University at Buffalo
Department of Computer Science and Engineering
CSE 368: Programming Assignment 2: Teach a taxi to Drive
Due date: 04 November 2025 at 11:59 pm

What Is OpenAI Gymnasium?

Gymnasium is a Python library that gives you **ready-made environments** for training and testing reinforcement learning agents.

Why it's useful:

- You don't have to build a game or world from scratch.
- It gives you an **easy interface** to try actions, get feedback, and track progress.
- Supports **hundreds of environments** (grid worlds, video games, robotics).

Environment for This Assignment: Taxi-v3

In `Taxi-v3`:

- You control a **taxi in a grid world**.
- You must **pick up** a passenger from one location and **drop them off** at the destination.
- You get:
    - **+20** for successful drop-off
    - **-1** for every step
    - **-10** for illegal pick/drop action

Your Mission

Your task is to:

1. Use Gymnasium to load the Taxi environment.
2. Train a simple RL agent (a taxi) to perform better over time.
3. Use a **score table** (Q-learning) to help it learn.
4. Show that the taxi **improves its behavior** after training.
5. The starter code will be very helpful, you are required to complete only the TODO sections of the code.

Taxi actions:

Action Number	**Meaning**	**What the Taxi Does**
0	Move South	Taxi moves one cell **down**
1	Move North	Taxi moves one cell **up**

Action Number	Meaning	What the Taxi Does
2	Move East	Taxi moves one cell **right**
3	Move West	Taxi moves one cell **left**
4	Pickup Passenger	Taxi attempts to **pick up** passenger at current location
5	Dropoff Passenger	Taxi attempts to **drop off** passenger at current location

What You Need

- Python 3
- Installed libraries:
- `pip install gymnasium numpy`

Steps to Follow

Step 1: Import libraries and initialize the environment

```
import gymnasium as gym
import numpy as np
import random

env = gym.make("Taxi-v3")
n_states = env.observation_space.n
n_actions = env.action_space.n
```

- This sets up the environment.
- You get the number of possible states and actions.

Step 2: Create the agent's memory (score table)

```
scores = np.zeros((n_states, n_actions))
```

- This table helps the agent **remember** which actions are good in which situations.
- It starts out with all zeros (no knowledge).

Step 3: Train the agent

Train for 2,000 episodes. In each episode:

- Start in a new state.
- Choose either a random action or the best known one.
- Try the action, get a reward.

- Update the score table using:

```
scores[state, action] += alpha * (reward + gamma * best_next - scores[state, action])
```

*this part is significantly different from how we evaluated rewards and other variables in class demo

- `alpha` = how quickly the agent learns
- `gamma` = how much it cares about future rewards

Step 4: Test the trained agent

After training, let the agent follow the **best actions** it has learned and print the total reward it gets.

Also, use:

```
env.render()
```

to visualize the steps.

Assignment Checklist

Your code must:

- Set up and reset the Taxi-v3 environment
- Create and update a score table
- Train the agent using a loop
- Use `epsilon` to allow exploration
- Show performance after training
- Use `env.render()` during testing

Submission Instructions

- Submit your `.py` file.
- Answer the reflection questions in UBLearns.

Grading Rubric (25 pts)

Item	Points
Code runs without errors	5
Correct use of environment and actions	5
Score table implemented and updated	5
Shows agent improvement after training	5
5 reflection questions answered clearly	5

Now answer the MCQ reflection questions in UBlearns. You can submit the python code as a group but every member must attempt the reflection questions individually.