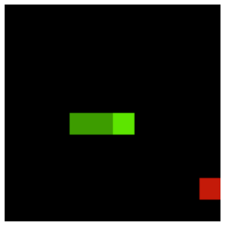
RL on Snake Environment

You may use any resources you like for the following task. However, if you use an LLM to generate any part of the code, please make a note in the section where it was used. This is meant to evaluate your ability, and we want to see your true skills to evaluate if you are a good fit for the lab. From my personal experience, many of the challenges you will encounter in the lab cannot be solved simply by relying on LLMs. Using them to pass this assessment will not be beneficial for either you or us.

Task:

- Implement a simple **Snake game environment** in Python using `Gymnasium`.
  - The **board** is a 10 * 10 grid of squares. The **snake** starts in the middle, moving to the right. Its body is 3 squares long. A piece of **food** appears randomly on any empty square.
  - The snake can move in four directions: **up, right, down, or left**. It cannot instantly turn back into itself (no 180° turns).
  - Each time the snake moves: If it hits the wall or runs into its own body, the game **ends immediately,** and the reward decreases by 10. If the snake's head lands on the food, the snake **grows longer**, the reward increases by 1, and a new piece of food is placed somewhere else.
  - If no food is eaten, the snake just **moves forward**, and its tail square disappears.
  - The observation should be size 3*10*10 (after flattening is 300), where the 3 channels represent
    - **Channel 0**: Snake's body positions (1 where snake occupies, 0 otherwise)
    - **Channel 1**: Food position (1 where food is, 0 otherwise)
    - **Channel 2**: Snake head position (1 where head is, 0 otherwise)
  -  This is an example of what the environment can look like in rendering; You can design it however you want.

- Train a reinforcement learning agent on your environment using **Proximal Policy Optimization (PPO) and Discretized-SAC** with **CleanRL**. You can also write your own PPO and SAC (I will be very impressed if you do so!).
- (Optional) Use wandb to track the reward of every episode, as well as the rendering of the training (use gym record wrapper).
- Please ensure your code runs end-to-end (including the training loop), and provide both the source code and a short README explaining how to run it.
- Some references and tips:
    - CleanRL: https://github.com/vwxyzjn/cleanrl/tree/master
    - Gymnasium: https://gymnasium.farama.org/index.html
    - Install environment: Gymnasium is part of CleanRL, so you only need to install CleanRL. In their documentation, they provide detailed instructions on how to install it on your local machine. You can also choose to use the Colab version, which is also provided in the CleanRL documentation. **Setting up environments can be tricky. If you are stuck setting up the environment, please let me know as soon as possible, so I can help you move forward.**
    - Computational resources: Since RL requires some amount of computational power on your machine, you might not be able to run it fast enough. If that is the case, I would recommend you either run it on Colab or set the **total_timesteps** argument in PPO.py to 10000 for testing your code. If your code can run successfully without issue, you can then send it to me and I will run the actual training for you.
- For anything I did not specify, you can design it yourself.

Submission:

- Please send me your code (either as a GitHub repo link, a zip file, or a Colab link) as soon as possible, **preferably within 3-5 days**.
- In your README, include Instructions to run the training script.