ML 题目：中文

在 Python 中实现以下规范。提交包含您实现的 python 源文件和 pdf 报告。报告必须记录结果，如“实验”部分所述。

环境

冬天来了。你和你的朋友在公园里扔飞盘，当你做了一个疯狂的投掷，飞盘落在湖中间。水面大部分都是冷冻的，但是有几个洞融化了冰。如果你踏入其中一个洞，你将陷入冰冷的水中。这个时候，国际飞盘短缺，所以你必须穿越湖泊并取回飞盘。

我们将使用 OpenAI Gym（https：// gym.openai.com/envs/FrozenLake-v0/）的 Frozen Lake 环境的修改版本来模拟这种情况。agent 控制角色在网格世界中的移动。网格的一些地面是可行走的，用“F”来表示冷冻，其他地方会导致 agent 落入水中，用“H”来表示孔。agent 因找到目标图块的可行走路径而获得奖励，目标为“G”。起点用“S”表示。

模板（注：最终状态为获得Goal的状态）

您将获得一个代码结构模板，您将使用该模板来实现此工作。 “main”函数初始化World和Agent。主要功能包括运行训练集的主循环。此外，还提供了绘图功能，以帮助您获得可视化结果。您将看到我们希望您展示三个图。第一个图包含由“world.q_Sinit_progress”记录的training episodes中初始状态下可用动作的q值。第二个图包含每个成功episode相对于training episodes(训练集)的步数的演变。一个成功的episode是使得agent获得目标的episode，如“episodeStepsCnt_progress”所记录（即，不仅仅是任何最终状态）。第三个图包含由“r_total_progress”记录的所返回的每个episode的演变。使用running_mean函数使最后的绘图更平滑。它是为您能够实现代码来更新这些值。

您可以在World类中编写用于运行训练集training episodes的代码，然后在Agent类中编写用于处理action-value函数和策略的代码。给定的代码已经具有相应的输入参数的功能。一些函数还有一些初始代码，可以帮助您开始编写自己的代码。

以下是 OpenAI GYM 的一些功能，可以帮助您完成此过程：

- env.observation_space.n: 返回环境中的状态数。
- env.action_space.n: 返回环境中的操作数。

• env.reset: 将 agent 设置为初始状态，并返回初始状态的 index(为 0).

• env.render: 打印当前环境状态，以帮助您可视化环境和 agent 的位置。

• env.step(a): 执行动作“a”并将 agent 移动到下一个状态。它返回下一个状态，奖励，一个指示是否到达最终状态的布尔值，以及您可以忽略的一些信息。
您可以自由探索OpenAI GYM提供的不同代码结构和功能。但是，我们提供的结构足以实现分配。

实现

您的任务是实现一个 domain-independent（域独立/域无关，不知道怎么译）的强化学习 Agent。您所实现的内容必须具有以下功能：

• predict_value(s): 返回一个向量，包含状态 s 中每个操作的值。

• update_value_Q(s, a, r, s_next): 使用 Q-learning 更新 state-action(以<s，a>表示)当前的估计值。

• update_value_S(s, a, r, s_next): 使用 Sarsa 更新 state-action(以<s，a>表示)当前的估计值。

• choose_action(s): 返回要在状态 s 中执行的操作，实现 ε-贪婪策略。

• run_episode_qlearning():运行一个 episode，以 Q-learning 算法进行学习

• run_episode_sarsa():运行一个 episode，以 Sarsa 算法进行学习

• run_evaluation_episode():运行一个执行当前最优策略的 episode。

注意：在实施 Q-learning 和 Sarsa 算法时，在更新值是关于最终状态（找到了 Goal）时必须格外小心。对于任何非最终状态，更新规则包括下一个状态中的下一个操作的值，
即对 Sarsa 而言：。。。。（第一个公式）

对 Q-learning 而言：。。。（第二个公式）

其中 α 是学习率，r 是直接奖励，γ 是折扣因子（代码中的 Gamma），s'是下一个状态，a'是下一个动作。如果状态是最终状态，则没有下一个状态 s'。在这种情况下，两种方法的更新都是：。。。（第三个公式）

PS:代码中的 `EPSILON` 应该是 `greedy police`

实验

对于以下每个实验，生成“代码部分”中所说明的三个图，并将它们添加到PDF报告中。

1，On-vs Off-policy

在这些实验中，您将研究 on-policy 和 off-policy learning（on 即 Sarsa, off 即 Q-learning，在莫烦 Python 看到的）之间的区别。运行以下四个实验，并将相应的图添加到报告中：

- RL Algorithm: Q-learning; ε = 0.8.
- RL Algorithm: Sarsa; ε = 0.8.
- RL Algorithm: Q-learning; ε = 0.1.
- RL Algorithm: Sarsa; ε = 0.1,

其中 ε 是用于探测（不知道这里探测啥意思）的 ε-贪婪策略（greedy policy）的参数。哪种算法学习 on-policy，哪一种学习 off-policy？on-policy 和 off-policy 的行为是如何影响已学习的值的功能？它是否也会影响收敛到最优策略的能力？

2，折扣因素

在这些实验中，您将研究折扣因子 γ 的影响。使用 Q-learning 算法执行以下实验，并将相应的图添加到报告中：

- $\gamma = 1$.
- $\gamma = 0.9$.
- $\gamma = 0$.

折扣因子的每个值对学习策略和值函数的意义（影响）是什么？

3，环境随机性

在这些实验中，您将研究环境随机性对 agent 的影响。使用 Q 学习算法执行以下实验，并将相应的图添加到报告中：

- Deterministic environment.---确定性环境。
- Stochastic environment.---随机环境。

从 ε= 0.5 的初始值开始，随着 episodes 的数量线性地减小 ε 的值，以便在最后 10％的 episodes 使 ε= 0.05。确定最高学习率（+- 0.05）使得 agent 能够学习最优策略。随机环境是如何影响已学习的值函数？这如何反映在学习率上？