MDP Assignment

# Assignment 1

## 1.1 Reflect on how versions 1 and 2 can deal with this case. See course slides for detail on the MDP.

A diagram of a number of squares

Description automatically generated with medium confidence

#### States:

Where Color1, Color2, and Color3 are discrete variables that represent 3 colors.

#### Actions

Where are discrete actions for each state.

#### Rewards

Where -1,0,1 are discrete rewards.

### Version1

Version 1 is well designed for This MDP scenario because the **environment** defines the stochastic model for the next **state** and **reward** based on the **current state** and **action**.

This means that the table in Figure defines the probability of getting to any **state** and receiving the **reward**.

### Version2

Version 2 is not that suitable for this scenario. In such a version we have

Which represents the probability of the **next state** based on the **current state** and **taken action**. To calculate these probabilities based on Table:

In this version the **reward** is defined as a deterministic function:

But we can’t define this **reward** in a deterministic way for this **environment**.

Hence this case couldn’t be represented with this version.

## 1.2 Implement a generic stochastic MDP in Python (version 2).

A **stochastic Markov Decision Process (MDP)** is a mathematical model used to make decision-making in environments where outcomes depend on actions with some uncertainty.

MDP is defined in mathematical terms by a set of states , a set of actions , and ) (or transition probabilities ), and a reward function )).

Here, ), denotes the probability of transitioning to state from state after taking action , and getting reward .

The objective in an MDP is to find a policy, a function from states to actions, that maximizes the Value function() with policy . the Value function() is typically calculated as the expected return . The return is sum of all the future rewards, formalized as:

or

Where is the discount factor, which shows how we should weight the future rewards. The nearest rewards will be more significant for the model.

The transition probabilities determine the likelihood of moving from one state to another and receive reward given a specific action and state. The objective of an MDP is to find a policy that maximizes the expected sum of rewards over time at each state:

## Initialization and Configuration

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

A computer screen shot of a code

Description automatically generated

1. **States**: Represents the different conditions that can exist within the environment as a list.
2. **Terminal States**: Represents the terminal (end) state of the Model.
3. **Transitions**: Defines actions and it’s the probability of moving from one state to another and receiving a reward given a specific action.

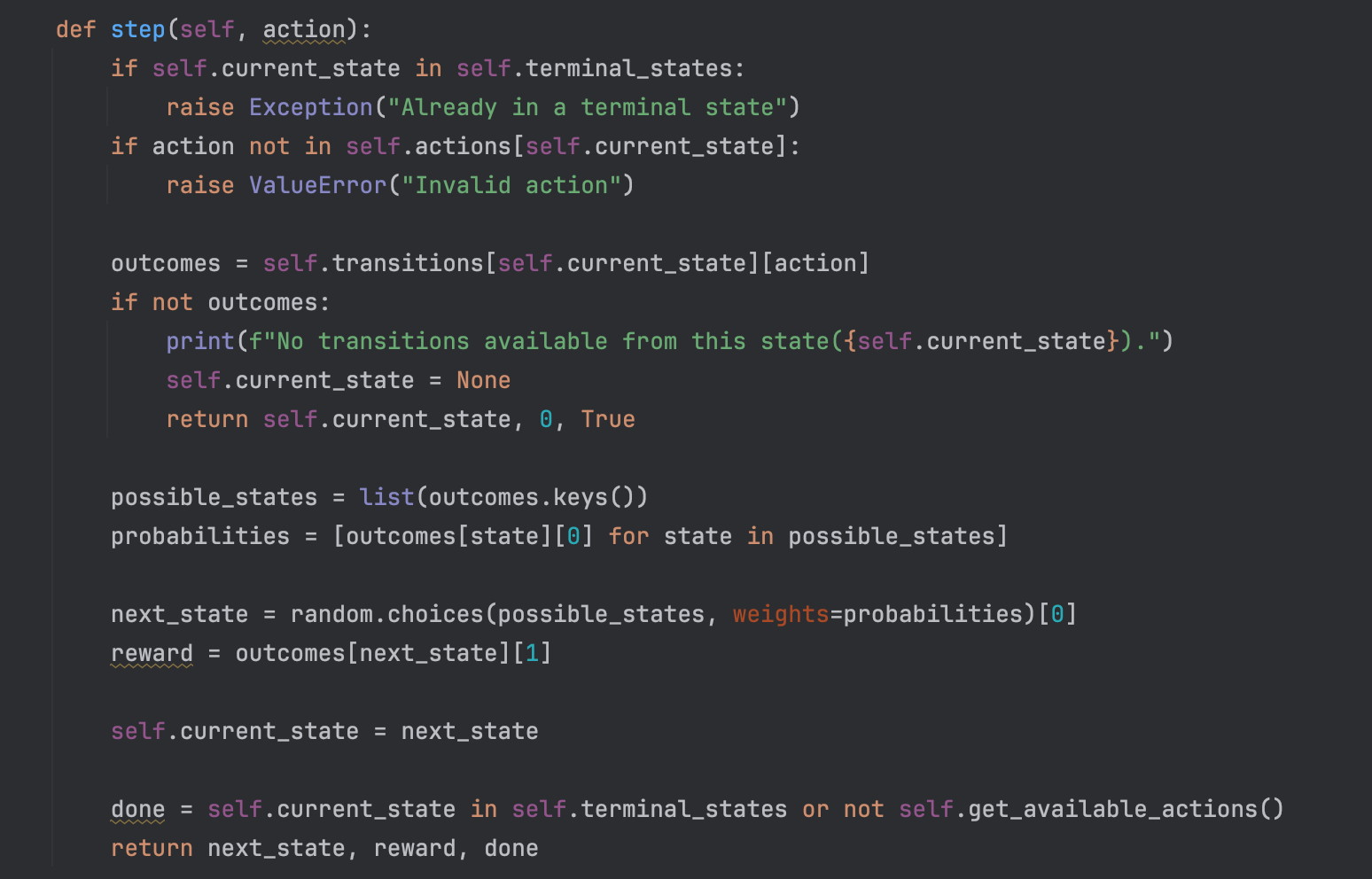
## Methods

* **reset()**: Resets the MDP to a random initial state within not terminal.

A black background with white text

Description automatically generated

* **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.



* **get\_available\_actions()**: Returns a list of valid actions that can be taken from the current state.

A black background with white text

Description automatically generated

<https://chat.openai.com/share/3ebc8fd4-23fe-46de-b68c-3e9bf261b4a1>

ChatGPT was mostly used to debug issues and implement mostly Python features and functionality, the MDP logic was done with knowledge from lectures and presentations.

# 1.3 Implement the MDP in the image

A diagram of a diagram

Description automatically generated

#### States:

Where are discrete variables that represent 3 states.

A screenshot of a cell phone

Description automatically generated

#### Actions

Where are discrete actions for each state.

#### Rewards

Where -1,5 are discrete rewards.

A screenshot of a computer code

Description automatically generated

Here transitions table define

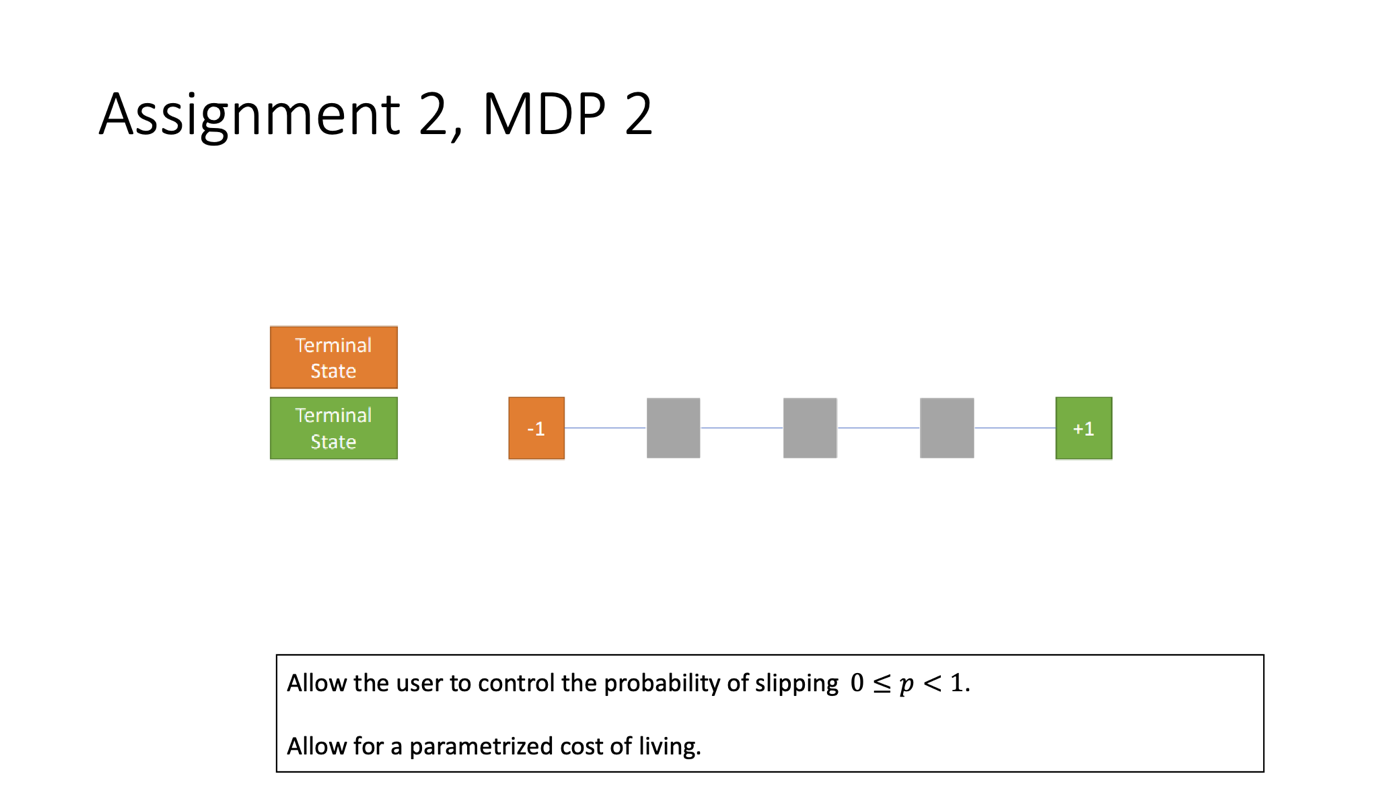
### MDP Test Run

A screenshot of a computer program

Description automatically generated

# Assignment 2

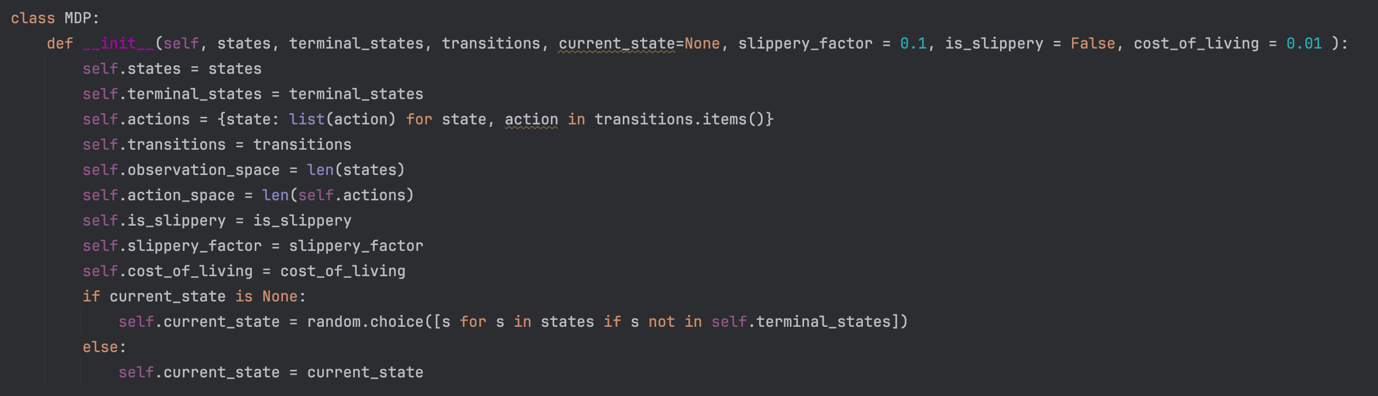
## Assignment 2, MDP 2



This scenario including Slipping and Cost of living require extension to generic MDP Code:

The slippery and cost of living is implemented inside enviornment and could be defined as hyperparameter

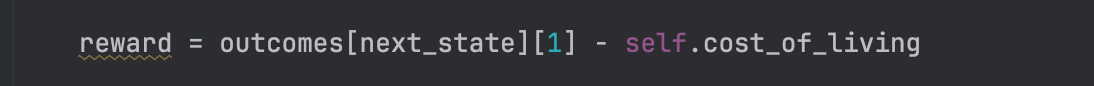
* Add new parameters is\_slippery, slippery\_factor, and cost\_of\_living to Initialization



* Add logic inside step function to take random step from avaliable is Slippery is enabled with defined probability

A screen shot of a computer

Description automatically generated

* Add logic inside step function to decrement reward by cost 0f living at each step  
    
  

#### States:

Where are discrete states inside the grid(starting from left).

#### Actions

Where are discrete actions for each state, except bound left and bound right.

#### Rewards

Where -1,1 are discrete rewards.

Transition table is defined with function:

## 

Or manually coded looks like:

A screen shot of a computer code

Description automatically generated

## MDP2.1 initialization and test run

A screenshot of a computer program

Description automatically generated

Here it is possible to see when agent select (randomly) action at the step 2 the action is executed instead due to slippery factor.

## Assignment 2, MDP 3

A screenshot of a graph

Description automatically generated

The Generic MDP class is capable of capturing such MDP model, we need to create staes and transitions functions , where are size of the grid

#### States

The state is discrete number, the states mapped from top left increasing horizontally and starting from each new row.

A screen shot of a computer code

Description automatically generated

#### Initial State

#### Terminal State

#### Actions

Where are discrete actions.

#### Rewards

Where -1,1 are discrete rewards.

#### Transitions

Transitions are defined with function:

A computer screen shot of a program

Description automatically generated

## MDP2.2 initialization and test run

A screenshot of a computer

Description automatically generated

Here MDP is initialized where are hyperparams to create grid, also here defined the slippery flag and slippery factor and cost of living.

#### Test Run

A screenshot of a computer program

Description automatically generated

Here we could see that enviornment behave properly

## Assignment 2, MDP 4

A screenshot of a computer

Description automatically generated

Generic MDP is still capable of capturing this scenario. I will use twice more states to represent the problem, each state is represented as vector where **key** is 0 if the key has not been obtained and 1 if it has.

#### States

The state is discrete number, the states mapped from top left increasing horizontally and starting from each new row.

All positions except key and terminal positions will have 2 states. Positions mapped in the way that horizonal is represented as numbers from 1 to 5 and key position is 6.

States are created with function (used tupples instead vectors for python):

A screen shot of a computer code

Description automatically generated

A black background with white text

Description automatically generated

#### Terminal States

#### Actions

Where are discrete actions.

#### Rewards

Where -1,1 are discrete rewards.

#### Transitions

#### 

Transitions are defined with this function and result is:

A screen shot of a computer code

Description automatically generated

Also this is good idea to give some small reward to the key obtain, that will especially help with agent learning, so agent will be likely going for the key to go for some reward, faster, than randomly be the case where it take the key and then open the door and receive reward only in such scenarion. So agent will learn faster.

## MDP2.4 initialization and test run

A screenshot of a computer program

Description automatically generated

In this certain scenarion I decided to choose initial state to (3,0) as middle of the map without key, to check the behaviour of enviornment on practice.

#### Test Run

A screenshot of a computer program

Description automatically generated