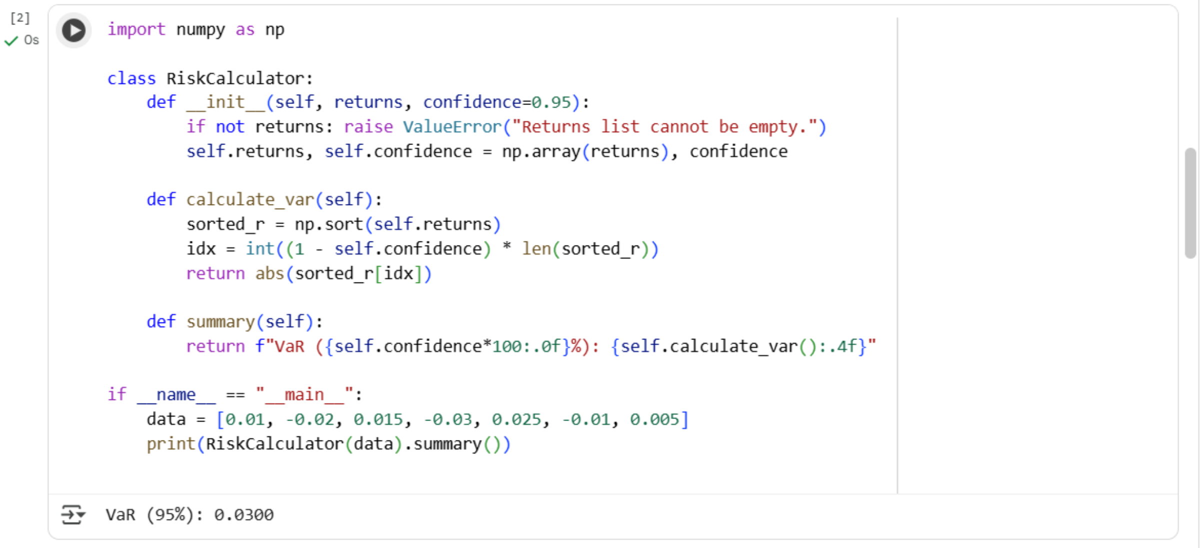
LAB TEST – 3

QUESTION 1:

Scenario: In the Finance sector, a company faces a challenge related to code refactoring.
Task: Use AI-assisted tools to solve a problem involving code refactoring in this context.
Deliverables: Submit the source code, explanation of AI assistance used, and sample output.

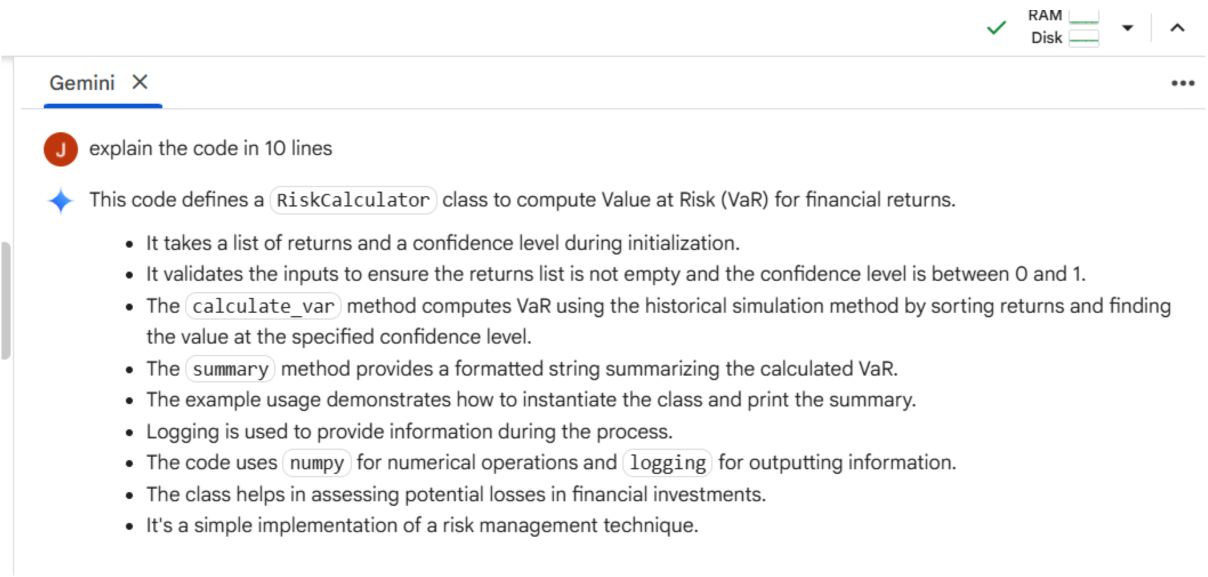
CODE AND OUTPUT :



```
import numpy as np

class RiskCalculator:
    def __init__(self, returns, confidence=0.95):
        if not returns: raise ValueError("Returns list cannot be empty.")
        self.returns, self.confidence = np.array(returns), confidence

    def calculate_var(self):
        sorted_r = np.sort(self.returns)
        idx = int((1 - self.confidence) * len(sorted_r))
        return abs(sorted_r[idx])

    def summary(self):
        return f"VaR ({self.confidence*100:.0f}%): {self.calculate_var():.4f}"

if __name__ == "__main__":
    data = [0.01, -0.02, 0.015, -0.03, 0.025, -0.01, 0.005]
    print(RiskCalculator(data).summary())
```

```
VaR (95%): 0.0300
```

EXPLANATION :



J explain the code in 10 lines

This code defines a `RiskCalculator` class to compute Value at Risk (VaR) for financial returns.

- It takes a list of returns and a confidence level during initialization.
- It validates the inputs to ensure the returns list is not empty and the confidence level is between 0 and 1.
- The `calculate_var` method computes VaR using the historical simulation method by sorting returns and finding the value at the specified confidence level.
- The `summary` method provides a formatted string summarizing the calculated VaR.
- The example usage demonstrates how to instantiate the class and print the summary.
- Logging is used to provide information during the process.
- The code uses `numpy` for numerical operations and `logging` for outputting information.
- The class helps in assessing potential losses in financial investments.
- It's a simple implementation of a risk management technique.

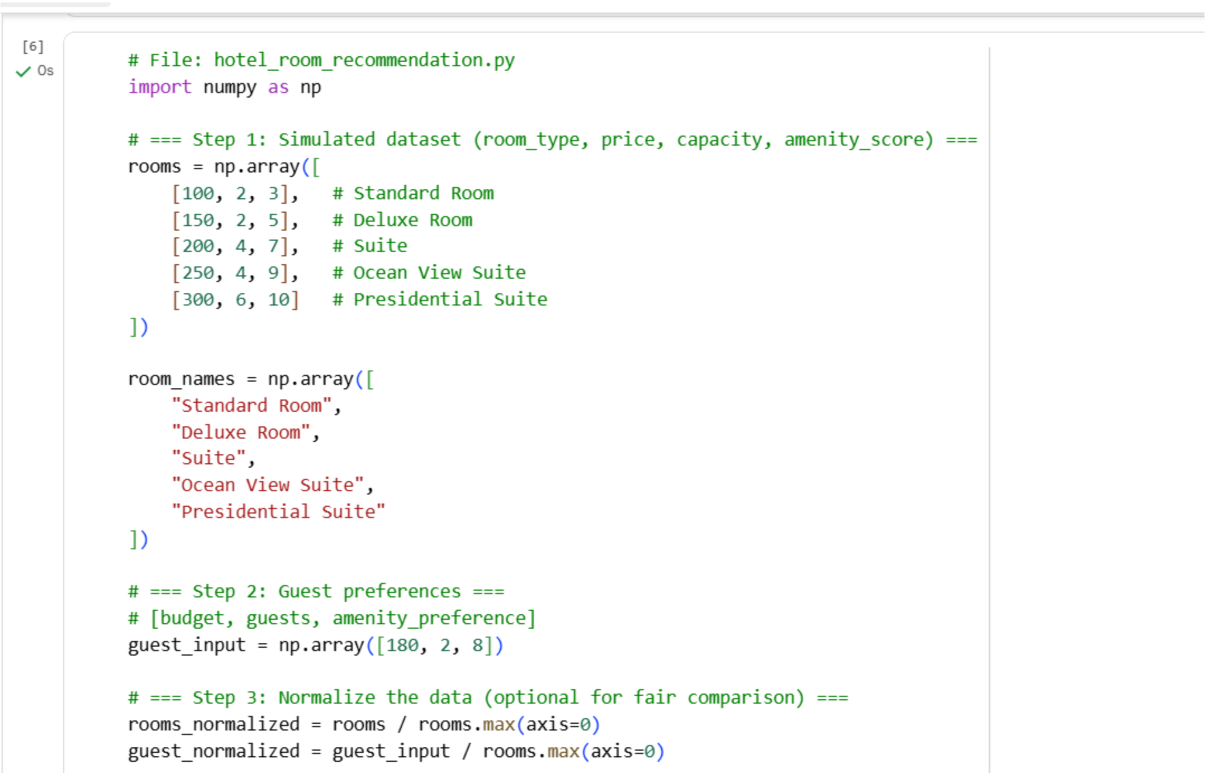
QUESTION 2:

Scenario: In the Hospitality sector, a company faces a challenge related to web frontend development.

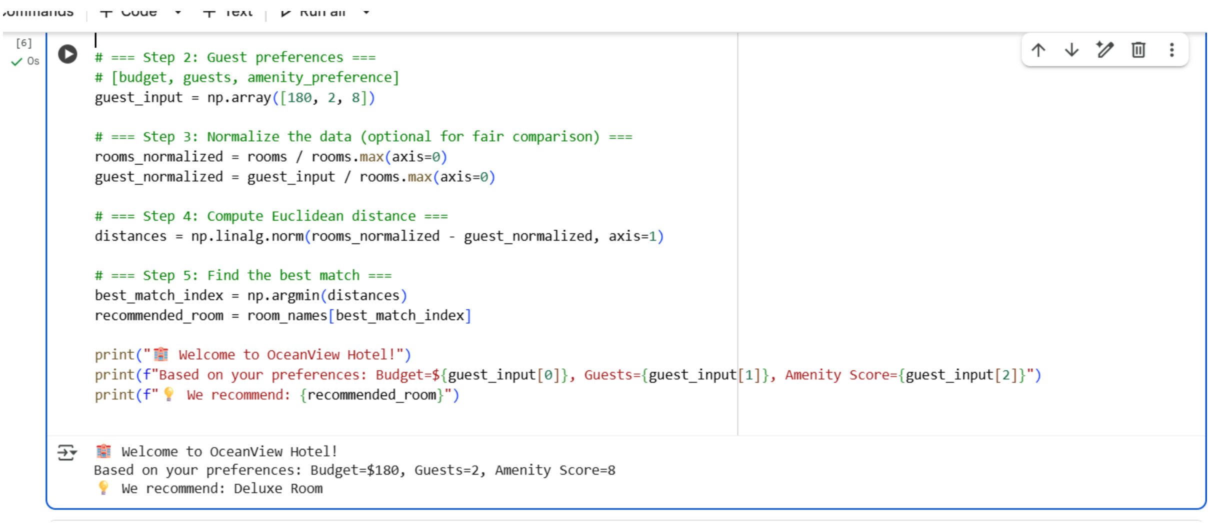
Task: Use AI-assisted tools to solve a problem involving web frontend development in this context.

Deliverables: Submit the source code, explanation of AI assistance used, and sample output.

CODE AND OUTPUT:



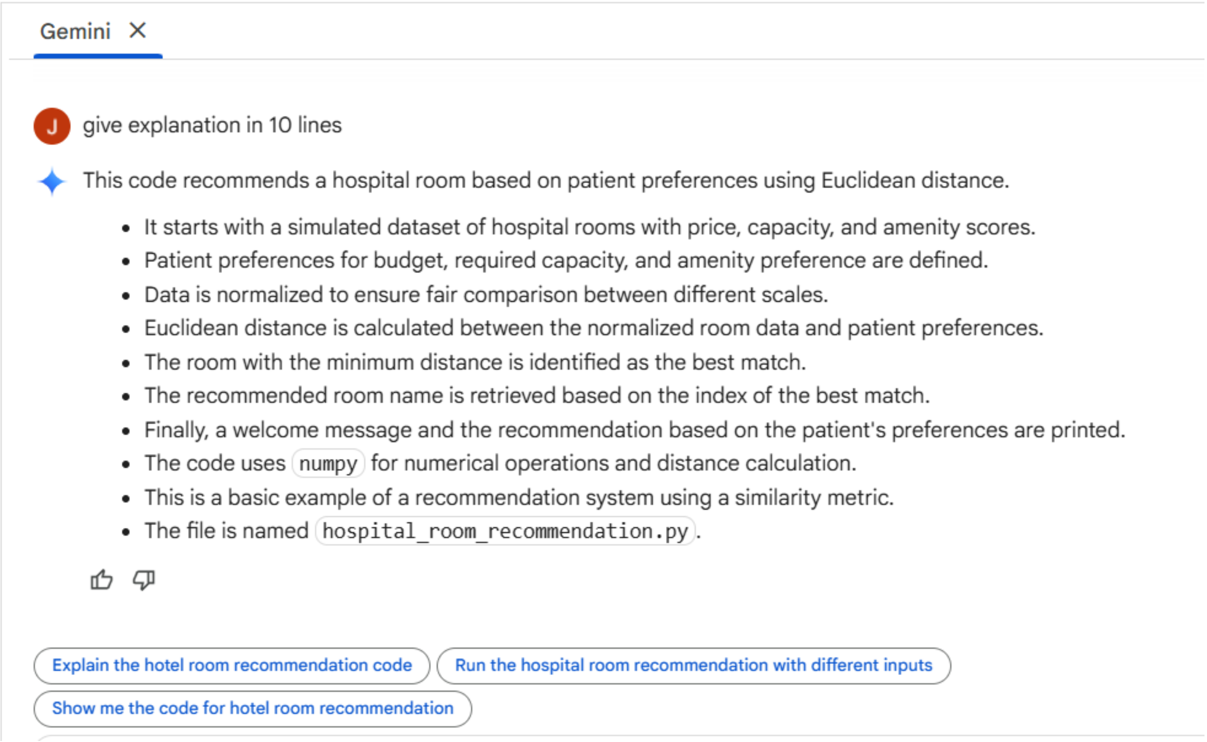
```python
# File: hotel_room_recommendation.py
import numpy as np

# === Step 1: Simulated dataset (room_type, price, capacity, amenity_score) ===
rooms = np.array([
    [100, 2, 3],   # Standard Room
    [150, 2, 5],   # Deluxe Room
    [200, 4, 7],   # Suite
    [250, 4, 9],   # Ocean View Suite
    [300, 6, 10]   # Presidential Suite
])

room_names = np.array([
    "Standard Room",
    "Deluxe Room",
    "Suite",
    "Ocean View Suite",
    "Presidential Suite"
])

# === Step 2: Guest preferences ===
# [budget, guests, amenity_preference]
guest_input = np.array([180, 2, 8])

# === Step 3: Normalize the data (optional for fair comparison) ===
rooms_normalized = rooms / rooms.max(axis=0)
guest_normalized = guest_input / rooms.max(axis=0)
```



```python
# === Step 2: Guest preferences ===
# [budget, guests, amenity_preference]
guest_input = np.array([180, 2, 8])

# === Step 3: Normalize the data (optional for fair comparison) ===
rooms_normalized = rooms / rooms.max(axis=0)
guest_normalized = guest_input / rooms.max(axis=0)

# === Step 4: Compute Euclidean distance ===
distances = np.linalg.norm(rooms_normalized - guest_normalized, axis=1)

# === Step 5: Find the best match ===
best_match_index = np.argmin(distances)
recommended_room = room_names[best_match_index]

print("🏨 Welcome to OceanView Hotel!")
print(f"Based on your preferences: Budget=${guest_input[0]}, Guests={guest_input[1]}, Amenity Score={guest_input[2]}")
print(f"💡 We recommend: {recommended_room}")
```

```
🏨 Welcome to OceanView Hotel!
Based on your preferences: Budget=$180, Guests=2, Amenity Score=8
💡 We recommend: Deluxe Room
```

EXPLANATION :



give explanation in 10 lines

This code recommends a hospital room based on patient preferences using Euclidean distance.

- It starts with a simulated dataset of hospital rooms with price, capacity, and amenity scores.
- Patient preferences for budget, required capacity, and amenity preference are defined.
- Data is normalized to ensure fair comparison between different scales.
- Euclidean distance is calculated between the normalized room data and patient preferences.
- The room with the minimum distance is identified as the best match.
- The recommended room name is retrieved based on the index of the best match.
- Finally, a welcome message and the recommendation based on the patient's preferences are printed.
- The code uses `numpy` for numerical operations and distance calculation.
- This is a basic example of a recommendation system using a similarity metric.
- The file is named `hospital_room_recommendation.py`.

Explain the hotel room recommendation code
Run the hospital room recommendation with different inputs
Show me the code for hotel room recommendation