LABTEST-4

NAME:VITTAM VENKATESH.

BATCH:15

HALLTICKET NUMBER:2403A52419

TASK-1

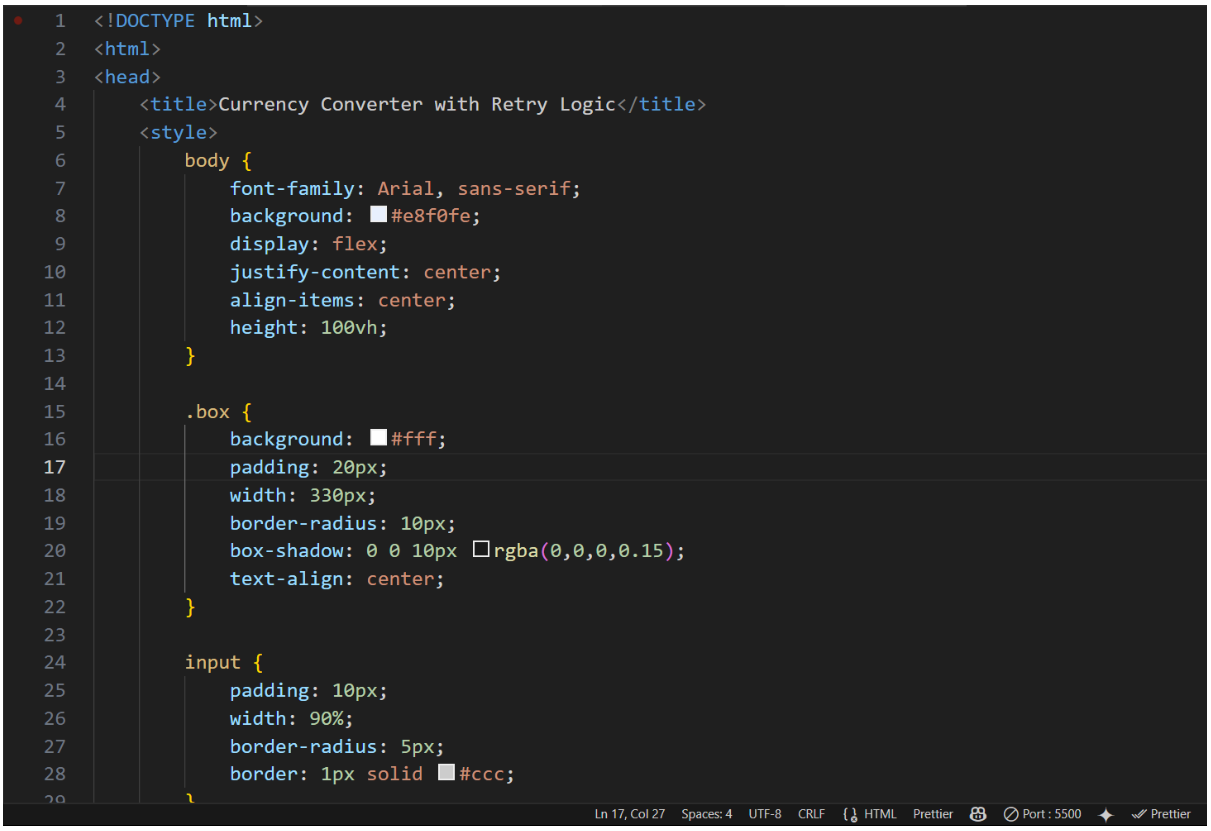
QUESTION-1:

Q1. (API Integration)
a) Connect to a Currency Conversion API.
b) Add retry logic when the API response is delayed.

PROMPT:

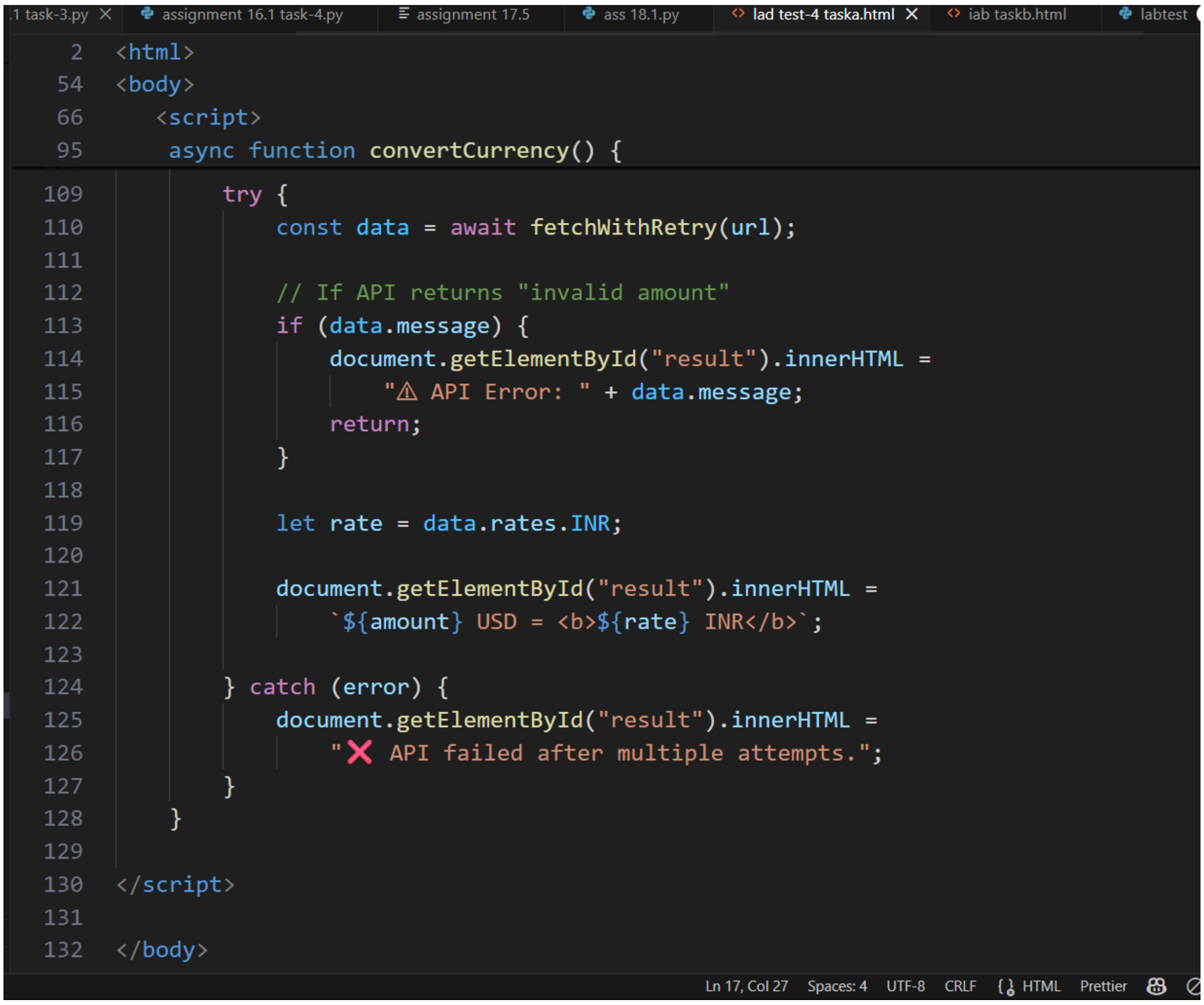
Write a Python program that connects to a Currency Conversion API to fetch the latest exchange rates.
Implement retry logic to handle delayed or failed API responses using techniques such as try-except blocks, loops, and time delays.

CODE:



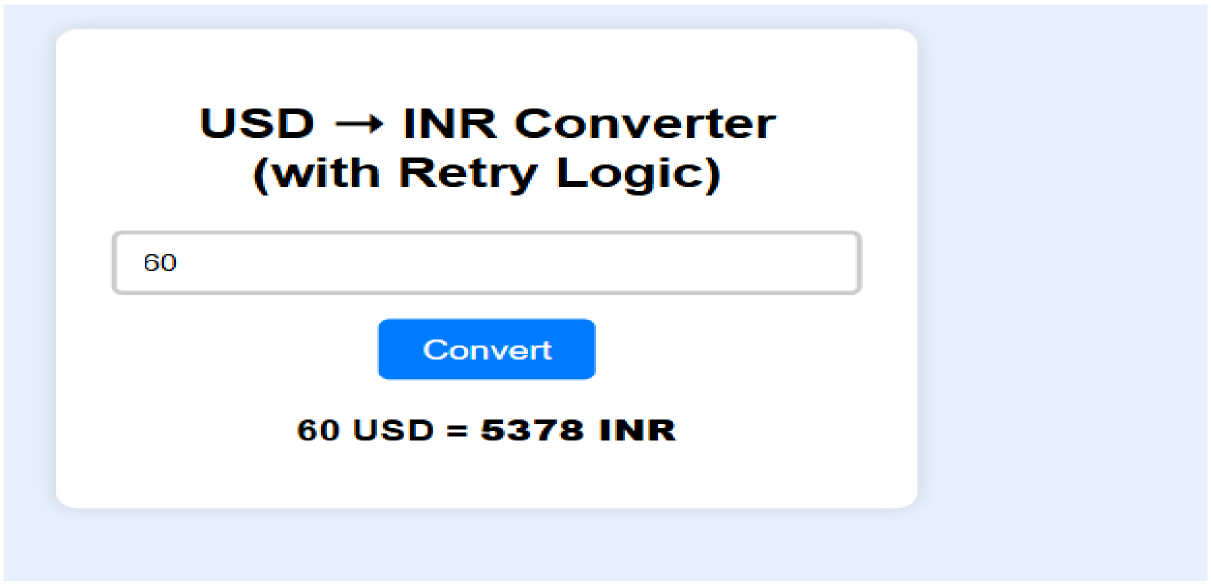
```
<!DOCTYPE html>
<html>
<head>
    <title>Currency Converter with Retry Logic</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: #e8f0fe;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }

        .box {
            background: #fff;
            padding: 20px;
            width: 330px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0,0,0,0.15);
            text-align: center;
        }

        input {
            padding: 10px;
            width: 90%;
            border-radius: 5px;
            border: 1px solid #ccc;
```

Ln 17, Col 27   Spaces: 4   UTF-8   CRLF   { } HTML   Prettier   Port : 5500   Prettier



```
<html>
<body>
  <script>
    async function convertCurrency() {
        try {
            const data = await fetchWithRetry(url);

            // If API returns "invalid amount"
            if (data.message) {
                document.getElementById("result").innerHTML =
                    "⚠ API Error: " + data.message;
                return;
            }

            let rate = data.rates.INR;

            document.getElementById("result").innerHTML =
                `${amount} USD = <b>${rate} INR</b>`;

        } catch (error) {
            document.getElementById("result").innerHTML =
                "❌ API failed after multiple attempts.";
        }
    }

</script>

</body>
```

OUTPUT:



USD → INR Converter (with Retry Logic)

60 USD = 5378 INR

OBSERVATION:

The program demonstrates how Python interacts with external APIs using HTTP requests. Retry logic ensures that the system remains stable even when the API is slow or temporarily unresponsive. By using loops, exceptions, and time delays, the program becomes more reliable and fault-tolerant. This approach increases the robustness of real-time applications where network issues or API delays are common.

TASK-2

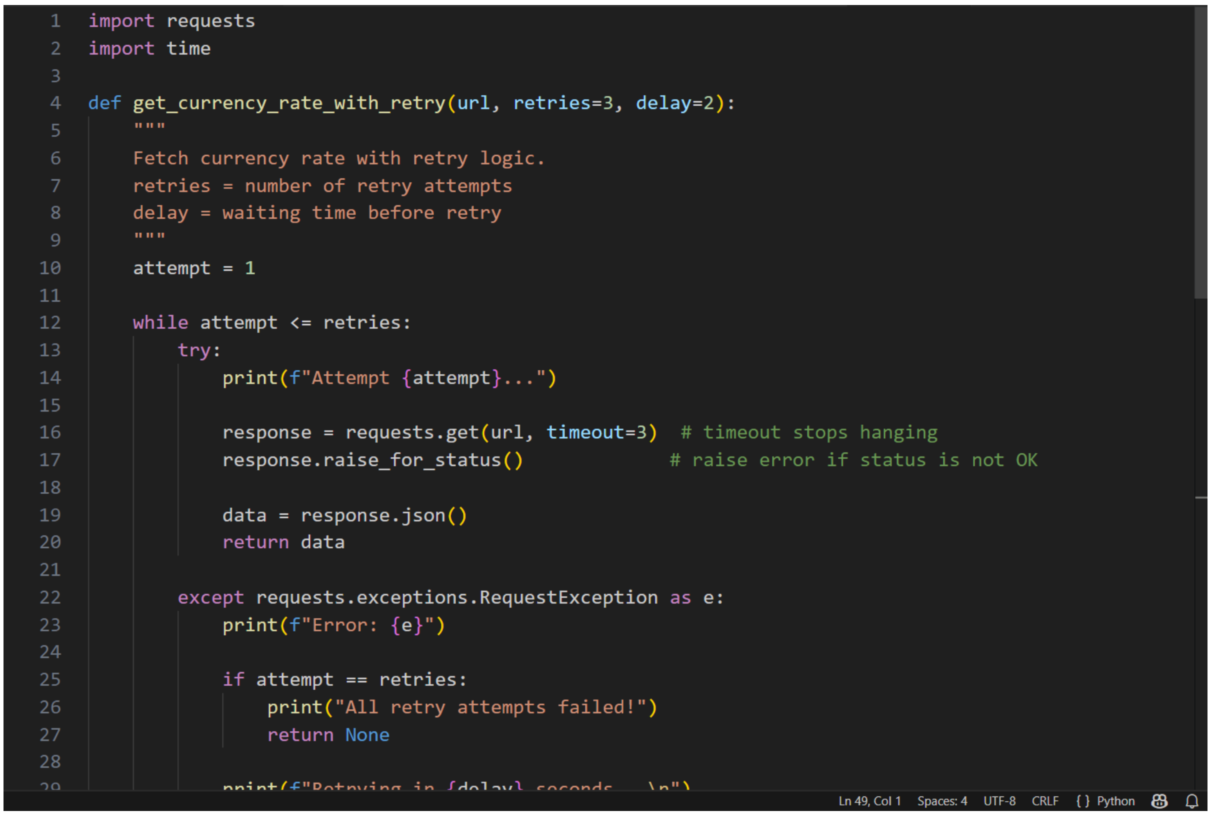
QUESTION-2:

Q2. (Code Translation)
a) Translate a C program for factorial calculation into Python.
b) Ensure the translated code follows Pythonic conventions

PROMPT:

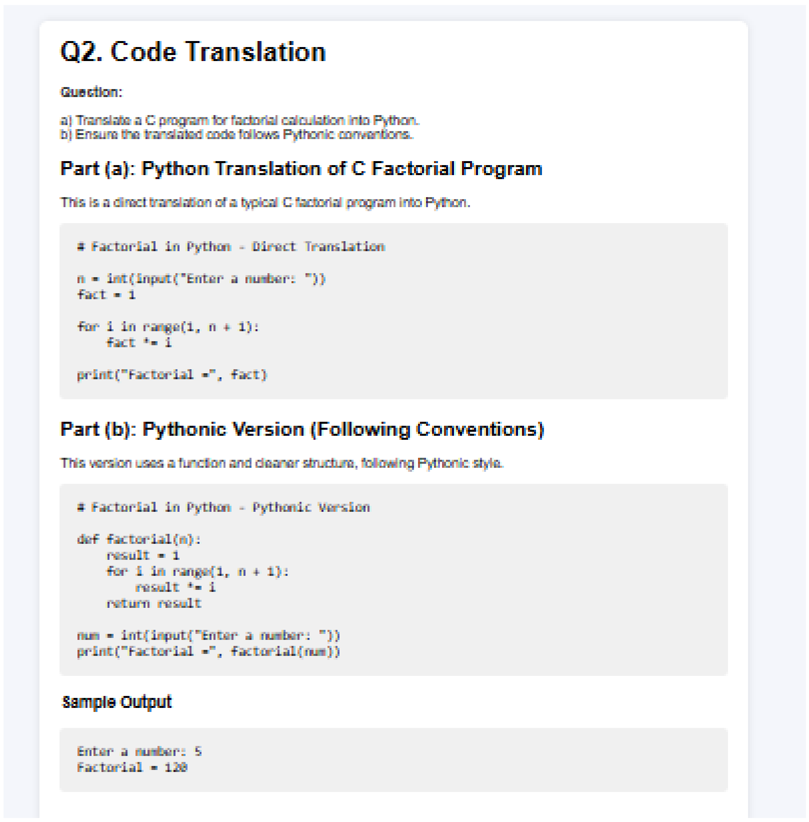
Translate the following C program that calculates the factorial of a number into Python. Ensure the translated Python code follows proper Pythonic conventions such as using functions, clear variable naming, range(), and f-strings.

CODE:



```python
import requests
import time

def get_currency_rate_with_retry(url, retries=3, delay=2):
    """
    Fetch currency rate with retry logic.
    retries = number of retry attempts
    delay = waiting time before retry
    """
    attempt = 1

    while attempt <= retries:
        try:
            print(f"Attempt {attempt}...")

            response = requests.get(url, timeout=3)  # timeout stops hanging
            response.raise_for_status()             # raise error if status is not OK

            data = response.json()
            return data

        except requests.exceptions.RequestException as e:
            print(f"Error: {e}")

            if attempt == retries:
                print("All retry attempts failed!")
                return None

            print(f"Retrying in {delay} seconds...\n")
```

OUTPUT:



Q2. Code Translation

Question:

a) Translate a C program for factorial calculation into Python.
b) Ensure the translated code follows Pythonic conventions.

Part (a): Python Translation of C Factorial Program

This is a direct translation of a typical C factorial program into Python.

```python
# Factorial in Python - Direct Translation

n = int(input("Enter a number: "))
fact = 1

for i in range(1, n + 1):
    fact *= i

print("Factorial =", fact)
```

Part (b): Pythonic Version (Following Conventions)

This version uses a function and cleaner structure, following Pythonic style.

```python
# Factorial in Python - Pythonic Version

def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

num = int(input("Enter a number: "))
print("Factorial =", factorial(num))
```

Sample Output

```
Enter a number: 5
Factorial = 120
```

OBSERVATION:

The Python version of the factorial program is simpler and more readable than the C version. Python does not require variable type declarations, and it handles large integers automatically, making the code shorter and easier to maintain. The use of functions, range(), and f-strings follow Pythonic conventions and improve clarity. In contrast, the C program requires more boilerplate code such as headers, data type declarations, and manual long-integer handling.