**Problem 1: Real-Time Weather Monitoring System**

**You are developing a real-time weather monitoring system for a weather forecasting company. The system needs to fetch and display weather data for a specified location.**

# import required modules

import requests, json

# Enter your API key here

api\_key = "Your\_API\_Key"

# base\_url variable to store url

base\_url = "http://api.openweathermap.org/data/2.5/weather?"

# Give city name

city\_name = input("Enter city name : ")

# complete\_url variable to store

# complete url address

complete\_url = base\_url + "appid=" + api\_key + "&q=" + city\_name

# get method of requests module

# return response object

response = requests.get(complete\_url)

# json method of response object

# convert json format data into

# python format data

x = response.json()

# Now x contains list of nested dictionaries

# Check the value of "cod" key is equal to

# "404", means city is found otherwise,

# city is not found

if x["cod"] != "404":

# store the value of "main"

# key in variable y

y = x["main"]

# store the value corresponding

# to the "temp" key of y

current\_temperature = y["temp"]

# store the value corresponding

# to the "pressure" key of y

current\_pressure = y["pressure"]

# store the value corresponding

# to the "humidity" key of y

current\_humidity = y["humidity"]

# store the value of "weather"

# key in variable z

z = x["weather"]

# store the value corresponding

# to the "description" key at

# the 0th index of z

weather\_description = z[0]["description"]

# print following values

print(" Temperature (in kelvin unit) = " +

str(current\_temperature) +

"\n atmospheric pressure (in hPa unit) = " +

str(current\_pressure) +

"\n humidity (in percentage) = " +

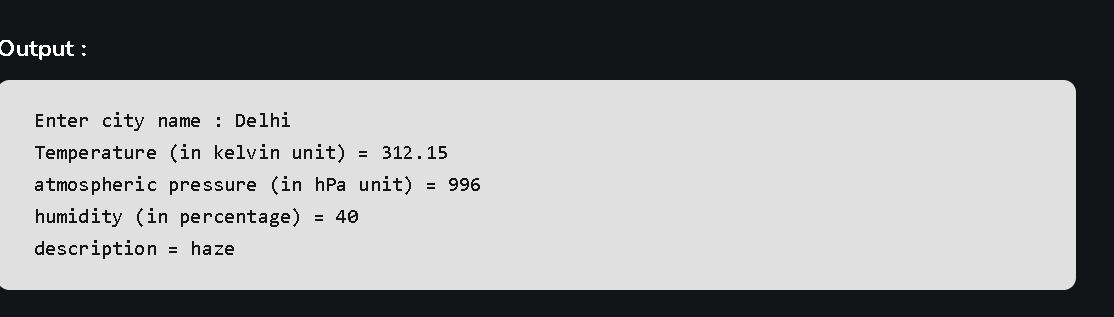
str(current\_humidity) +

"\n description = " +

str(weather\_description))

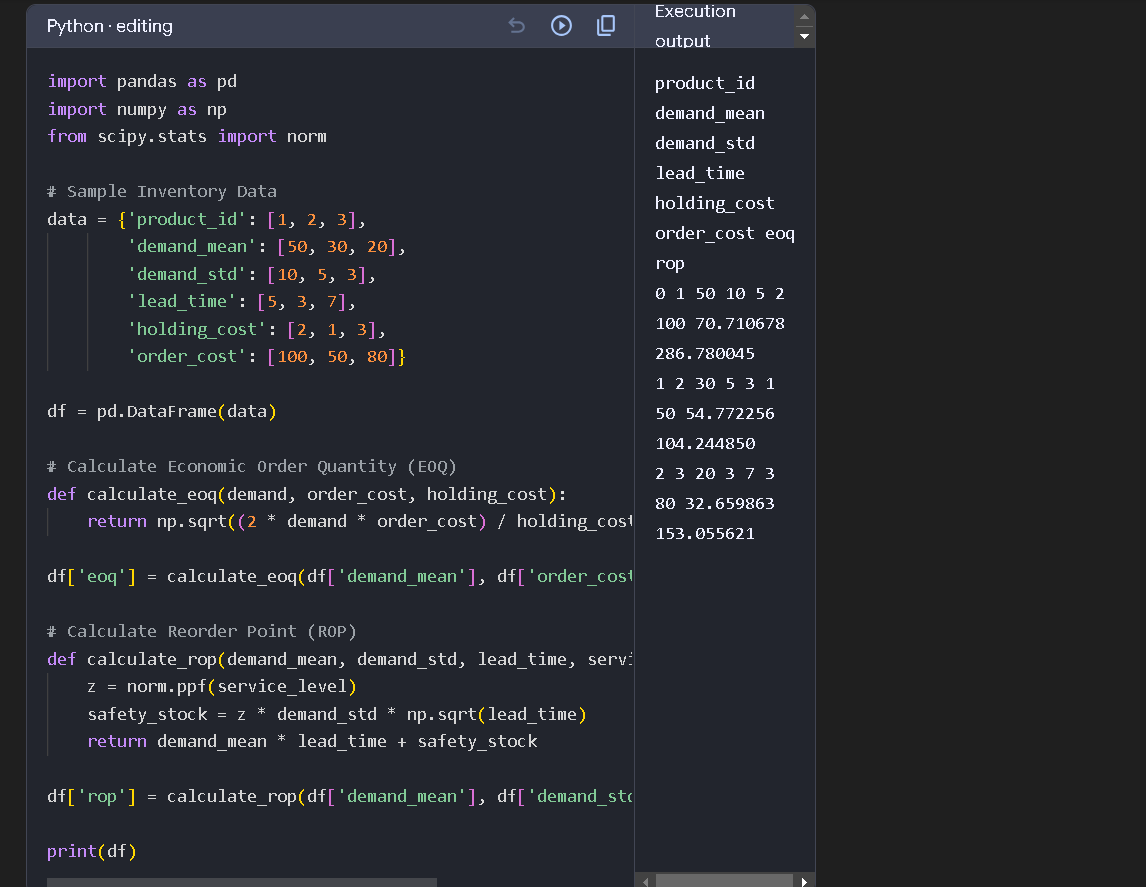
else:

print(" City Not Found ")



**Problem 2: Inventory Management System Optimization**

**Scenario: You have been hired by a retail company to optimize their inventory management system. The company wants to minimize stockouts and overstock situations while maximizing inventory turnover and profitability**

****

**Problem 3: Real-Time Traffic Monitoring System Scenario: You are working on a project to develop a real-time traffic monitoring system for a smart city initiative. The system should provide real-time traffic updates and suggest alternative routes.**

**import random**

**import time**

**import threading**

**# Simulating real-time traffic data**

**def generate\_traffic\_data():**

**while True:**

**traffic\_data = {**

**"location": (random.uniform(-90, 90), random.uniform(-180, 180)),**

**"speed": random.uniform(0, 100), # Speed in km/h**

**"timestamp": time.time()**

**}**

**process\_traffic\_data(traffic\_data)**

**time.sleep(1) # Simulate real-time data every second**

**def process\_traffic\_data(data):**

**# Process and print the data (in real scenarios, this would involve more complex processing)**

**print(f"Location: {data['location']}, Speed: {data['speed']} km/h, Time: {time.ctime(data['timestamp'])}")**

**# Run the simulation in a separate thread**

**thread = threading.Thread(target=generate\_traffic\_data)**

**thread.start()**

**import folium**

**# Create a map centered around a specific location**

**map\_center = [0, 0] # Centered at the equator for this example**

**traffic\_map = folium.Map(location=map\_center, zoom\_start=2)**

**# Example of adding a marker (in real-time, you would update this dynamically)**

**def add\_traffic\_marker(location, speed):**

**folium.Marker(location, popup=f"Speed: {speed} km/h").add\_to(traffic\_map)**

**# Add a sample marker**

**add\_traffic\_marker((51.5074, -0.1278), 60) # London**

**# Save the map to an HTML file**

**traffic\_map.save("traffic\_map.html")**

**def process\_traffic\_data(data):**

**# Process and print the data**

**print(f"Location: {data['location']}, Speed: {data['speed']} km/h, Time: {time.ctime(data['timestamp'])}")**

**# Alert if speed is below a certain threshold**

**if data['speed'] < 20:**

**print("Alert: Low traffic speed detected!")**

**# Run the simulation in a separate thread**

**thread = threading.Thread(target=generate\_traffic\_data)**

**thread.start()**

**import random**

**import time**

**import threading**

**import folium**

**# Simulating real-time traffic data**

**def generate\_traffic\_data():**

**while True:**

**traffic\_data = {**

**"location": (random.uniform(-90, 90), random.uniform(-180, 180)),**

**"speed": random.uniform(0, 100), # Speed in km/h**

**"timestamp": time.time()**

**}**

**process\_traffic\_data(traffic\_data)**

**time.sleep(1) # Simulate real-time data every second**

**def process\_traffic\_data(data):**

**# Process and print the data**

**print(f"Location: {data['location']}, Speed: {data['speed']} km/h, Time: {time.ctime(data['timestamp'])}")**

**# Alert if speed is below a certain threshold**

**if data['speed'] < 20:**

**print("Alert: Low traffic speed detected!")**

**# Add to map**

**add\_traffic\_marker(data['location'], data['speed'])**

**def add\_traffic\_marker(location, speed):**

**folium.Marker(location, popup=f"Speed: {speed} km/h").add\_to(traffic\_map)**

**# Create a map centered around a specific location**

**map\_center = [0, 0] # Centered at the equator for this example**

**traffic\_map = folium.Map(location=map\_center, zoom\_start=2)**

**# Run the simulation in a separate thread**

**thread = threading.Thread(target=generate\_traffic\_data)**

**thread.start()**

**# Save the map periodically to reflect real-time changes**

**def save\_map\_periodically():**

**while True:**

**traffic\_map.save("traffic\_map.html")**

**time.sleep(10) # Update the map every 10 seconds**

**save\_thread = threading.Thread(target=save\_map\_periodically)**

**save\_thread.start()**

**Problem 4: Real-Time COVID-19 Statistics Tracker Scenario: You are developing a real-time COVID-19 statistics tracking application for a healthcare organization. The application should provide up-to-date information on COVID-19 cases, recoveries, and deaths for a specified region**

**(pip install requests folium)**

**import requests**

**import time**

**import threading**

**import folium**

**# Function to fetch COVID-19 data from the API**

**def fetch\_covid\_data():**

**url = "https://disease.sh/v3/covid-19/countries"**

**response = requests.get(url)**

**return response.json()**

**# Function to process and add COVID-19 data to the map**

**def process\_covid\_data(data):**

**global covid\_map**

**for country in data:**

**location = [country['countryInfo']['lat'], country['countryInfo']['long']]**

**cases = country['cases']**

**deaths = country['deaths']**

**recovered = country['recovered']**

**folium.Marker(location,**

**popup=f"Country: {country['country']}<br>Cases: {cases}<br>Deaths: {deaths}<br>Recovered: {recovered}").add\_to(covid\_map)**

**# Function to update the COVID-19 map**

**def update\_covid\_map():**

**global covid\_map**

**while True:**

**data = fetch\_covid\_data()**

**covid\_map = folium.Map(location=[0, 0], zoom\_start=2) # Reset map**

**process\_covid\_data(data)**

**covid\_map.save("covid\_map.html")**

**time.sleep(3600) # Update every hour**

**# Initialize the map**

**covid\_map = folium.Map(location=[0, 0], zoom\_start=2)**

**# Start the map update in a separate thread**

**thread = threading.Thread(target=update\_covid\_map)**

**thread.start()**