Exercise No. 2			
Topic:	**Topic 2.1: Introduction to Probability in AI**	**Week No.**	5-6
Course Code:	**CSST101**	**Term:**	1st Semester
Course Title:	**Advance Representation and Reasoning**	**Academic Year:**	2024-2025
Student Name		**Section**	
Due date		**Points**	

Exercise 1: Introduction to Probability Theory in AI

Objective: Understand the fundamentals of probability theory and its application in AI for decision-making under uncertainty.

Tasks:

1. **Basic Probability Calculation**: Write Python functions that calculate the following:
   - **Joint Probability**: The probability of two events happening together.
   - **Marginal Probability**: The probability of an individual event, summing over all possible outcomes.
   - **Conditional Probability**: The probability of an event given that another event has occurred.

Example:

```python
def joint_probability(p_A, p_B):
    return p_A * p_B

def marginal_probability(p_A, p_B):
    return p_A + p_B - joint_probability(p_A, p_B)

def conditional_probability(p_B_given_A, p_A, p_B):
    return (p_B_given_A * p_A) / p_B

# Example usage
p_A = 0.3  # Probability of event A
p_B = 0.4  # Probability of event B
p_B_given_A = 0.8  # Conditional probability of B given A

print(f"Joint Probability: {joint_probability(p_A, p_B)}")
print(f"Marginal Probability: {marginal_probability(p_A, p_B)}")
print(f"Conditional Probability: {conditional_probability(p_B_given_A, p_A, p_B)}")
```

Assessment Task 1: Basic Probability Calculations

1. Start with a Coding Task:

- Write Python functions to compute **joint probability**, **marginal probability**, and **conditional probability**.

2. Interactive Question:

- **Question**: If the probability of event A is 0.3 and the probability of event B is 0.4, what is the joint probability of A and B assuming they are independent?

- **Answer using Code**: Implement the function joint_probability(p_A, p_B) to compute the result and submit it to see if your code is correct.

3. Feedback:

- If correct, you'll receive immediate feedback that explains the formula you used.
- If incorrect, hints will be provided to guide you toward the correct implementation.

Exercise 2: Decision-Making Under Uncertainty

Objective: Apply probability theory to make decisions in uncertain environments, such as AI systems that need to weigh the outcomes of different choices.

Tasks:

1. **Simulate a Decision Process**: Write a Python script that simulates a decision-making process where the outcomes are uncertain.
    - Model the probability of success and failure for a decision.
    - Calculate the **expected value** of making a particular decision.

2. **Decision Tree Example**: Create a decision tree where each branch represents a different action and associated probabilities for outcomes (profit or loss).
    - Example: Simulate a robot making decisions in an uncertain environment with probabilities for success and failure.

Example:

```
import numpy as np

def simulate_decision(num_simulations, p_success, reward_success, reward_failure):
    outcomes = []
    for _ in range(num_simulations):
        if np.random.rand() < p_success:
            outcomes.append(reward_success)
        else:
            outcomes.append(reward_failure)
    return np.mean(outcomes)

# Example: Simulating 1000 decision attempts
p_success = 0.7  # Probability of success
reward_success = 1000  # Reward for success
reward_failure = -500  # Penalty for failure

average_outcome = simulate_decision(1000, p_success, reward_success, reward_failure)
print(f"Expected value of decision: {average_outcome}")
```

Assessment Task 2: Bayesian Inference

1. **Interactive Bayesian Model**:
    - Implement a Python function that performs **Bayesian inference** to update the probability of a hypothesis based on new evidence.
    - For this task, you'll be asked to compute the probability of having a disease given a positive test result.

2. **Interactive Scenario**:
    - **Scenario**: A test for a disease has a 90% accuracy for positive results, and 5% of people without the disease test positive. If 1% of the population has the disease, what is the probability that someone who tested positive actually has the disease?

3. **Submit Your Answer**:
    - Use your Bayesian inference function to compute the result and submit it.
    - **Interactive Feedback**: After submitting, you'll receive feedback explaining Bayes' Theorem and the role of prior probability, likelihood, and evidence.

Exercise 3: Applying Probability Theory in AI for Diagnosis

Objective: Use probabilistic reasoning to model uncertainty in a real-world scenario such as medical diagnosis.

Tasks:

1. Medical Diagnosis Example: Model a scenario where an AI must diagnose a disease based on test results.

- Define the **prior probabilities** of having the disease.
- Define the **likelihood** of the test being positive given the presence or absence of the disease.
- Use **Bayes' Theorem** to calculate the **posterior probability** of the disease given a positive test result.

Example:

```python
def bayesian_inference(prior, likelihood_positive_given_disease, likelihood_positive):
    posterior = (likelihood_positive_given_disease * prior) / likelihood_positive
    return posterior

# Example probabilities
prior_disease = 0.01  # Probability of having the disease (prior)
likelihood_positive_given_disease = 0.9  # Likelihood of a positive test given the disease
likelihood_positive = 0.05  # Probability of a positive test overall

posterior_disease = bayesian_inference(prior_disease, likelihood_positive_given_disease, likelihood_positive)
print(f"Posterior probability of disease given positive test: {posterior_disease}")
```

Assessment Task 3: Decision-Making Under Uncertainty

1. **Interactive Decision Simulation**:
    - You are tasked with simulating a decision-making process using Python. In this simulation, an AI system must choose between two investment options with uncertain outcomes (profit or loss).

2. **Interactive Task**:
    - **Task**: Implement a function that simulates the decision-making process for 1000 iterations. The probability of success is 70%, and the reward for success is $1000, while the penalty for failure is -$500.
    - **Interactive Input**: Adjust the probability of success or failure and simulate different outcomes.

3. **Question**: What is the average return after 1000 iterations? How does changing the probability affect the result?

4. **Submit and Feedback**:
    - Run your simulation and submit the results.
    - Feedback will include an explanation of how probability distributions affect decision-making under uncertainty.

Exercise 4: Probability Distribution in AI

Objective: Work with probability distributions that are key in AI for representing uncertainty.

Tasks:

1. **Simulate Probability Distributions**: Write a Python script that generates data from common probability distributions (e.g., binomial, normal) using numpy or scipy.
2. **Visualize the Distribution**: Use a plotting library (e.g., matplotlib) to visualize the distribution and interpret the results.

Examples:

```python
import numpy as np
import matplotlib.pyplot as plt

# Simulating a binomial distribution (e.g., flipping a coin)
n_trials = 1000
p_head = 0.5
binomial_distribution = np.random.binomial(n=1, p=p_head, size=n_trials)

# Plotting the distribution
plt.hist(binomial_distribution, bins=2)
plt.title('Binomial Distribution (Coin Flips)')
plt.show()
```

Assessment Task 4: Real-World Scenario

1. **Interactive Real-World Problem**:
   - You are asked to model a **real-world scenario** in AI, such as predicting whether it will rain based on factors like humidity and cloud cover. Use conditional probabilities to model this scenario.

2. **Scenario Simulation**:
   - **Task**: Implement a Python script that uses conditional probability to predict the likelihood of rain given evidence (humidity and cloud cover).
   - **Interactive Scenario**: You can change the input conditions (e.g., increase humidity or cloud cover) to see how the probability of rain changes in real-time.

3. **Submit and Feedback**:
   - Submit your prediction and see the outcomes in a dynamically updated plot. Feedback will explain how the evidence you input affects the overall prediction.

Exercise 5: Real-World Application of Probability in AI

Objective: Apply probability theory to real-world AI scenarios, such as predicting customer behavior or making decisions in uncertain markets.

Tasks:

1. **Customer Behavior Prediction**: Suppose you are working for an e-commerce company and need to predict the probability that a customer will make a purchase based on previous behavior.
    - Define probabilities for different user actions (e.g., clicking on a product, adding to cart, making a purchase).
    - Use **conditional probability** to predict the likelihood of a purchase given that a customer added an item to their cart.

Example:

```
# Example probabilities based on observed customer behavior
p_purchase_given_cart = 0.4  # Conditional probability of purchase given an item in the cart
p_cart = 0.3  # Probability of adding an item to the cart

# Calculating the probability of a customer making a purchase
p_purchase = conditional_probability(p_purchase_given_cart, p_cart, 1)
print(f"Probability of purchase: {p_purchase}")
```

Assessment Task 5: Probability Distributions Visualization

1. **Interactive Visualization Task**:
   - Write Python code to generate and visualize **binomial** and **normal** probability distributions.

2. **Interactive Question**:
   - **Task**: Simulate 1000 coin flips using a binomial distribution and visualize the outcomes.
   - **Follow-Up**: How does increasing the number of trials or changing the probability of heads affect the distribution?

3. **Interactive Graph**:
   - Submit your code, and a real-time plot of the distribution will be generated. You can adjust the number of trials or probabilities and see how the graph updates.
   - **Feedback**: Receive instant feedback on your graph, explaining how different distribution parameters affect the outcome.

These exercises will help you understand the fundamentals of **Probability Theory in AI** and how it is applied to make decisions under uncertainty, which is crucial for intelligent systems such as recommendation engines, medical diagnosis models, and decision-making robots.

Submission Format:

- Upload your Python scripts and Notebook to your GitHub repository.
- Ensure the repository is well-organized, with folders and files clearly labeled (e.g., scripts/, notebooks/, README.md)
- Filename [SECTION]-[SURNAME]-EXER2 e.g. 3A-BERNARDINO-EXER2

Rubric for Introduction to Probability in AI Exercises

Criteria	Excellent (10 points)	Good (8 points)	Fair (5 points)	Poor (2 points)
Correctness of Implementation	All probability functions, Bayesian inference, and simulations are correctly implemented, producing accurate and expected results for all tasks.	Most functions and models are correctly implemented with minor errors or omissions.	Basic implementation of the tasks, but significant errors or missing key components.	Incorrect or incomplete implementation of functions, models, or simulations.
Application of Concepts	Demonstrates a deep understanding of probability theory, Bayesian inference, and decision-making under uncertainty, applying these concepts correctly in each exercise.	Demonstrates a good understanding with minor conceptual gaps, but overall applies concepts correctly.	Shows basic understanding, but there are noticeable gaps or confusion about core concepts.	Little or no understanding of probability theory and decision-making under uncertainty.
Code Quality and Structure	Code is well-organized, clean, easy to read, and includes appropriate comments. Efficient use of functions and libraries.	Code is organized and functional, but may lack clarity or sufficient comments in some parts.	Code runs but is somewhat disorganized, lacks comments, or is inefficient.	Code is disorganized, difficult to follow, and lacks necessary comments or structure.
Problem Solving and Creativity	Creative and effective problem-solving is demonstrated. Simulations and scenarios reflect thoughtful choices and modeling.	Demonstrates good problem-solving skills and appropriate choices in modeling scenarios, with minor gaps in creativity.	Basic problem-solving approach; scenarios lack depth or are overly simplified.	Little effort in problem-solving; simulations or scenarios are unrealistic or poorly thought out.
Documentation and Explanation	Comprehensive, clear, and well-documented explanations in code comments and any written explanations (e.g., Google Colab notebooks). The rationale for choices is well-articulated.	Good explanations with minor gaps in detail or clarity. Most choices are explained but lack depth in some areas.	Limited or unclear explanations. Rationale for choices is not fully articulated.	Poor or missing explanations for code and choices. Little to no effort in explaining or documenting.
Simulations and Real-World Application	Simulations are highly realistic, logically sound, and reflect a deep understanding of decision-making under uncertainty. Application to real-world problems is accurate and thoughtful.	Simulations are adequate and logically sound but lack depth or complexity. Real-world applications are correct but basic.	Simulations are limited and may contain logical errors or simplifications. Real-world applications are shallow or incorrect.	Simulations are unrealistic, ineffective, or missing. No real-world applications, or applications are incorrect.