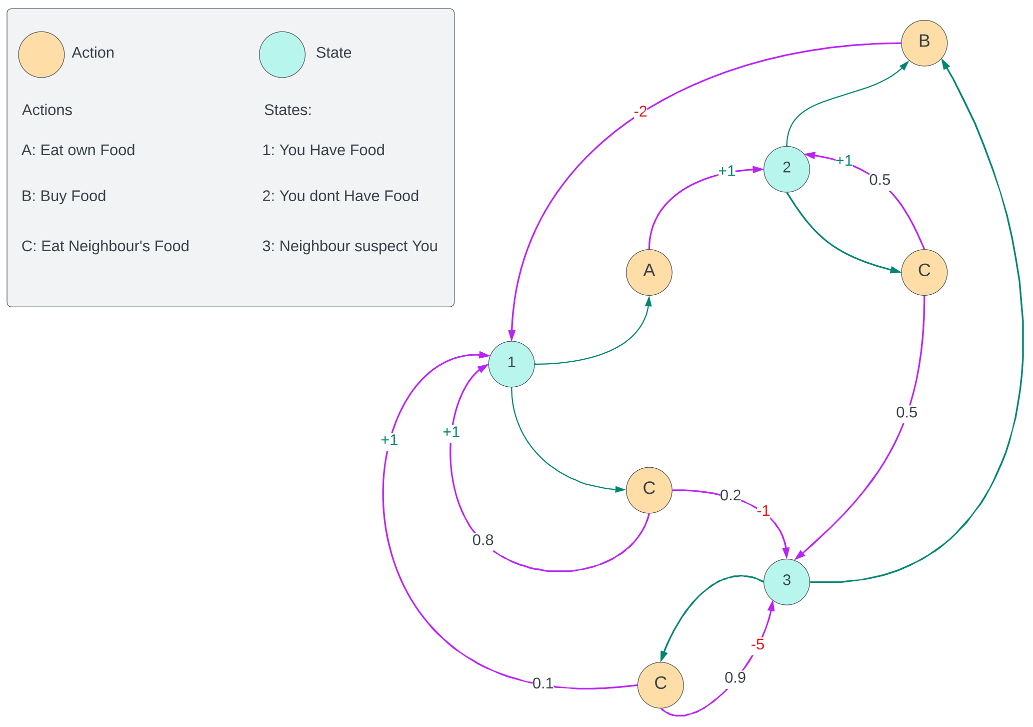
MDP: Student Housing

Overview

This Markov Decision Process (MDP) models designs a scenario revolving around the management of food and interactions with a neighbor in Student Housing. This model captures the constarints and possible actions of decision-making in everyday scenarios.



States

The MDP contains three distinct states reflecting different scenarios a person might face in the context of food management:

1. **You Have Food**: This state represents the scenario where the individual has food to consume.
2. **You Don't Have Food**: This state represents case where the individual has run out of food.
3. **Neighbor Suspects You**: This state represents case where neighbour suspect you stealing his food.

Actions

Actions are defined for each state, determining the transitions and influencing the outcomes based on the choice made:

- *Eat Own Food*: Consume available food.
- *Buy Food*: Buying food.
- *Take Neighbor Food*: Attempting to take food from the neighbor.

Transitions

Transitions between states are determined by the actions taken. Some action in a particular state has an uncertainty of outcome:

- From **You Have Food**:
    - *Eat Own Food* always result in "You Don't Have Food".
    - *Take Neighbor Food* has a 20% chance of resulting in "Neighbor Suspects You" and an 80% chance of resulting in "You Have Food".
- From **You Don’t Have Food**:
    - *Buy Food* always results in "You Have Food".
    - *Take Neighbor Food* has a 50% chance of resulting in "Neighbor Suspects You" and a 50% chance of remaining in "You Have Food".
- From **Neighbor Suspects You**:
    - *Buy Food* transitions the individual back to "You Have Food".
    - *Take Neighbor Food* has a 90% chance of resulting in "Neighbor Suspects You" state and a 10% chance of resulting in "You Have Food".

Rewards

Rewards are associated with transitions between states, based on the actions taken:

- **You Have Food**:
    - *Eat Own Food* leads to a reward of +1.
    - *Take Neighbor Food* could result in a reward of +1if goes to “You Have Food” or a penalty of -1 if "Neighbor Suspects You"

- **You dont Have Food**:
    - *Buy Food* results in a reward of -2.
    - *Take Neighbor Food* could lead to no change (0) or a penalty of -1 if goes to "Neighbor Suspects You"
- **Neighbor Suspects You**:
    - *Buy Food* has no reward of -2.
    - *Take Neighbor Food* can lead to a severe penalty of -5 if goes to "Neighbor Suspects You" or a reward of +1 if goes to "You have Food"

Markov Decision Process (MDP) Model

A **stochastic Markov Decision Process (MDP)** is a mathematical model used to optimize decision-making in environments where outcomes are uncertain and depend on both the actions of a decision-maker and random events.

s defined in mathematical terms by a set of states S, a set of actions A, transition probabilities $P(s' \mid s, a)$, and a reward function $R(s, a, s')$. Here, $P(s' \mid s, a)$, denotes the probability of transitioning to state s' from state s after taking action a, and $R(s, a, s')$. represents the reward received after moving from state s to state s' via action a.

The objective in an MDP is to find a policy π, a function from states to actions, that maximizes the expected cumulative reward. This is typically calculated as the sum of discounted rewards over time, formalized as $E[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t, S_{t+1},)]$where γ is a discount factor that weighs the importance of future rewards.

The transition probabilities determine the likelihood of moving from one state to another given a specific action, and rewards provide a numerical value for each state transition. The objective in an MDP is to find a policy that maximizes the expected sum of rewards over time.

Initialization and Configuration

The **MDP** class is initialized with five main components: states, actions, transitions, rewards, and current state.

```python
class MDP:
    def __init__(self, states, actions, transitions, rewards, current_state=None):
        self.states = states
        self.actions = actions
        self.transitions = transitions
        self.rewards = rewards
        if current_state is None:
            self.current_state = random.choice(list(states.keys()))
        else:
            self.current_state = current_state
```

1. **States**: Represents the different conditions that can exist within the environment as a list. In this case:

```python
states = [
    'You have Food',
    'You dont have Food',
    'Neighbour suspect you'
]
```

2. **Actions**: Specifies possible actions that can be taken in each state:

```python
actions = {
    'You have Food': ['Eat own food', 'Take neighbour Food'],
    'You dont have Food': ['Buy Food', 'Take neighbour Food'],
    'Neighbour suspect you': ['Buy Food', 'Take neighbour Food']
}
```

3. **Transitions**: Defines the probability of moving from one state to another given a specific action:

```python
transitions = {
    'You have Food': {
        'Eat own food': {'You dont have Food': 1.0},
        'Take neighbour Food': {'Neighbour suspect you': 0.2, 'You have Food': 0.8}
    },
    'You dont have Food': {
        'Buy Food': {'You have Food': 1.0},
        'Take neighbour Food': {'Neighbour suspect you': 0.5, 'You have Food': 0.5}
    },
    'Neighbour suspect you': {
        'Buy Food': {'You have Food': 1.0},
        'Take neighbour Food': {'Neighbour suspect you': 0.9, 'You have Food': 0.1}
    }
}
```

4. **Rewards**: Maps each state-action-next state triplet to a numerical reward, quantifying the immediate value of a transition:

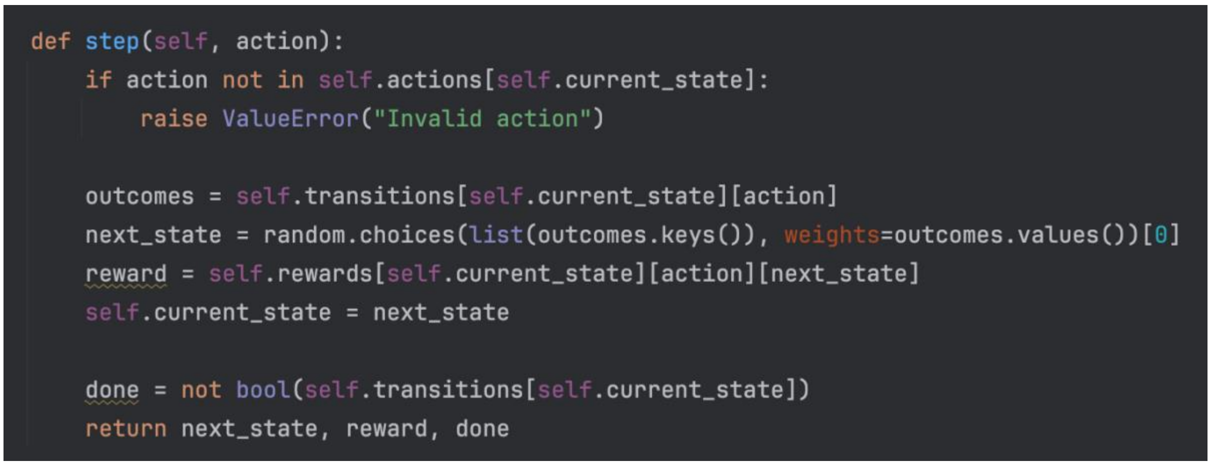
```python
rewards = {
    'You have Food': {
        'Eat own food': {'You dont have Food': 1},
        'Take neighbour Food': {'Neighbour suspect you': -1, 'You have Food': 1}
    },
    'You dont have Food': {
        'Buy Food': {'You have Food': -2},
        'Take neighbour Food': {'Neighbour suspect you': -1, 'You have Food': 0}
    },
    'Neighbour suspect you': {
        'Buy Food': {'You have Food': -2},
        'Take neighbour Food': {'Neighbour suspect you': -5, 'You have Food': 1}
    }
}
```

Methods

- **reset()**: Resets the MDP to a random initial state.

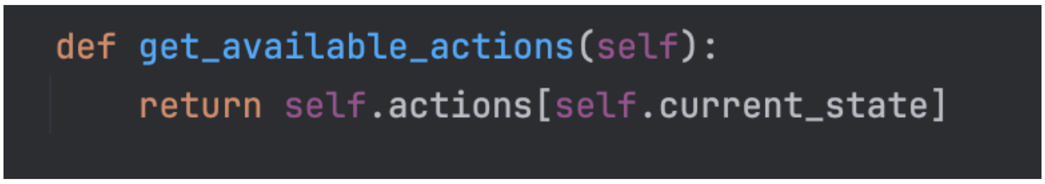
```python
def reset(self):
    self.current_state = random.choice(self.states)
    return self.current_state
```

- **step(action)**: Takes an action, determines the next state from current state based on the transition probabilities, updates the current state, and returns the new state, reward, and whether the state is terminal.



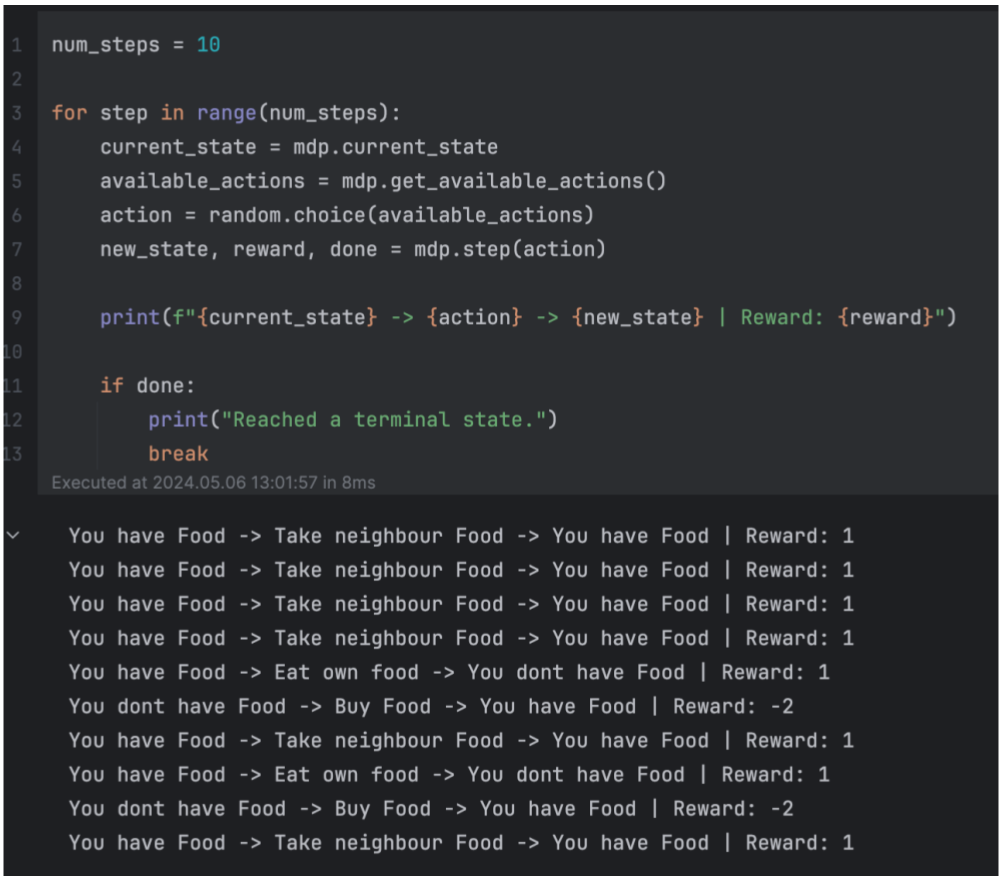
```python
def step(self, action):
    if action not in self.actions[self.current_state]:
        raise ValueError("Invalid action")

    outcomes = self.transitions[self.current_state][action]
    next_state = random.choices(list(outcomes.keys()), weights=outcomes.values())[0]
    reward = self.rewards[self.current_state][action][next_state]
    self.current_state = next_state

    done = not bool(self.transitions[self.current_state])
    return next_state, reward, done
```

- **get_available_actions()**: Returns a list of valid actions that can be taken from the current state.



```python
def get_available_actions(self):
    return self.actions[self.current_state]
```

Stochastic Process Example



```python
num_steps = 10

for step in range(num_steps):
    current_state = mdp.current_state
    available_actions = mdp.get_available_actions()
    action = random.choice(available_actions)
    new_state, reward, done = mdp.step(action)

    print(f"{current_state} -> {action} -> {new_state} | Reward: {reward}")

    if done:
        print("Reached a terminal state.")
        break
```

Executed at 2024.05.06 13:01:57 in 8ms

```
You have Food -> Take neighbour Food -> You have Food | Reward: 1
You have Food -> Take neighbour Food -> You have Food | Reward: 1
You have Food -> Take neighbour Food -> You have Food | Reward: 1
You have Food -> Take neighbour Food -> You have Food | Reward: 1
You have Food -> Eat own food -> You dont have Food | Reward: 1
You dont have Food -> Buy Food -> You have Food | Reward: -2
You have Food -> Take neighbour Food -> You have Food | Reward: 1
You have Food -> Eat own food -> You dont have Food | Reward: 1
You dont have Food -> Buy Food -> You have Food | Reward: -2
You have Food -> Take neighbour Food -> You have Food | Reward: 1
```

Conclusion

This MDP for Student Housing provides a clear model for predicting the best strategy in managing food resources and interactions with neighbors. The model defines possible states and actions, making it possible to evaluate the outcomes of various decisions under uncertainty and to strategize effectively for optimal results in a shared living environment.