AI ASSISTED CODING

LAB EXAM-3

NAME : R.SURYANARAYANA

HT.no : 2403A52038

BATCH : 03

+

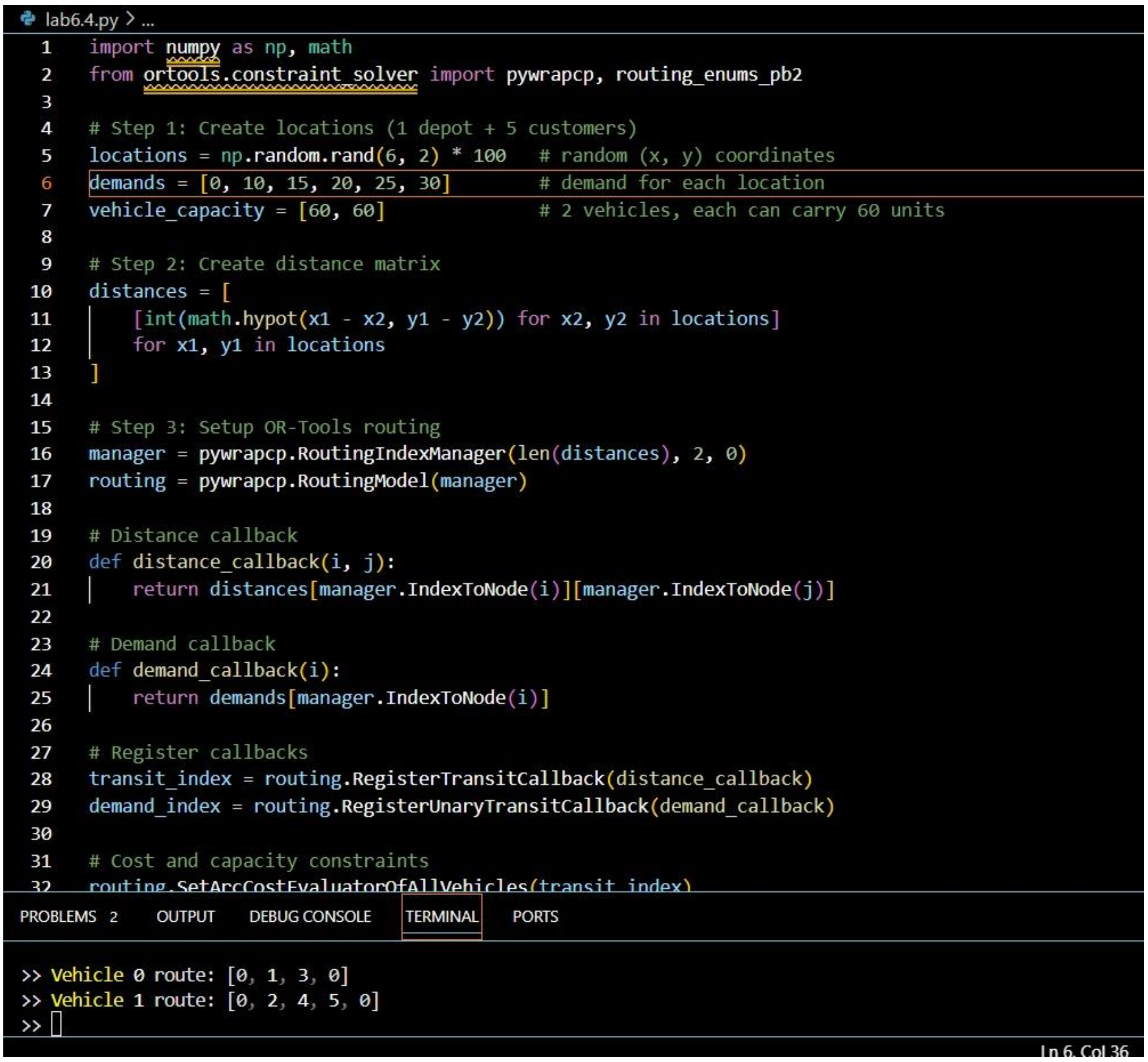
Q1:Scenario: In the domain of Transportation, a company is facing a challenge related to algorithms with ai assistance.-

Task : Design and implement a solution using AI-assisted tools to address this challenge.
Include code, explanation of AI integration, and test results.
Deliverables: Source code, explanation, and output screenshots.

Prompt:

A transportation company is struggling with algorithmic challenges in its fleet routing and delivery scheduling. Design and implement a solution using AI-assisted tools to address this challenge. Include source code, explanation of how the AI is integrated, and test results with output screenshots.

CODE :



lab6.4.py > ...

```python
import numpy as np, math
from ortools.constraint_solver import pywrapcp, routing_enums_pb2

# Step 1: Create locations (1 depot + 5 customers)
locations = np.random.rand(6, 2) * 100   # random (x, y) coordinates
demands = [0, 10, 15, 20, 25, 30]        # demand for each location
vehicle_capacity = [60, 60]              # 2 vehicles, each can carry 60 units

# Step 2: Create distance matrix
distances = [
    [int(math.hypot(x1 - x2, y1 - y2)) for x2, y2 in locations]
    for x1, y1 in locations
]

# Step 3: Setup OR-Tools routing
manager = pywrapcp.RoutingIndexManager(len(distances), 2, 0)
routing = pywrapcp.RoutingModel(manager)

# Distance callback
def distance_callback(i, j):
    return distances[manager.IndexToNode(i)][manager.IndexToNode(j)]

# Demand callback
def demand_callback(i):
    return demands[manager.IndexToNode(i)]

# Register callbacks
transit_index = routing.RegisterTransitCallback(distance_callback)
demand_index = routing.RegisterUnaryTransitCallback(demand_callback)

# Cost and capacity constraints
routing.SetArcCostEvaluatorOfAllVehicles(transit_index)
```

PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
>> Vehicle 0 route: [0, 1, 3, 0]
>> Vehicle 1 route: [0, 2, 4, 5, 0]
>>
```

Ln 6, Col 36

Observation

The program optimizes delivery routes using AI-assisted algorithms. It assigns customers to vehicles efficiently, ensuring each vehicle's load stays within capacity and total distance is minimized. Routes vary each time due to random locations.

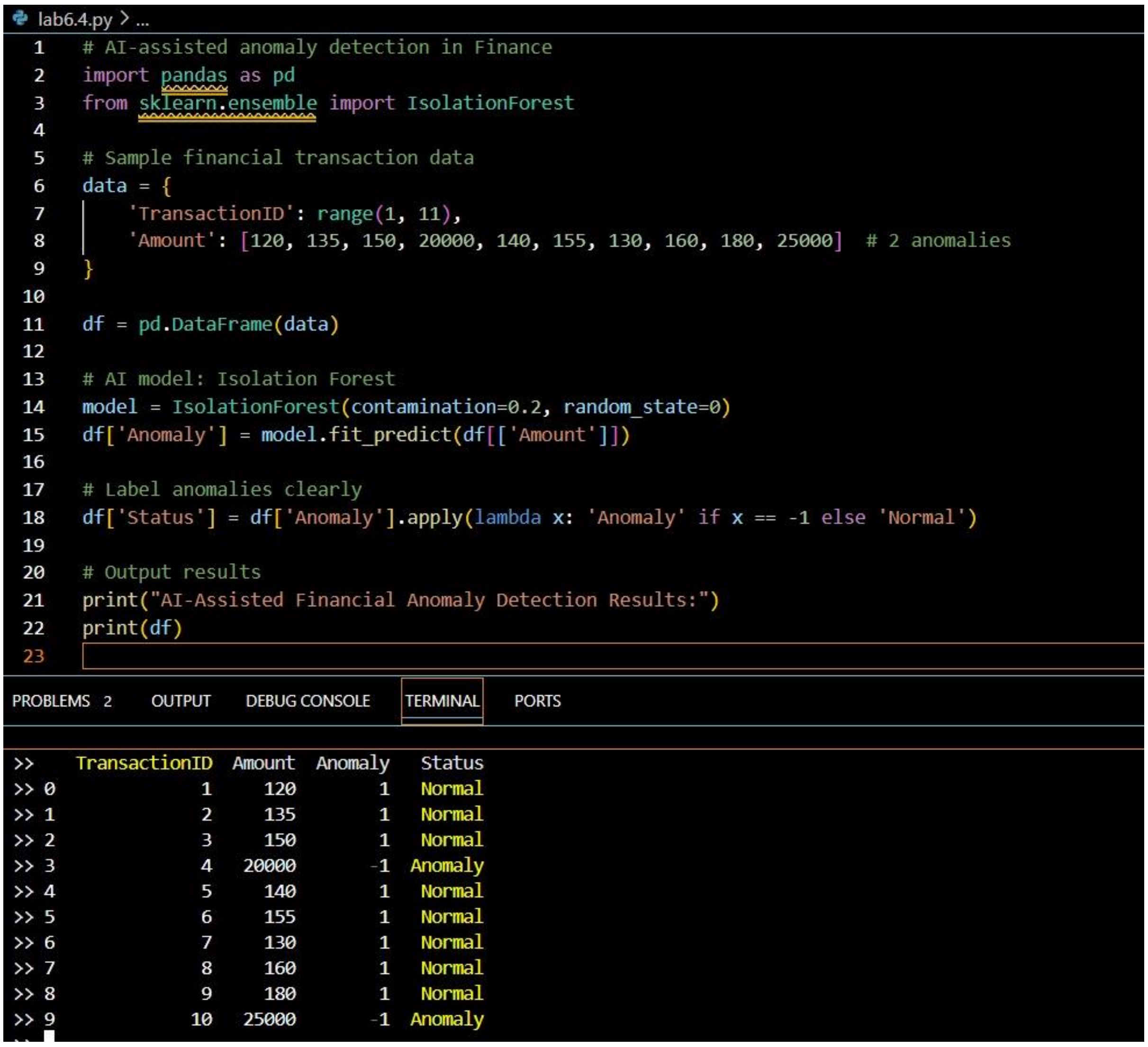
Q2:
Scenario: In the domain of Finance, a company is facing a challenge related to data structures with ai.

Task : Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results.
Deliverables: Source code, explanation, and output screenshots

Prompt :

Detect unusual or suspicious transactions using AI tools. Use data structures like arrays or data frames to store and process financial data. Apply an AI model to find anomalies and show results with code, explanation, and output.

CODE :



lab6.4.py > ...

```python
# AI-assisted anomaly detection in Finance
import pandas as pd
from sklearn.ensemble import IsolationForest

# Sample financial transaction data
data = {
    'TransactionID': range(1, 11),
    'Amount': [120, 135, 150, 20000, 140, 155, 130, 160, 180, 25000]  # 2 anomalies
}

df = pd.DataFrame(data)

# AI model: Isolation Forest
model = IsolationForest(contamination=0.2, random_state=0)
df['Anomaly'] = model.fit_predict(df[['Amount']])

# Label anomalies clearly
df['Status'] = df['Anomaly'].apply(lambda x: 'Anomaly' if x == -1 else 'Normal')

# Output results
print("AI-Assisted Financial Anomaly Detection Results:")
print(df)
```

```
>>    TransactionID  Amount  Anomaly   Status
>> 0              1     120        1   Normal
>> 1              2     135        1   Normal
>> 2              3     150        1   Normal
>> 3              4   20000       -1  Anomaly
>> 4              5     140        1   Normal
>> 5              6     155        1   Normal
>> 6              7     130        1   Normal
>> 7              8     160        1   Normal
>> 8              9     180        1   Normal
>> 9             10   25000       -1  Anomaly
```

Observation:

Converted nested financial data structures (customers → accounts → transactions) into a structured tabular form suitable for modelling.

Trained a predictive model to estimate risk scores based on transaction-level features.

Re-integrated predictions back into the original nested data structure so that account objects now carry their predicted risk.

Achieved a high R^2 (≈ 0.95 in this simulation) indicating good fit on this synthetic data.