ID 220505542

Tatyana Selemenchuk

Software management Project

“Earthquake Prediction”

import random

import time

from datetime import datetime, timedelta

# Historical earthquake data for cities in Turkey, including geographic coordinates

earthquake\_data = {

"Istanbul": {

"historical\_quakes": [447, 554, 1509, 1766],

"total\_quakes": 4,

"coordinates": (41.0082, 28.9784) # Latitude, Longitude

},

"Ankara": {

"historical\_quakes": [1940, 2020],

"total\_quakes": 2,

"coordinates": (39.9334, 32.8597)

},

"Izmir": {

"historical\_quakes": [1653, 1688, 1999],

"total\_quakes": 3,

"coordinates": (38.4192, 27.1287)

},

"Bursa": {

"historical\_quakes": [1855, 1999],

"total\_quakes": 2,

"coordinates": (40.1826, 29.0668)

},

"Antalya": {

"historical\_quakes": [1998],

"total\_quakes": 1,

"coordinates": (36.8842, 30.7056)

},

"Adana": {

"historical\_quakes": [1998, 2003],

"total\_quakes": 2,

"coordinates": (37.0019, 35.3285)

},

"Konya": {

"historical\_quakes": [1970, 2020],

"total\_quakes": 2,

"coordinates": (37.8714, 32.4846)

},

"Gaziantep": {

"historical\_quakes": [1986, 2001],

"total\_quakes": 2,

"coordinates": (37.0662, 37.3833)

},

"Kayseri": {

"historical\_quakes": [1990, 2006],

"total\_quakes": 2,

"coordinates": (38.7312, 35.4787)

},

"Mersin": {

"historical\_quakes": [1997],

"total\_quakes": 1,

"coordinates": (36.8055, 34.6204)

}

}

# Function to simulate recent seismic activity

def recent\_seismic\_activity():

"Simulate recent seismic activity (True for high, False for low)."

return random.choice([True, False])

# Function to predict earthquake details

def predict\_earthquake(location):

"Simulate predicting an earthquake at a specific location."

city\_data = earthquake\_data[location]

historical\_quakes = city\_data["historical\_quakes"]

# Assess recent seismic activity

recent\_activity = recent\_seismic\_activity()

# Calculate prediction probability

probability = (len(historical\_quakes) + (1 if recent\_activity else 0)) / 10

probability\_percentage = round(probability \* 100, 2) # Convert to percentage

# Determine if an earthquake is predicted based on calculated probability

prediction = random.random() < probability

if prediction:

prediction\_date = datetime.now() + timedelta(days=random.randint(1, 7)) # Predict within the next week

# Estimate magnitude based on probability

magnitude = round(random.uniform(4.0, 7.0) \* probability, 1) # Adjust magnitude based on probability

return [

{

"historical\_data": f"Historical data indicates {len(historical\_quakes)} significant quakes.",

"predicting\_by\_location": "High seismic activity" if recent\_activity else "Low seismic activity",

"recent\_activity": recent\_activity,

"probability": f"{probability\_percentage}%",

"magnitude": magnitude,

"coordinates": city\_data["coordinates"]

},

prediction\_date.strftime('%Y-%m-%d')

]

else:

return [

{

"historical\_data": f"Historical data indicates {len(historical\_quakes)} significant quakes.",

"predicting\_by\_location": "Low seismic activity",

"recent\_activity": recent\_activity,

"probability": f"{probability\_percentage}%",

"magnitude": "N/A",

"coordinates": city\_data["coordinates"]

},

None

]

# Function to display earthquake prediction

def display\_prediction(prediction\_info):

if prediction\_info[1]: # If a prediction date exists

prediction\_details = prediction\_info[0]

print("\n EARTHQUAKE PREDICTION ")

print(f"Date: {prediction\_info[1]}")

print(f"Historical Data: {prediction\_details['historical\_data']}")

print(f"Prediction Status: {prediction\_details['predicting\_by\_location']}")

print(f"Recent Activity: {'Yes' if prediction\_details['recent\_activity'] else 'No'}")

print(f"Probability: {prediction\_details['probability']}")

print(f"Estimated Magnitude: {prediction\_details['magnitude']}")

print(f"Coordinates: {prediction\_details['coordinates']}")

print("END OF PREDICTION \n")

else:

print("\nNO EARTHQUAKE PREDICTED ")

print(f"Historical Data: {prediction\_info[0]['historical\_data']}")

print(f"Prediction Status: {prediction\_info[0]['predicting\_by\_location']}")

print(f"Recent Activity: {'Yes' if prediction\_info[0]['recent\_activity'] else 'No'}")

print(f"Probability: {prediction\_info[0]['probability']}")

print(f"Coordinates: {prediction\_info[0]['coordinates']}")

print("END OF PREDICTION \n")

# Main Program

def main():

while True:

location = input("\nEnter the city for earthquake prediction (or type 'EOT' to exit): ")

if location.upper() == 'EOT':

print("Ending transmission. Goodbye!")

break

if location not in earthquake\_data:

print(f"This city is not in the list. Please choose from the following cities:")

print(", ".join(earthquake\_data.keys()))

continue

print("Predicting earthquake...")

time.sleep(2) # Simulate time taken for prediction

# Get the earthquake prediction

prediction\_info = predict\_earthquake(location)

# Display the prediction

display\_prediction(prediction\_info)

# Run the main program

if \_\_name\_\_ == "\_\_main\_\_":

main()