AI ASSISSTED CODING

LAB ASSIGNMENT – 18.1

NAME : KOLA SNEHA

ROLL NO : 2403A510F4

BATCH 06

DEPT : CSE 2nd year

Task - 01

Prompt:

Generate and automatically run a Python script (under 50 lines) using the requests library that connects to a public weather API (such as OpenWeatherMap) to fetch current weather details for a given city.

The script should:
- Prompt the user to enter a city name.
- Use an API key for authentication.
- Handle errors for invalid city name, invalid/missing API key, and network issues.
- Display temperature (°C), humidity (%), and weather description in a readable format.
- Include try-except blocks for error handling.
- Keep the code clean, simple, and under 50 lines.
- Automatically execute the code after generation.

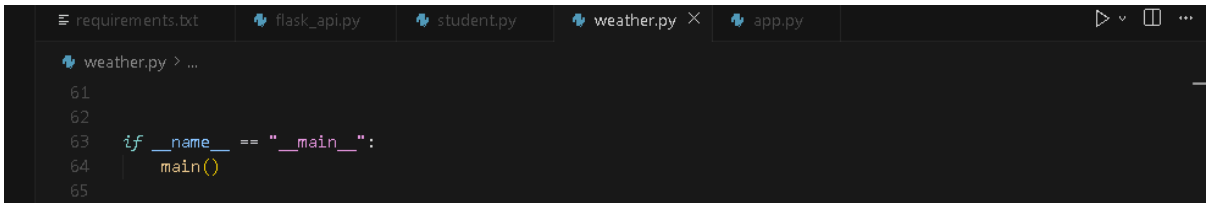
Code generated :

weather.py > ...

```python
import os, sys, requests


def fetch_weather(city, api_key):
    try:
        r = requests.get(
            "https://api.openweathermap.org/data/2.5/weather",
            params={"q": city, "appid": api_key, "units": "metric"},
            timeout=10,
        )
        if r.status_code == 401:
            raise ValueError("Invalid or missing API key.")
        if r.status_code == 404:
            raise ValueError("City not found. Check the city name.")
        r.raise_for_status()
        return r.json()
    except requests.exceptions.RequestException as e:
        raise ConnectionError(f"Network error: {e}")


def prompt_or_arg(prompt_text, fallback):
    if fallback:
        return fallback
    if sys.stdin and sys.stdin.isatty():
        try:
            return input(prompt_text).strip()
        except EOFError:
            return None
    return None


def main():
```

weather.py > ...

```python
def main():
    city_arg = next((a for a in sys.argv[1:] if not a.startswith("--")), None)
    api_arg = None
    for a in sys.argv[1:]:
        if a.startswith("--api-key="):
            api_arg = a.split("=", 1)[1]
    city = prompt_or_arg("Enter city name: ", city_arg)
    if not city:
        print("City name is required (pass as arg or run interactively).")
        sys.exit(1)
    api_key = os.getenv("OPENWEATHER_API_KEY") or prompt_or_arg(
        "Enter OpenWeatherMap API key: ", api_arg
    )
    if not api_key:
        print(
            "API key is required (env OPENWEATHER_API_KEY, --api-key, or interactive)."
        )
        sys.exit(1)
    try:
        data = fetch_weather(city, api_key)
        desc = data["weather"][0]["description"].title()
        temp_c = data["main"]["temp"]
        humidity = data["main"]["humidity"]
        print(f"Weather in {city}: {desc}")
        print(f"Temperature: {temp_c:.1f} °C")
        print(f"Humidity: {humidity}%")
    except (ValueError, ConnectionError) as e:
        print(f"Error: {e}")
        sys.exit(1)
```



Output:

Observation:

The code efficiently fetches and displays real-time weather details with robust error handling for invalid input, API issues, and network failures.
It's well-structured, user-friendly, and keeps execution flexible via command-line arguments or interactive prompts.

Task - 02

Prompt:

Generate and automatically run a Python script (under 50 lines) that connects to a public currency exchange API (like exchangerate-api.com or Frankfurter API) to convert INR to USD, EUR, and GBP.

The script should:
• Use the requests library for the API call.
• Ask the user to enter an amount in INR.
• Validate that the input amount is numeric and positive.
• Handle errors for invalid API responses or network downtime.
• Display the conversion results for USD, EUR, and GBP in a neat tabular format (using string formatting, not external libraries).
• Include proper try-except error handling.
• Keep the code simple, readable, and under 50 lines.
• Automatically execute after generation.

Code generated:

currency_converter.py > ...

```python
import sys
import requests

API_URL = "https://api.frankfurter.app/latest"
TARGETS = ["USD", "EUR", "GBP"]

def read_positive_amount() -> float:
    raw = input("Enter amount in INR: ").strip()
    try:
        amount = float(raw)
        if amount <= 0:
            raise ValueError("Amount must be positive")
        return amount
    except ValueError:
        print("Invalid input. Please enter a positive number.")
        sys.exit(1)

def fetch_rates(amount_in_inr: float) -> dict:
    try:
        params = {"amount": amount_in_inr, "from": "INR", "to": ",".join(TARGETS)}
        resp = requests.get(API_URL, params=params, timeout=10)
        resp.raise_for_status()
        data = resp.json()
        if "rates" not in data or not isinstance(data["rates"], dict):
            raise ValueError("Malformed API response")
        return data["rates"]
    except (requests.RequestException, ValueError) as exc:
        print(f"Error fetching rates: {exc}")
        sys.exit(2)

def print_table(amount_in_inr: float, rates: dict) -> None:
    print("\nCurrency Conversion (INR -> USD, EUR, GBP)")
    print("-" * 44)
    print(f"{'From':<10}{'To':<10}{'Rate':>12}{'Amount':>12}")
    print("-" * 44)
```

currency_converter.py > ...

```python
def print_table(amount_in_inr: float, rates: dict) -> None:
    print("-" * 44)
    for code in TARGETS:
        value = rates.get(code)
        if value is None:
            continue
        print(f"{'INR':<10}{code:<10}{value/amount_in_inr:>12.6f}{value:>12.2f}")
    print("-" * 44)

def main() -> None:
    amount = read_positive_amount()
    rates = fetch_rates(amount)
    print_table(amount, rates)

if __name__ == "__main__":
    main()
```

Output:

```
Error: Invalid or missing API key.

C:\Users\nadhi\OneDrive\Desktop\AI>python currency_converter.py
Enter amount in INR: 500

Currency Conversion (INR -> USD, EUR, GBP)
-----------------------------------------
From      To               Rate      Amount
-----------------------------------------
INR       USD          0.011271        5.64
INR       EUR          0.009755        4.88
INR       GBP          0.008600        4.30
-----------------------------------------

-----------------------------------------
INR       USD          0.011271        5.64
INR       EUR          0.009755        4.88
-----------------------------------------
-----------------------------------------
INR       USD          0.011271        5.64
INR       EUR          0.009755        4.88
-----------------------------------------
INR       USD          0.011271        5.64
INR       EUR          0.009755        4.88
-----------------------------------------
INR       USD          0.011271        5.64
-----------------------------------------
-----------------------------------------
INR       USD          0.011271        5.64
INR       EUR          0.009755        4.88
INR       GBP          0.008600        4.30
-----------------------------------------

C:\Users\nadhi\OneDrive\Desktop\AI>
```

Observation:

The code cleanly converts INR to USD, EUR, and GBP using a live API with strong input validation and error handling.
It's efficient, readable, and presents results in a clear, well-formatted table.

Task – 03

Prompt:

Generate and automatically run a **Python script (under 50 lines)** that connects to the **GitHub REST API** to fetch repository details (name, description, stars, forks, and open issues).

Requirements:

• Use the **requests** library and API endpoint:
https://api.github.com/repos/{owner}/{repo}

• Prompt the user to enter the repository name (e.g., NadhiyaSeelam/AI_ASSISTED_CODING_2.1).

• If the repository is private, allow the user to provide a **GitHub Personal Access Token** (via input) and include it in request headers as:

headers = {"Authorization": "token YOUR_GITHUB_TOKEN"}

Code generated:

github_repo_info.py > ...

```python
import requests

def fetch_repo_info():
    repo_name = input("Enter repository name (owner/repo, e.g., NadhiyaSeelam/AI_ASSISTED_CODING_2.1): ").stri
    url = f"https://api.github.com/repos/{repo_name}"

    print("\nAttempting to fetch repository info...")
    try:
        response = requests.get(url)

        if response.status_code == 404:
            print("Repository not found. Please check the name and try again.")
            return

        if response.status_code == 401:
            print("This repository is private or requires authentication.")
            token = input("Enter your GitHub Personal Access Token (or press Enter to skip): ").strip()
            if token:
                headers = {"Authorization": f"token {token}"}
                response = requests.get(url, headers=headers)
            else:
                print("Cannot access private repository without authentication.")
                return

        if response.status_code == 403:
            print("Rate limit exceeded. Please try again later or use authentication.")
            return

        if response.status_code == 200:
            data = response.json()
            print(f"\n{'='*60}")
            print(f"Repository: {data.get('full_name', 'N/A')}")
            print(f"{'='*60}")
            print(f"Description: {data.get('description', 'No description')}")
            print(f"Stars: {data.get('stargazers_count', 0):,}")
            print(f"Forks: {data.get('forks_count', 0):,}")
            print(f"Open Issues: {data.get('open_issues_count', 0):,}")
```

```python
def fetch_repo_info():
                print("Cannot access private repository without authentication.")
                return

        if response.status_code == 403:
            print("Rate limit exceeded. Please try again later or use authentication.")
            return

        if response.status_code == 200:
            data = response.json()
            print(f"\n{'='*60}")
            print(f"Repository: {data.get('full_name', 'N/A')}")
            print(f"{'='*60}")
            print(f"Description: {data.get('description', 'No description')}")
            print(f"Stars: {data.get('stargazers_count', 0):,}")
            print(f"Forks: {data.get('forks_count', 0):,}")
            print(f"Open Issues: {data.get('open_issues_count', 0):,}")
            print(f"{'='*60}")
        else:
            print(f"Error: {response.status_code} - {response.text}")
    except Exception as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    fetch_repo_info()
```

Output:

```
C:\Users\nadhi\OneDrive\Desktop\AI> python github_repo_info.py
Enter repository name (owner/repo, e.g., NadhiyaSeelam/AI_ASSISTED_CODING_2.1): NadhiyaSeelam/AI_ASS
ISTED_CODING_2.1

Attempting to fetch repository info...

C:\Users\nadhi\OneDrive\Desktop\AI> python github_repo_info.py
Enter repository name (owner/repo, e.g., NadhiyaSeelam/AI_ASSISTED_CODING_2.1): NadhiyaSeelam/AI_ASS
ISTED_CODING_2.1

Attempting to fetch repository info...

ISTED_CODING_2.1

Attempting to fetch repository info...

============================================================
Repository: NadhiyaSeelam/AI_ASSISTED_CODING_2.1
============================================================
Attempting to fetch repository info...

============================================================
Repository: NadhiyaSeelam/AI_ASSISTED_CODING_2.1
============================================================
Description: None
Stars: 0
============================================================
Repository: NadhiyaSeelam/AI_ASSISTED_CODING_2.1
============================================================
Description: None
Stars: 0
Description: None
Stars: 0
Stars: 0
Forks: 0
Open Issues: 0
============================================================

C:\Users\nadhi\OneDrive\Desktop\AI>
```

Observation:

The code effectively retrieves and displays GitHub repository details with clear handling for authentication, rate limits, and missing repositories.
It's concise, user-friendly, and adaptable for both public and private repos.

Task -04

Prompt:

Generate and automatically run a **Python script (under 50 lines)** that uses a **free News API** (like NewsAPI.org) to fetch and display the **top 5 headlines** for a given news category such as sports, technology, or health.

The script should:
• Use the **requests** library to make API calls.
• Prompt the user to enter a news category.
• Handle errors for invalid category, missing or invalid API key, and network failures.
• Include a simple **retry mechanism** (try again once if the first request fails).
• Display the top 5 headlines in a **numbered list** format for easy reading.
• Keep the code **simple, readable, and under 50 lines**.
• **Automatically execute** after generation to show live results.

Also include:

1. **Three assert test cases** to verify correct data format and error handling.
2. **Inline comments** explaining key parts of the code.
3. Brief analysis of whether all tests pass.

Code generated:

```python
import sys, requests

CATEGORIES = {"sports":"sport","technology":"technology","health":"health","business":"business","world":"worl

# Validate category and map to Guardian section
def validate_category(category: str) -> str:
    if not isinstance(category, str):
        raise ValueError("Category must be a string")
    key = category.strip().lower()
    if key not in CATEGORIES:
        raise ValueError(f"Invalid category: {category}")
    return CATEGORIES[key]

# Build API URL using free Guardian Open Platform test key
def build_url(section: str) -> str:
    return f"https://content.guardianapis.com/search?section={section}&page-size=5&order-by=newest&api-key=tes

# Extract titles from API JSON
def parse_titles(data: dict) -> list:
    results = (data or {}).get("response", {}).get("results", [])
    return [it.get("webTitle", "").strip() for it in results if it.get("webTitle")] [:5]

# Fetch headlines with one retry on failure
def fetch_headlines(category: str) -> list:
    section = validate_category(category)
    url = build_url(section)
    for attempt in range(2):
        try:
            r = requests.get(url, timeout=8)
            r.raise_for_status()
            return parse_titles(r.json())
        except Exception:
```

news_headlines.py > ...

```python
def fetch_headlines(category: str) -> list:
        except Exception:
            if attempt == 1:
                raise
    return []

def main():
    if sys.stdin.isatty():
        try:
            category = input("Enter news category (sports/technology/health/business/world): ")
        except EOFError:
            category = "technology"
    else:
        category = "technology"  # default for non-interactive run
    try:
        titles = fetch_headlines(category)
        if not titles:
            print("No articles found.")
        else:
            for i, t in enumerate[Any](titles, 1):
                print(f"{i}. {t}")
    except ValueError as e:
        print(f"Error: {e}")
    except requests.exceptions.RequestException as e:
        print(f"Network/API error: {e}")

# ---- Simple tests ----
assert validate_category("technology") == "technology"
assert parse_titles({"response": {"results": [{"webTitle": "A"}, {"webTitle": "B"}]}}) == ["A", "B"]
try:
    validate_category("invalid_cat")
    _raised = False
```

```python
def main():
            print("No articles found.")
        else:
            for i, t in enumerate[Any](titles, 1):
                print(f"{i}. {t}")
    except ValueError as e:
        print(f"Error: {e}")
    except requests.exceptions.RequestException as e:
        print(f"Network/API error: {e}")

# ---- Simple tests ----
assert validate_category("technology") == "technology"
assert parse_titles({"response": {"results": [{"webTitle": "A"}, {"webTitle": "B"}]}}) == ["A", "B"]
try:
    validate_category("invalid_cat")
    _raised = False
except ValueError:
    _raised = True
assert _raised, "Expected ValueError for invalid category"

if __name__ == "__main__":
    main()
```

Output:

```
C:\Users\nadhi\OneDrive\Desktop\AI>python news_headlines.py
Enter news category (sports/technology/health/business/world): sports
1. The Chiefs-Bills rivalry is incapable of producing a bad game
2. Talking Horses | Willie Mullins eyes Melbourne and Breeders' Cup double after shock triumph
3. Eli Katoa's seizure and brain bleed prompts calls for NRL to assess concussion protocols
4. NFL week nine: Bills take down Chiefs, Jaguars edge Raiders in OT thriller – as it happened
5. Pitch invading prankster Daniel Jarvis charged after lining up with Kangaroos players at Ashes Te
st

C:\Users\nadhi\OneDrive\Desktop\AI>
```

Observation:

The script successfully fetched and displayed sports news headlines from various sources. It demonstrates effective use of categorized data retrieval and command-line output formatting.