RIPHAH
INTERNATIONAL
UNIVERSITY

Artificial Intelligence

Lab 09 Tasks

Name: Dua Amir

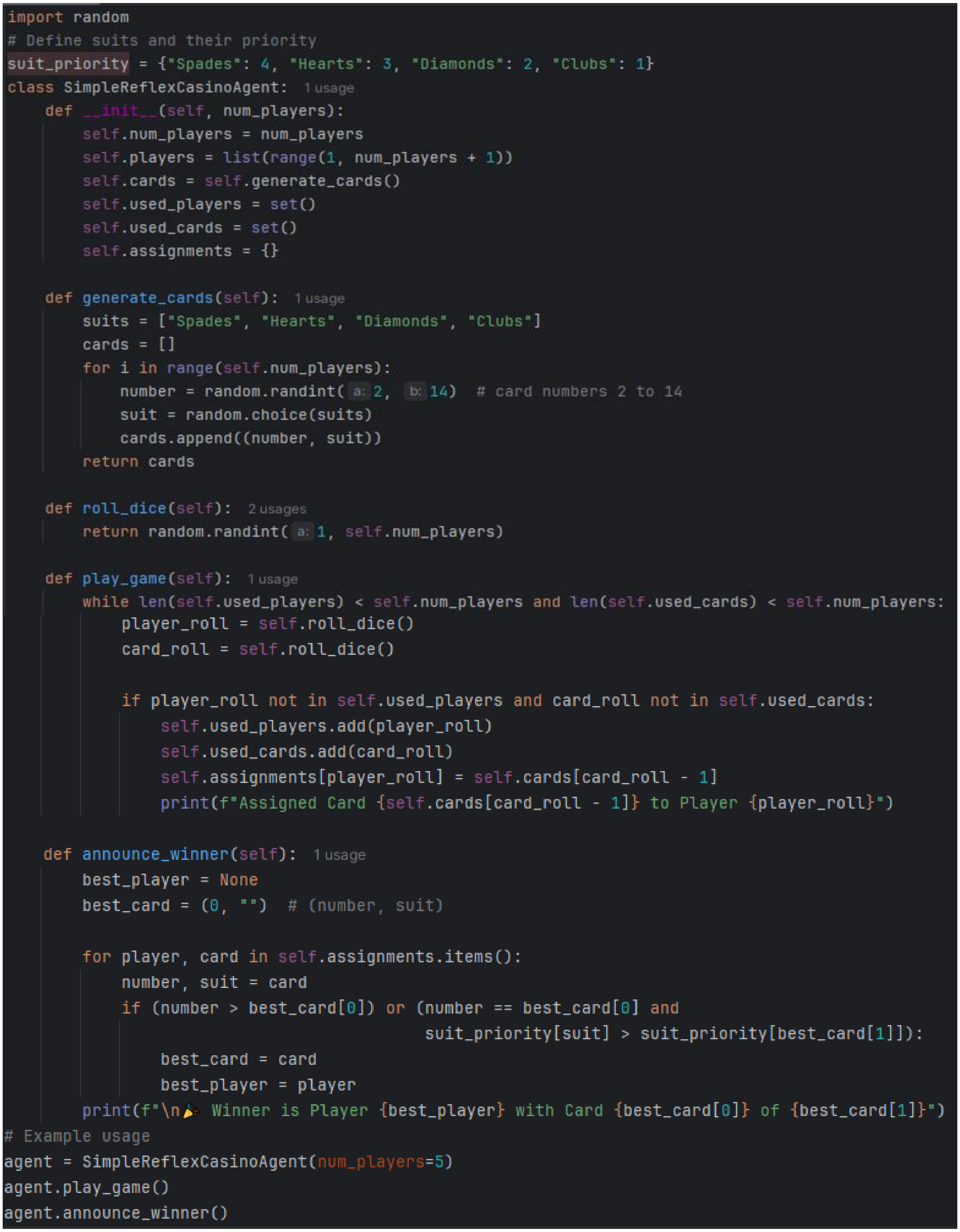
Sap ID: 47849

Batch: BSCS-6th semester

Lab Instructor:

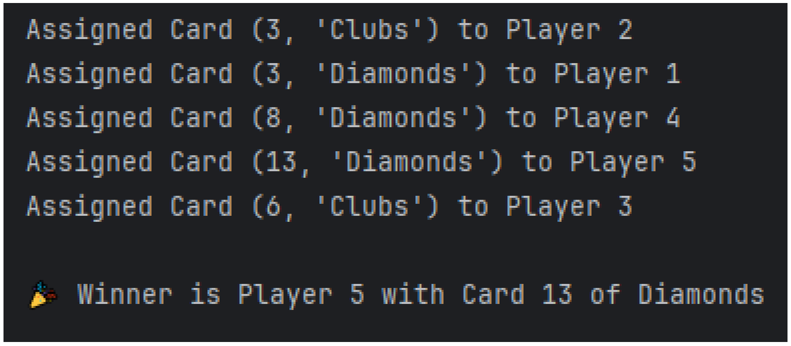
Ayesha Akram

Task2.
Solution:



```
import random
# Define suits and their priority
suit_priority = {"Spades": 4, "Hearts": 3, "Diamonds": 2, "Clubs": 1}
class SimpleReflexCasinoAgent:  1 usage
    def __init__(self, num_players):
        self.num_players = num_players
        self.players = list(range(1, num_players + 1))
        self.cards = self.generate_cards()
        self.used_players = set()
        self.used_cards = set()
        self.assignments = {}

    def generate_cards(self):  1 usage
        suits = ["Spades", "Hearts", "Diamonds", "Clubs"]
        cards = []
        for i in range(self.num_players):
            number = random.randint( a: 2,  b: 14)  # card numbers 2 to 14
            suit = random.choice(suits)
            cards.append((number, suit))
        return cards

    def roll_dice(self):  2 usages
        return random.randint( a: 1, self.num_players)

    def play_game(self):  1 usage
        while len(self.used_players) < self.num_players and len(self.used_cards) < self.num_players:
            player_roll = self.roll_dice()
            card_roll = self.roll_dice()

            if player_roll not in self.used_players and card_roll not in self.used_cards:
                self.used_players.add(player_roll)
                self.used_cards.add(card_roll)
                self.assignments[player_roll] = self.cards[card_roll - 1]
                print(f"Assigned Card {self.cards[card_roll - 1]} to Player {player_roll}")

    def announce_winner(self):  1 usage
        best_player = None
        best_card = (0, "")  # (number, suit)

        for player, card in self.assignments.items():
            number, suit = card
            if (number > best_card[0]) or (number == best_card[0] and
                                           suit_priority[suit] > suit_priority[best_card[1]]):
                best_card = card
                best_player = player
        print(f"\n🎉 Winner is Player {best_player} with Card {best_card[0]} of {best_card[1]}")
# Example usage
agent = SimpleReflexCasinoAgent(num_players=5)
agent.play_game()
agent.announce_winner()
```

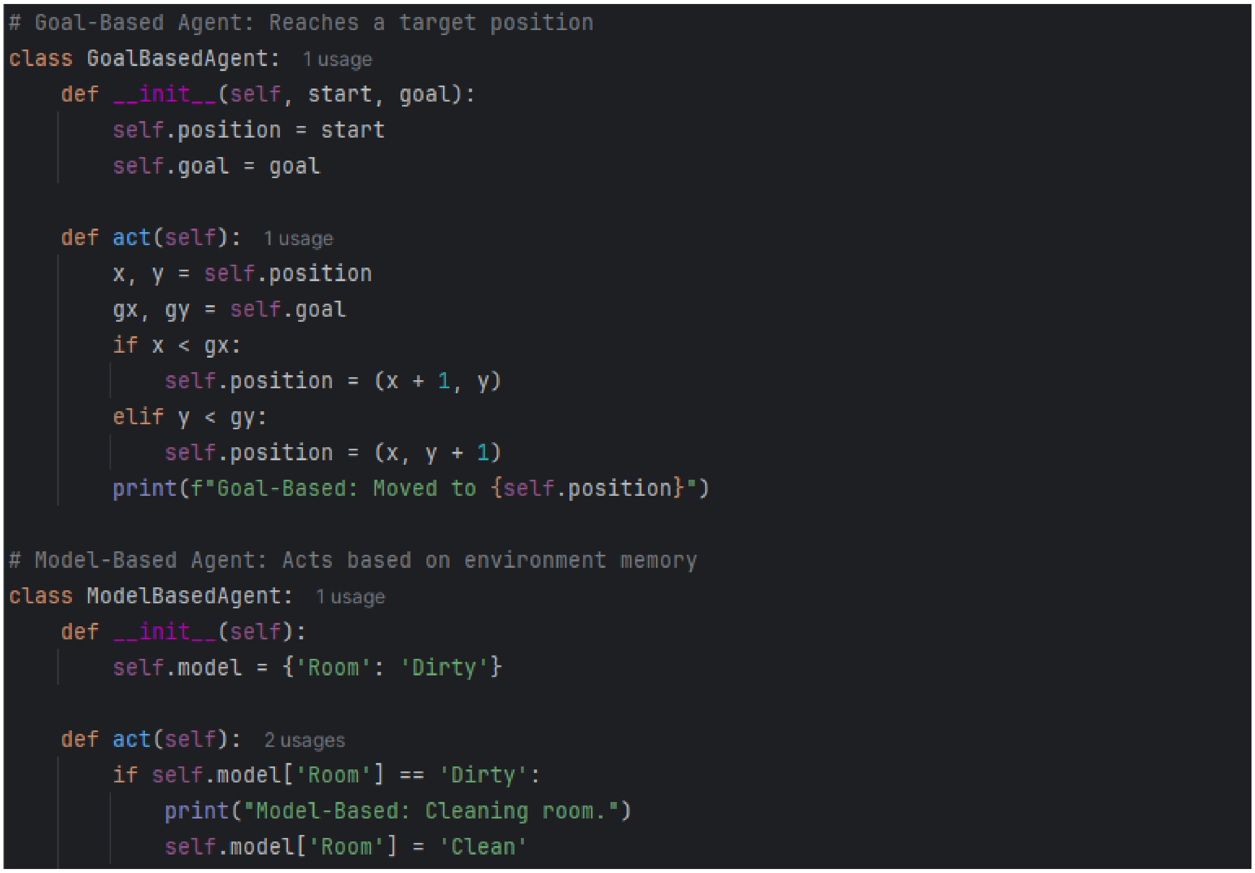
Output:



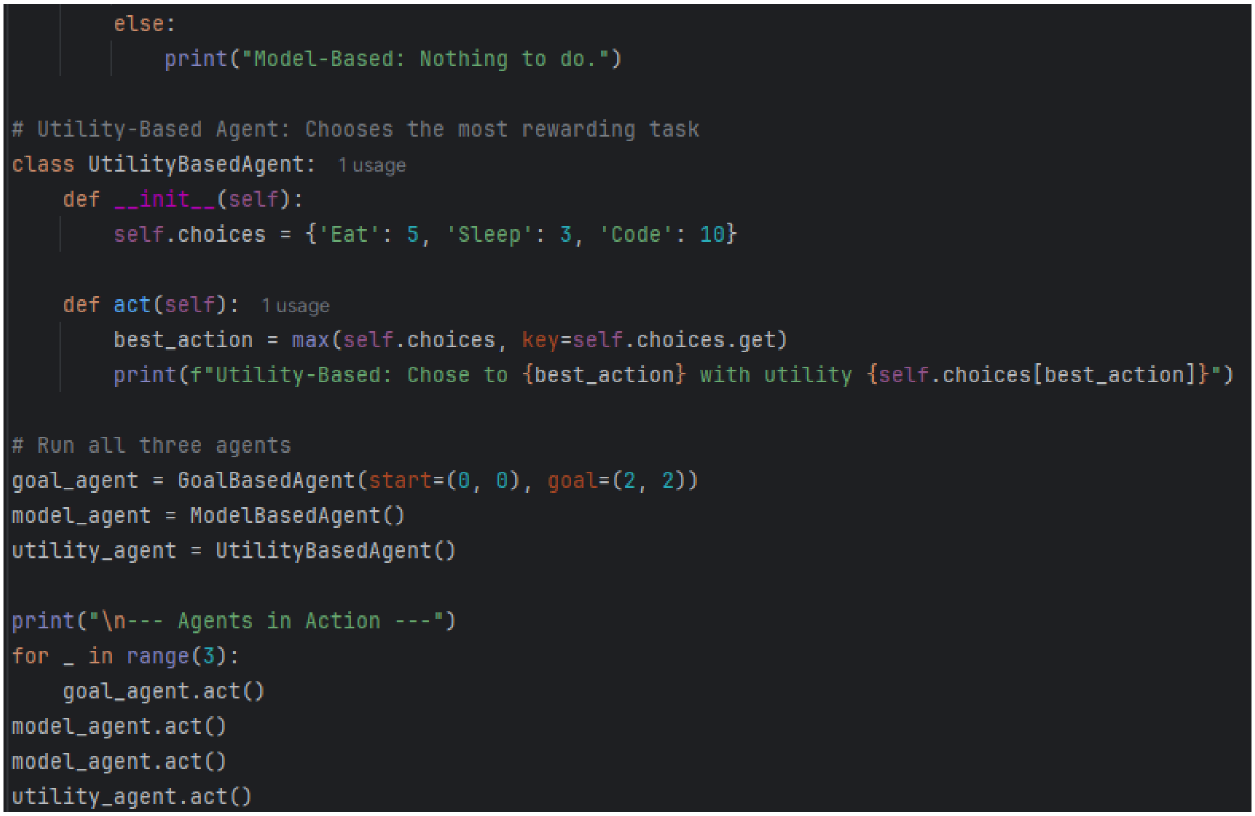
```
Assigned Card (3, 'Clubs') to Player 2
Assigned Card (3, 'Diamonds') to Player 1
Assigned Card (8, 'Diamonds') to Player 4
Assigned Card (13, 'Diamonds') to Player 5
Assigned Card (6, 'Clubs') to Player 3

🎉 Winner is Player 5 with Card 13 of Diamonds
```

Task2.

Solution:

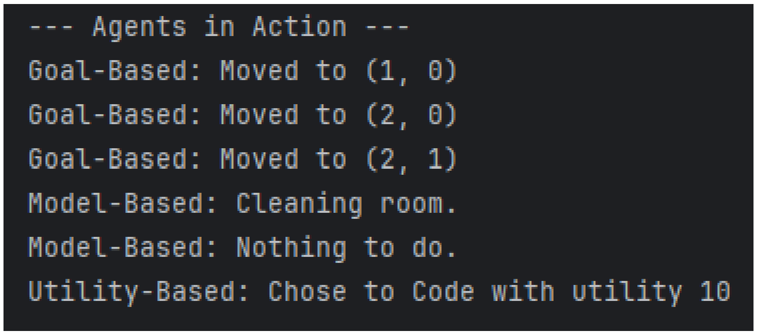


```
# Goal-Based Agent: Reaches a target position
class GoalBasedAgent:  1 usage
    def __init__(self, start, goal):
        self.position = start
        self.goal = goal

    def act(self):  1 usage
        x, y = self.position
        gx, gy = self.goal
        if x < gx:
            self.position = (x + 1, y)
        elif y < gy:
            self.position = (x, y + 1)
        print(f"Goal-Based: Moved to {self.position}")

# Model-Based Agent: Acts based on environment memory
class ModelBasedAgent:  1 usage
    def __init__(self):
        self.model = {'Room': 'Dirty'}

    def act(self):  2 usages
        if self.model['Room'] == 'Dirty':
            print("Model-Based: Cleaning room.")
            self.model['Room'] = 'Clean'
```



```
        else:
            print("Model-Based: Nothing to do.")

# Utility-Based Agent: Chooses the most rewarding task
class UtilityBasedAgent:  1 usage
    def __init__(self):
        self.choices = {'Eat': 5, 'Sleep': 3, 'Code': 10}

    def act(self):  1 usage
        best_action = max(self.choices, key=self.choices.get)
        print(f"Utility-Based: Chose to {best_action} with utility {self.choices[best_action]}")

# Run all three agents
goal_agent = GoalBasedAgent(start=(0, 0), goal=(2, 2))
model_agent = ModelBasedAgent()
utility_agent = UtilityBasedAgent()

print("\n--- Agents in Action ---")
for _ in range(3):
    goal_agent.act()
model_agent.act()
model_agent.act()
utility_agent.act()
```

Output:



```
--- Agents in Action ---
Goal-Based: Moved to (1, 0)
Goal-Based: Moved to (2, 0)
Goal-Based: Moved to (2, 1)
Model-Based: Cleaning room.
Model-Based: Nothing to do.
Utility-Based: Chose to Code with utility 10
```