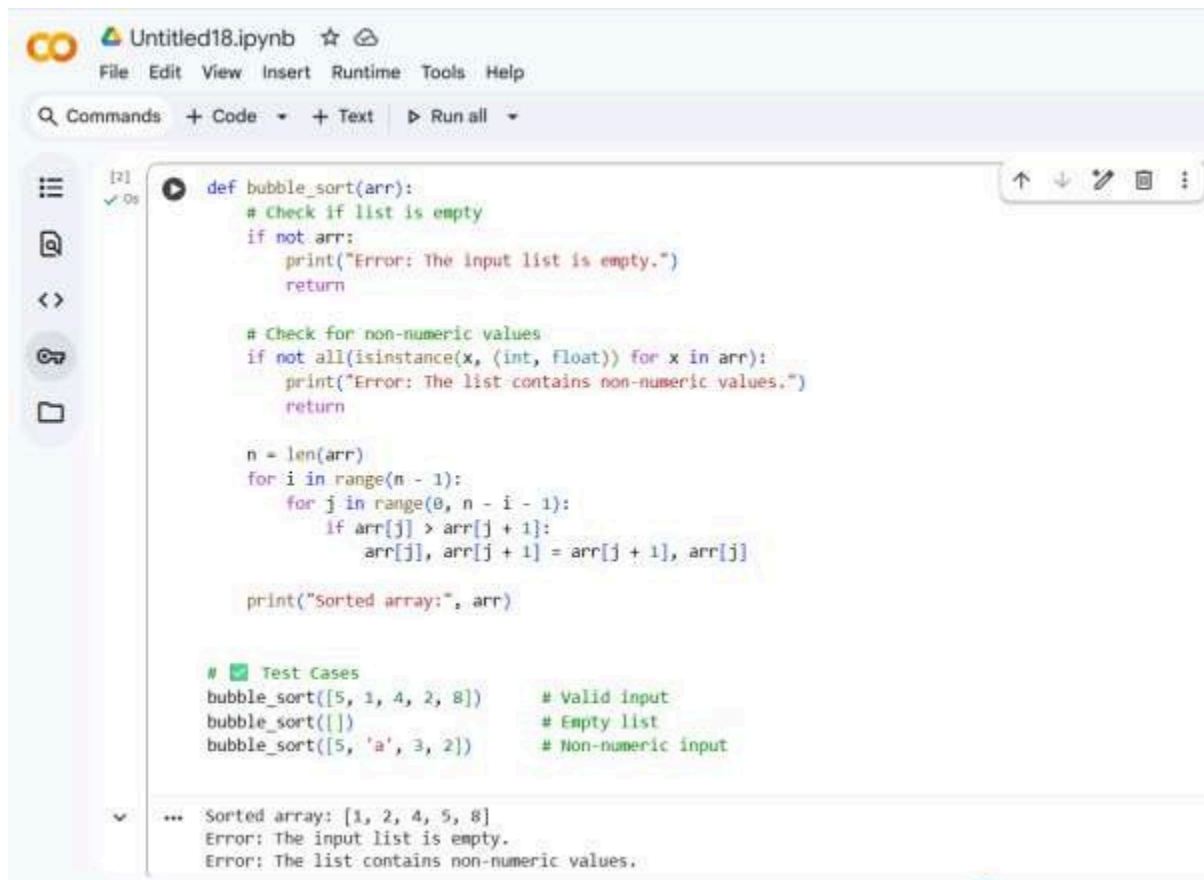
```

def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

# Example usage
arr = [5, 1, 4, 2, 8]
bubble_sort(arr)
print("Sorted array:", arr)

```

The output of the code cell is displayed at the bottom: 'Sorted array: [1, 2, 4, 5, 8]'.



The screenshot shows a Jupyter Notebook titled 'Untitled18.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with icons for commands, code, text, and running all cells. The notebook contains a Python function `bubble_sort(arr)` that sorts a list in ascending order. The function includes error handling for empty lists and non-numeric values. Below the function, there are test cases for valid input, empty list, and non-numeric input. The output of the notebook shows the sorted array `[1, 2, 4, 5, 8]` and the error messages for the invalid inputs.

```
def bubble_sort(arr):  
    # Check if list is empty  
    if not arr:  
        print("Error: The input list is empty.")  
        return  
  
    # Check for non-numeric values  
    if not all(isinstance(x, (int, float)) for x in arr):  
        print("Error: The list contains non-numeric values.")  
        return  
  
    n = len(arr)  
    for i in range(n - 1):  
        for j in range(0, n - i - 1):  
            if arr[j] > arr[j + 1]:  
                arr[j], arr[j + 1] = arr[j + 1], arr[j]  
  
    print("Sorted array:", arr)  
  
# Test Cases  
bubble_sort([5, 1, 4, 2, 8]) # Valid input  
bubble_sort([]) # Empty list  
bubble_sort([5, 'a', 3, 2]) # Non-numeric input
```

Output:

```
Sorted array: [1, 2, 4, 5, 8]  
Error: The input list is empty.  
Error: The list contains non-numeric values.
```

LABQUESTION2:

PROMPT:

You are part of a development team that is modernizing a company's legacy C++ system.

The old system reads a text file, processes its content, and writes the results to a new file.

Task1: Use AI-

assisted coding to translate the given C++ file read-and-write program into JavaScript (Node.js).

Ensure that the JavaScript version correctly reads data from a

text file and writes processed output (for example, uppercase text) into a new file.

Task2

Extend the JavaScript(Node.js) file-handling program by adding **error handling**.

The updated script should handle missing input files, invalid permissions, or any I/O errors gracefully, showing user-friendly messages without crashing the program.

CODE :

```
#include<iostream>
> #include
<fstream>
#include <string>
using namespace std;

int main(){
    ifstream inputFile("input.txt");
    ofstream outputFile("output.txt");
    string line;

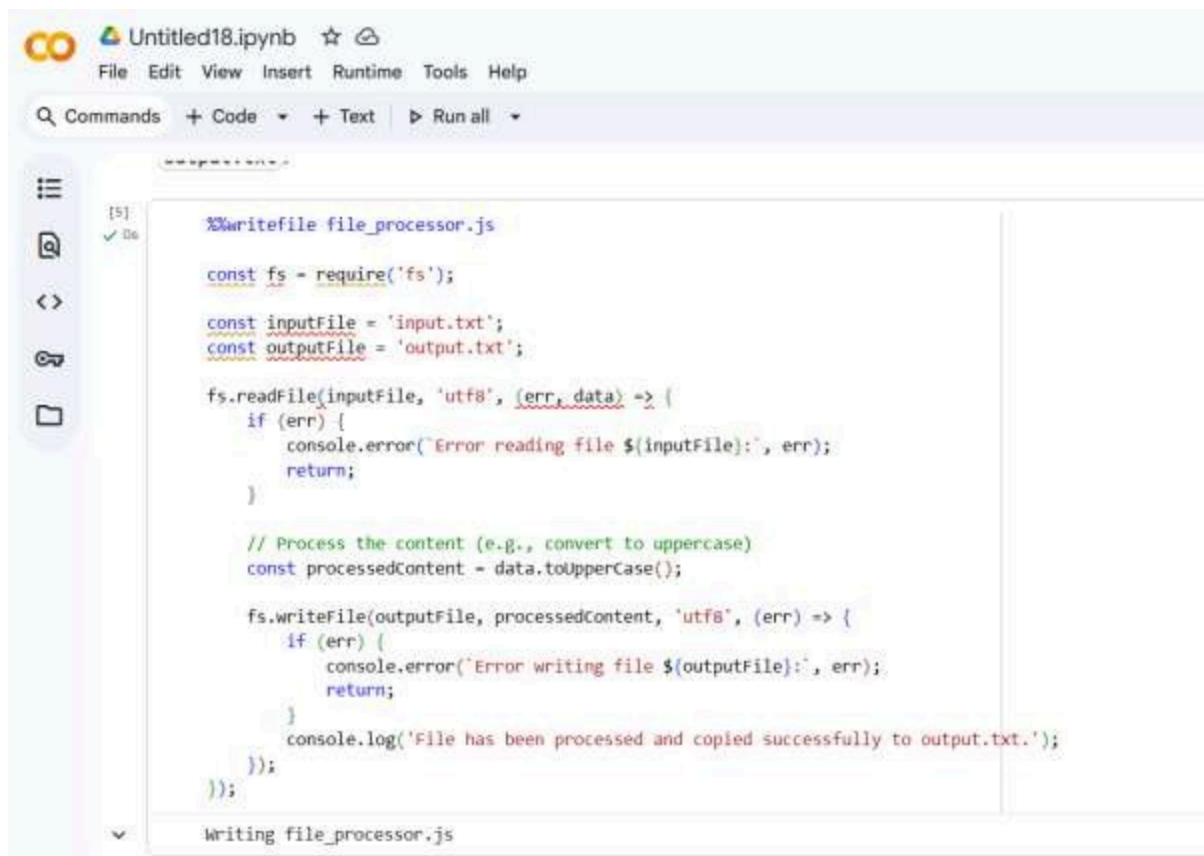
    if(inputFile.is_open() && outputFile.is_open()){ while
        ile (getline(inputFile, line)) {
            outputFile<<line<<endl;
        }
        cout<<"File has been copied successfully."<<endl;
    }else{
        cout<<"Error opening file."<<endl;
    }

    inputFile.close();
```

```
outputFile.close();
```

```
return 0;
```

```
}
```



The screenshot shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar (Commands, + Code, + Text, Run all). The notebook is titled "Untitled18.ipynb". The code cell contains the following JavaScript code:

```
%writefile file_processor.js

const fs = require('fs');

const inputFile = 'input.txt';
const outputFile = 'output.txt';

fs.readFile(inputFile, 'utf8', (err, data) => {
  if (err) {
    console.error('Error reading file ${inputFile}:', err);
    return;
  }

  // Process the content (e.g., convert to uppercase)
  const processedContent = data.toUpperCase();

  fs.writeFile(outputFile, processedContent, 'utf8', (err) => {
    if (err) {
      console.error('Error writing file ${outputFile}:', err);
      return;
    }
    console.log('File has been processed and copied successfully to output.txt.');
```

The code cell is titled "Writing file_processor.js".

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

First, let's create a sample `input.txt` file that our Node.js script will read from.

```
[4] ✓ Os
with open('input.txt', 'w') as f:
    f.write('Hello from C++ legacy system!\n')
    f.write('This is a test line.\n')
    f.write('Node.js will process this.')

print("Created 'input.txt' successfully.")
```

```
... Created 'input.txt' successfully.
```

Now, here is the JavaScript (Node.js) translation of your C++ program. It uses Node.js's built-in `fs` module to handle file operations. I've also added a step to convert the content to uppercase before writing it to `output.txt`.

```
[9] ✓ Os
%%writefile file_processor.js

const fs = require('fs');

const inputFile = 'input.txt';
const outputFile = 'output.txt';

// Function to handle errors gracefully with more specific messages
function handleError(err, operation, filePath) {
    let errorMessage = `Error during ${operation} of ${filePath}: `;
    if (err.code === 'ENOENT') {
        errorMessage += 'File not found. Please ensure the input file exists.';
    }
```

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[8] ✓ Os
// Function to handle errors gracefully with more specific messages
function handleError(err, operation, filePath) {
    let errorMessage = `Error during ${operation} of ${filePath}: `;
    if (err.code === 'ENOENT') {
        errorMessage += 'File not found. Please ensure the input file exists.';
    } else if (err.code === 'EACCES') {
        errorMessage += 'Permission denied. Please check file permissions.';
    } else {
        errorMessage += `An unexpected I/O error occurred: ${err.message}`;
    }
    console.error(errorMessage);
}

fs.readFile(inputFile, 'utf8', (err, data) => {
    if (err) {
        handleError(err, 'reading', inputFile);
        return;
    }

    // Process the content (e.g., convert to uppercase)
    const processedContent = data.toUpperCase();

    fs.writeFile(outputFile, processedContent, 'utf8', (err) => {
        if (err) {
            handleError(err, 'writing', outputFile);
            return;
        }
        console.log('File has been processed and copied successfully to output.txt.');
```


LABQUESTION3:

PROMPT:

API Call Translation

You are part of a software development team migrating a data pipeline from Python to Java.

The Python prototype makes an API call to fetch weather data and prints the response.

Task1: Use AI-assisted coding to translate the following Python API call program into Java while ensuring equivalent functionality (fetch data and print the JSON response).

Task2

Extend the translated Java API program to include **robust error handling**.

The Java version should gracefully handle the following scenarios:

- Invalid API key (HTTP 401)
- Request timeout
- No internet connection or network issues

Print clear error messages for each case without the program crashing.

CODE:

```
import requests
```

```
def get_weather():
```

```

api_key="your_api_key_here"
city = "London"
url=
f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api_
key}"

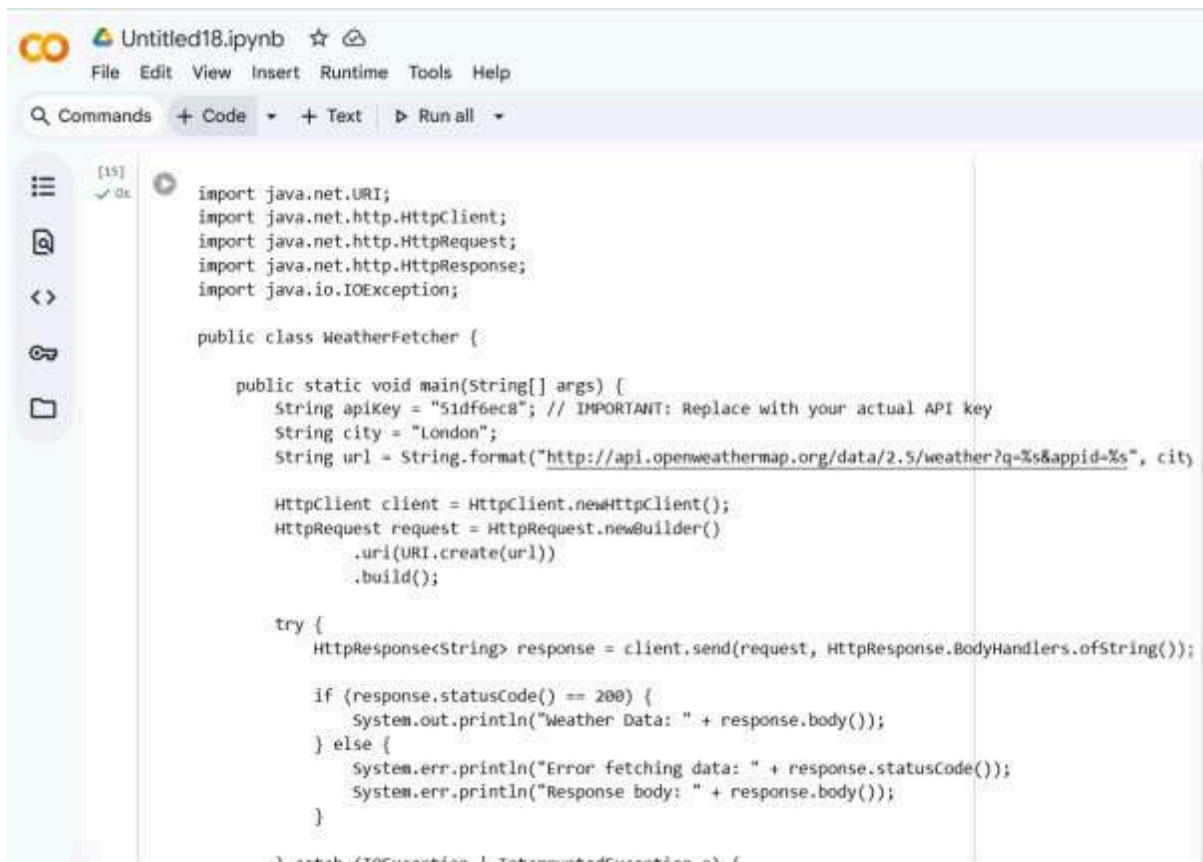
```

```

response=requests.get(url)
if response.status_code==200:
    print("WeatherData:",response.json())
) else:
    print("Errorfetchingdata:",response.status_code)

```

get_weather()



The screenshot shows a Jupyter Notebook window titled "Untitled18.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with options like "Commands", "Code", "Text", and "Run all". On the left, there is a sidebar with icons for file management and execution. The main area displays a Java code snippet for a class named "WeatherFetcher". The code imports necessary Java libraries (java.net.URI, java.net.http.HttpClient, java.net.http.HttpRequest, java.net.http.HttpResponse, and java.io.IOException). It defines a main method that takes an array of arguments. Inside the main method, it sets an API key (commented as "IMPORTANT: Replace with your actual API key"), sets the city to "London", and formats a URL using String.format. It then creates an HttpClient, builds an HttpRequest with the URL, and sends it. A try-catch block handles the response: if the status code is 200, it prints the weather data; otherwise, it prints an error message and the response body. The code is enclosed in a try-catch block for IOException and InterruptedException.

```

[15] ✓ 0s
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.io.IOException;

public class WeatherFetcher {

    public static void main(String[] args) {
        String apiKey = "51df6ec8"; // IMPORTANT: Replace with your actual API key
        String city = "London";
        String url = String.format("http://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s", city,
        apiKey);

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

        try {
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

            if (response.statusCode() == 200) {
                System.out.println("Weather Data: " + response.body());
            } else {
                System.err.println("Error fetching data: " + response.statusCode());
                System.err.println("Response body: " + response.body());
            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[15] ✓ Out

```
String url = String.format("http://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s", city, apiKey);

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

try {
    HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

    if (response.statusCode() == 200) {
        System.out.println("Weather Data: " + response.body());
    } else {
        System.err.println("Error fetching data: " + response.statusCode());
        System.err.println("Response body: " + response.body());
    }
} catch (IOException | InterruptedException e) {
    System.err.println("An error occurred during the API call: " + e.getMessage());
    e.printStackTrace();
}
}
```

Overwriting WeatherFetcher.java

Now, let's compile and run the Java code.

Since Colab primarily supports Python, we'll compile and run the Java code using shell commands. Please ensure

Untitled18.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[19] ✓ Out

```
work, for demonstration purposes, I'll write this Java code to a file named weatherfetcher.java.
```

```
%writefile WeatherFetcher.java

import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.time.Duration; // Import Duration for timeouts

public class WeatherFetcher {

    public static void main(String[] args) {
        // IMPORTANT: Replace with your actual API key, or use an invalid one for testing 401 error
        String apiKey = "51df6ec8";
        String city = "London";
        String url = String.format("http://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s", city, apiKey);

        HttpClient client = HttpClient.newBuilder()
            .connectTimeout(Duration.ofSeconds(10)) // Set connection timeout
            .build();

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .timeout(Duration.ofSeconds(15)) // Set request timeout
            .build();

        try {
```



Untitled18.ipynb ☆ ☁

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all ▼

[19]
✓ 0s

```
try {
    HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

    if (response.statusCode() == 200) {
        System.out.println("Weather Data: " + response.body());
    } else if (response.statusCode() == 401) {
        System.err.println("Error: Invalid API Key. Please check your OpenWeatherMap API key.");
        System.err.println("Response body: " + response.body());
    } else if (response.statusCode() == 404) {
        System.err.println("Error: City not found or invalid request. Status Code: " + response.statusCode());
        System.err.println("Response body: " + response.body());
    } else {
        System.err.println("Error fetching data. Status Code: " + response.statusCode());
        System.err.println("Response body: " + response.body());
    }
} catch (IOException e) {
    if (e instanceof java.net.UnknownHostException) {
        System.err.println("Network Error: Unknown host. Check your internet connection or URL.");
    } else if (e instanceof java.net.SocketTimeoutException) {
        System.err.println("Network Error: Request timed out. The server took too long to respond.");
    } else {
        System.err.println("Network or I/O Error: An error occurred during the API call: " + e.getMessage());
    }
    e.printStackTrace();
} catch (InterruptedException e) {
    System.err.println("Error: The API call was interrupted: " + e.getMessage());
    Thread.currentThread().interrupt(); // Restore the interrupted status
    e.printStackTrace();
}
```

