

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 人工智能 |
| 姓 名： | 王俊 |
| 学 院： | 海洋学院 |
| 专 业： | 海洋工程与技术 |
| 学 号： | 3170100186 |
| 指导教师： | 吴飞 |

2021年 6 月 23 日

**Lab7—机器人自动走迷宫**

课程名称： 人工智能 实验类型： 综合

实验项目名称： 图像恢复

学生姓名： 王俊 专业： 海洋工程与技术 学号： 3170100186

同组学生姓名： None 指导老师： 吴飞

实验地点： 曹西503 实验日期： 2021 年 6 月 23 日

1. 问题重述

* 背景：

强化学习作为机器学习算法的一种，其模式也是让智能体在“训练”中学到“经验”，以实现给定的任务。但不同于监督学习与非监督学习，在强化学习的框架中，我们更侧重通过智能体与环境的交互来学习。通常在监督学习和非监督学习任务中，智能体往往需要通过给定的训练集，辅之以既定的训练目标（如最小化损失函数），通过给定的学习算法来实现这一目标。然而在强化学习中，智能体则是通过其与环境交互得到的奖励进行学习。这个环境可以是虚拟的（如虚拟的迷宫），也可以是真实的（自动驾驶汽车在真实道路上收集数据）。

在强化学习中有五个核心组成部分，它们分别是：环境（Environment）、智能体（Agent）、状态（State）、动作（Action）和奖励（Reward）。在某一时间节点t：

-智能体在从环境中感知其所处的状态

-智能体根据某些准则选择动作

-环境根据智能体选择的动作，向智能体反馈奖励

通过合理的学习算法，智能体将在这样的问题设置下，成功学到一个在状态 选择动作 的策略 。

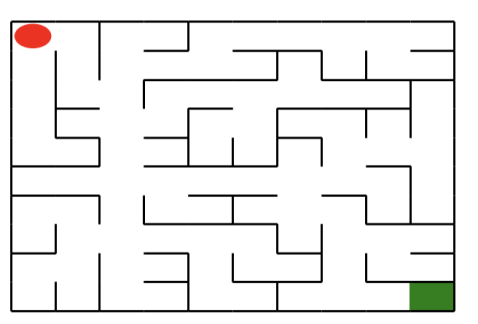
* 具体实验要求：

--在本实验中使用 Python 语言。

--使用基础搜索算法完成机器人走迷宫。

--使用 Deep QLearning 算法完成机器人走迷宫。

--算法部分需要自己实现，不能使用现成的包、工具或者接口。



1. 设计思想和代码内容

**2.1基础搜索算法介绍**

对于迷宫游戏，常见的三种的搜索算法有广度优先搜索、深度优先搜索和最佳优先搜索（A\*)。

* 广度优先搜索算法：

主要通过建立一颗搜索树并进行层次遍历实现。

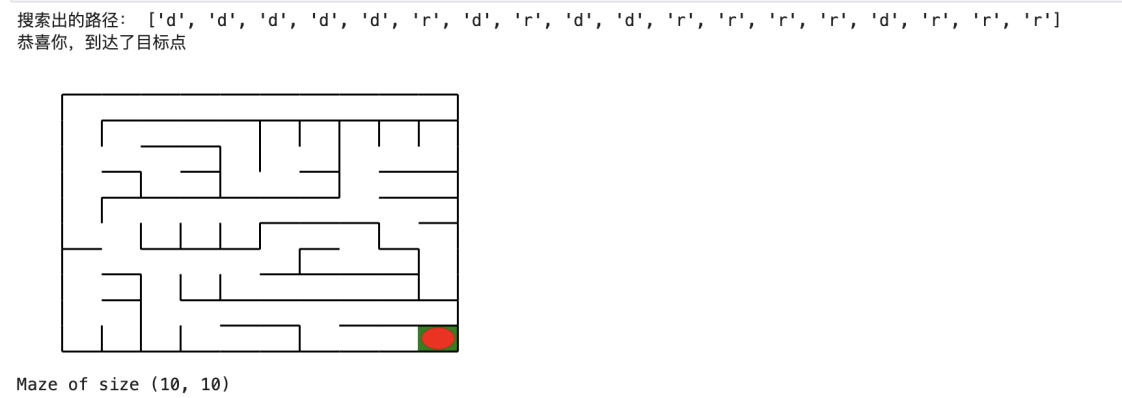
每个节点表示为以 Class SearchTree 实例化的对象，类属性有：**当前节点位置、到达当前节点的动作、当前节点的父节点、当前节点的子节点；**

--valid\_actions(): 用以获取机器人可以行走的位置(即不能穿墙);

--expand(): 对于未拓展的子节点进行拓展；

--backpropagation(): 回溯搜索路径。

**测试广度优先搜索算法:**



* 深度优先搜索算法：

1.设置节点栈，用于深度遍历:

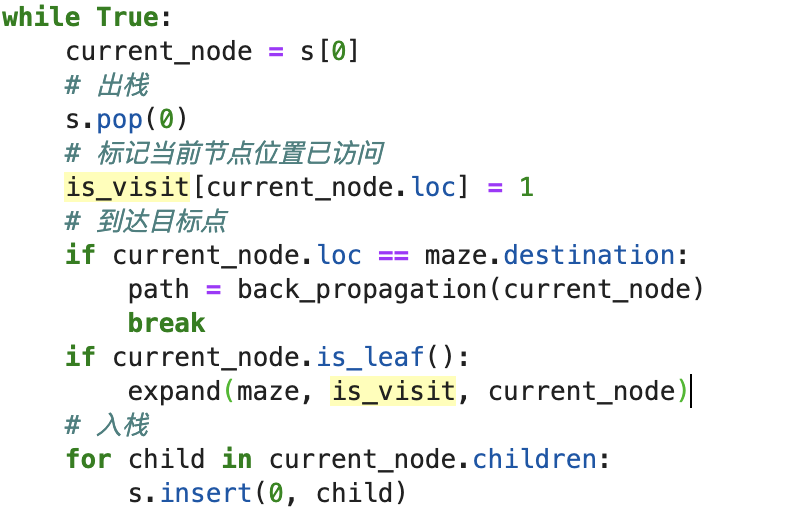


2.标记迷宫的各个位置是否被访问过:



1. 循环进行dfs：

从root节点开始，不停地进行深度优先搜索，直到找到路径才break

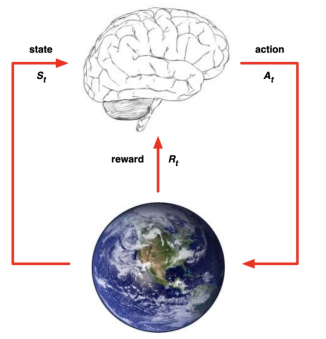


**测试深度优先搜索算法:**



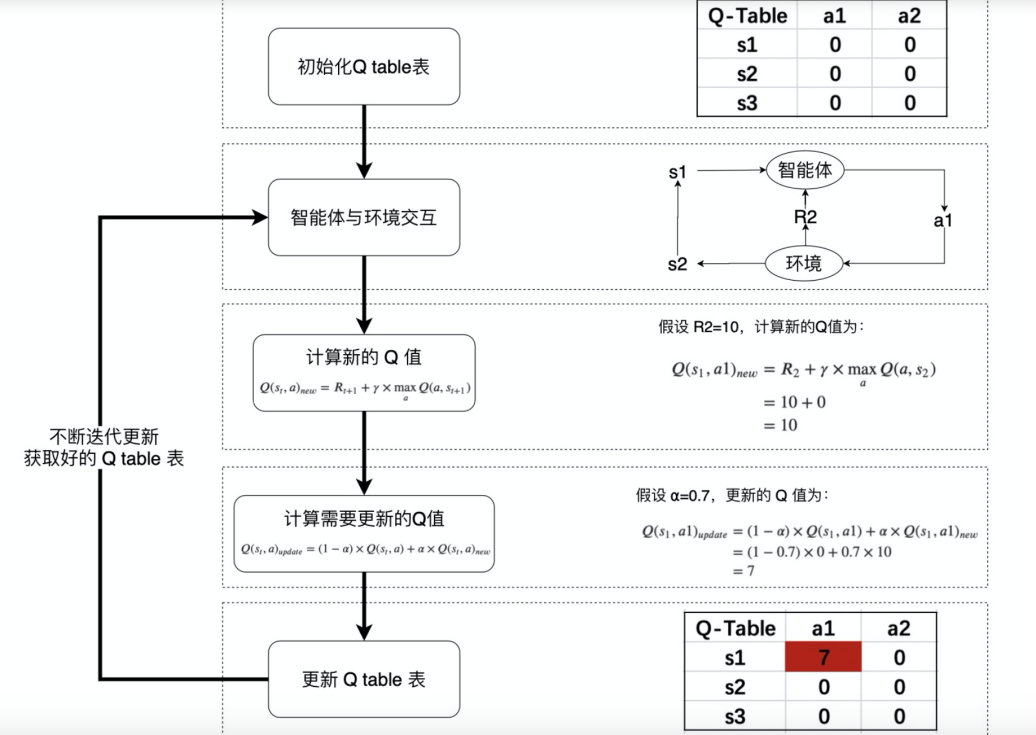
**2.2 强化学习算法**

通过合理的学习算法，智能体将在这样的问题设置下，成功学到一个在状态 st 选择动作 at 的策略 π(st)=at。

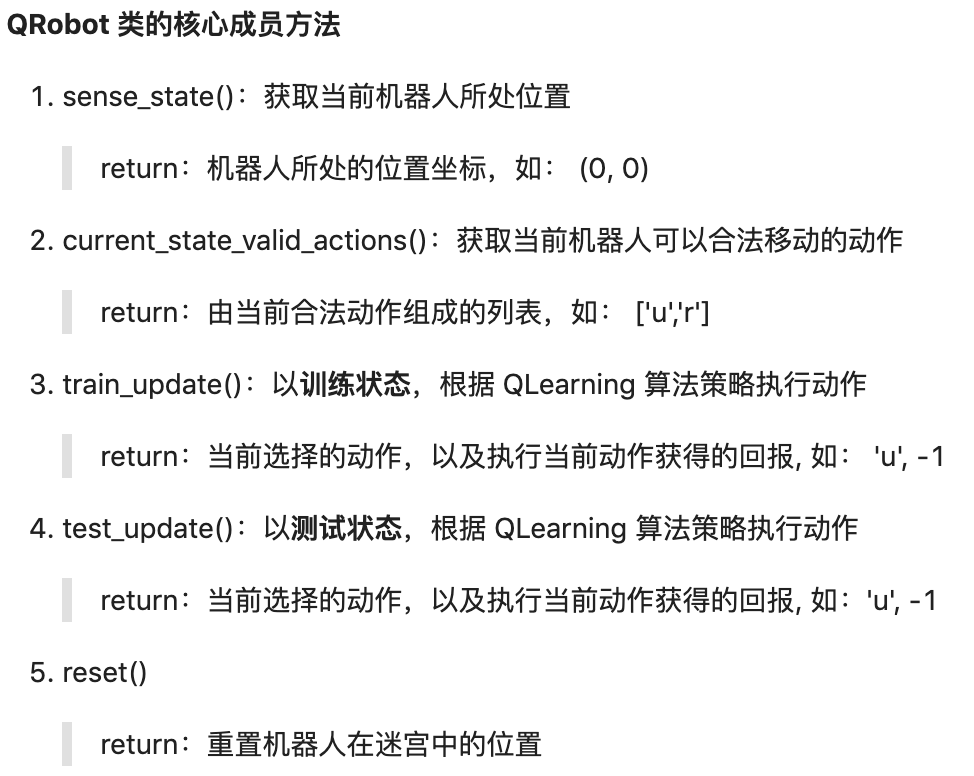


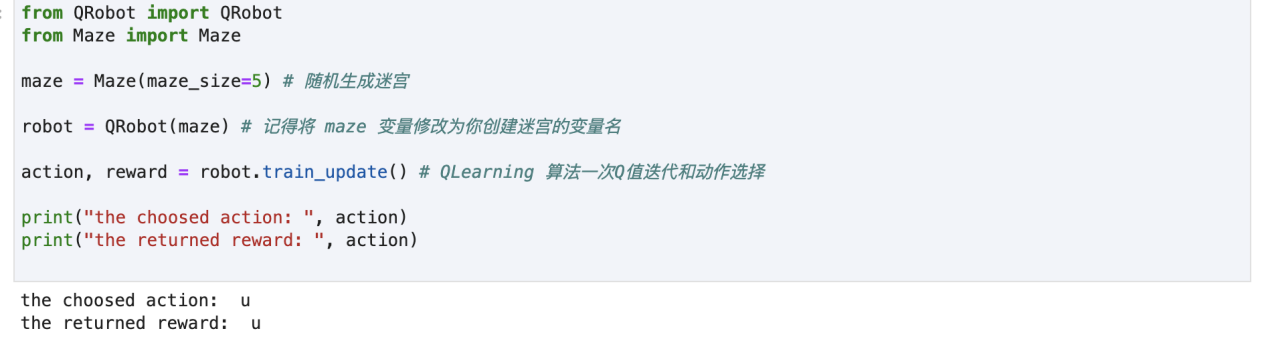
* QLearning 算法：

Q-Learning 是一个值迭代（Value Iteration）算法。  
与策略迭代（Policy Iteration）算法不同，值迭代算法会计算每个”状态“或是”状态-动作“的值（Value）或是效用（Utility），然后在执行动作的时候，会设法最大化这个值。  
因此，对每个状态值的准确估计，是值迭代算法的核心。  
通常会考虑**最大化动作的长期奖励**，即不仅考虑当前动作带来的奖励，还会考虑动作长远的奖励。

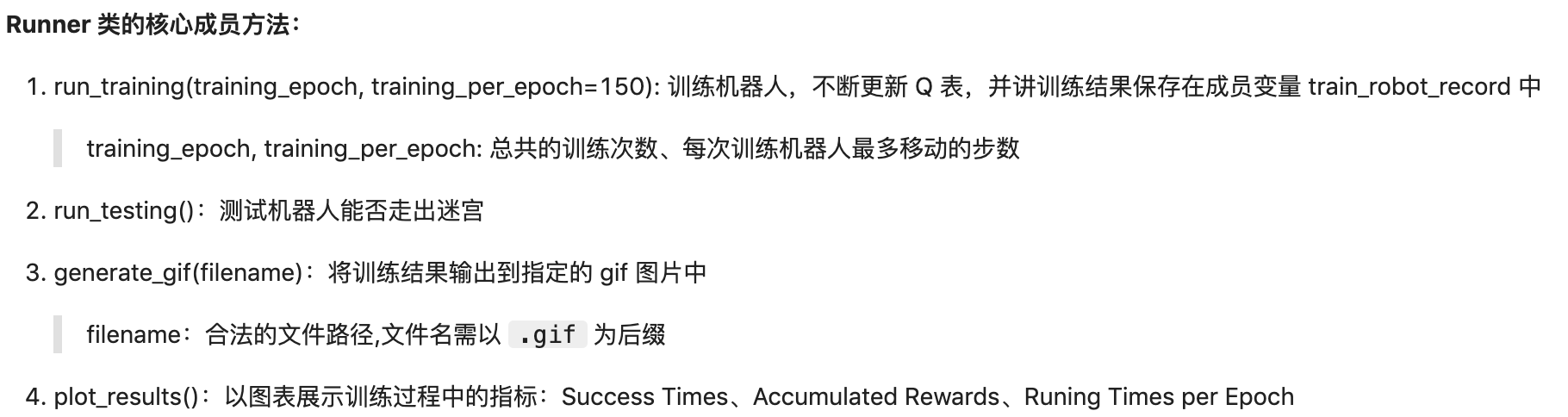


* **QRobot 类：**





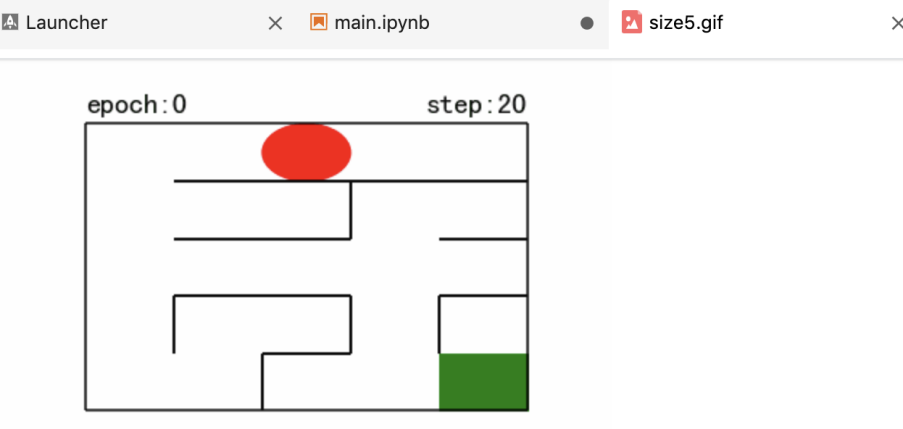
* **Runner 用于机器人的训练和可视化：**



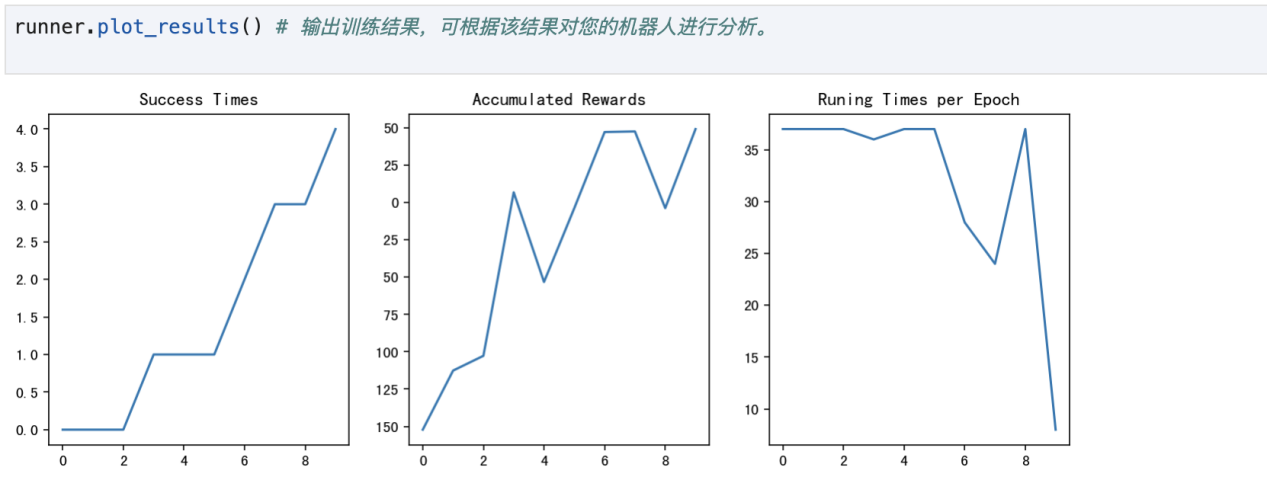
设定训练参数、训练、查看结果：



可以在gift图中查看结果：

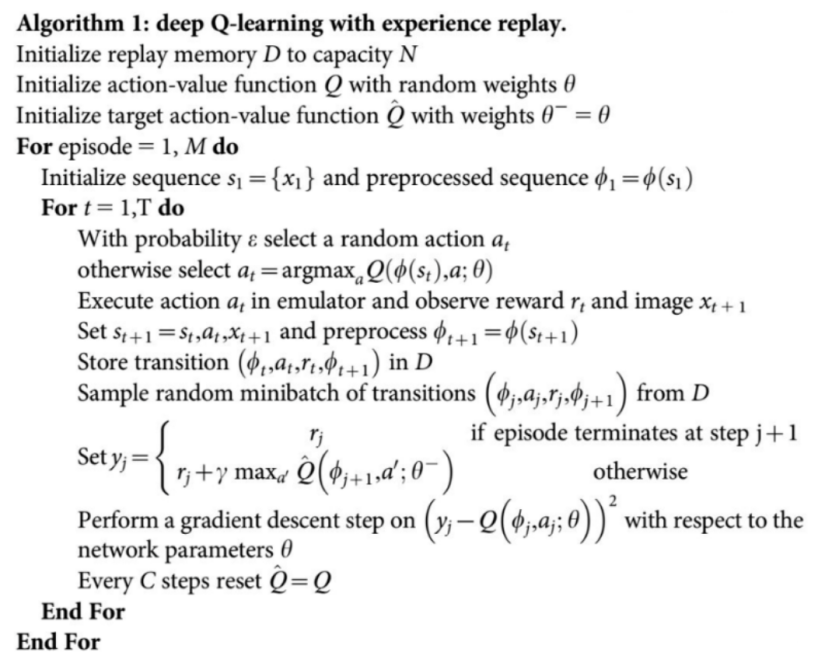


输出结果，便于分析：

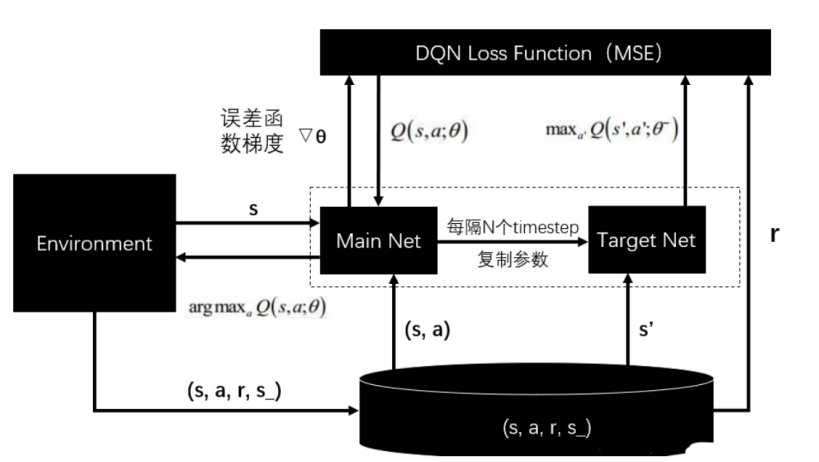


**2.3 实现 DQN 算法**

算法流程：



DQN 算法框架图：

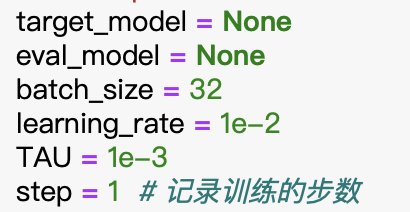


 完成 DQN 算法：

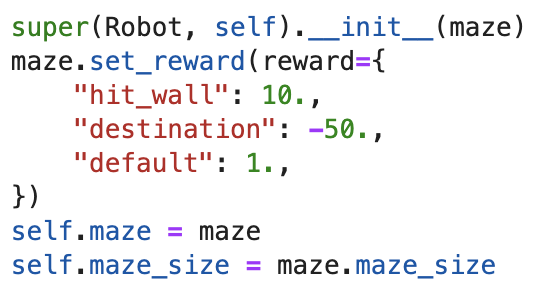
QLearning parameters：



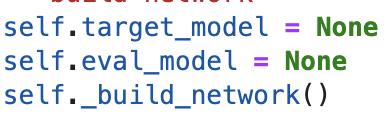
some parameters of neural network：



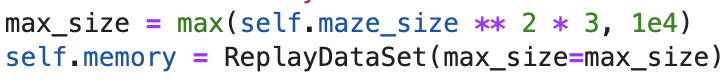
//初始化 Robot 类：



//build network：



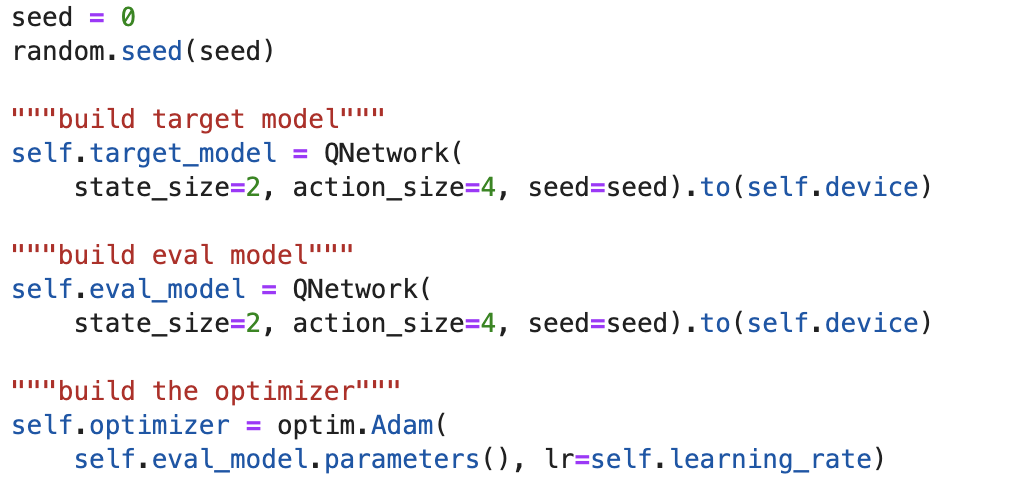
//create the memory to store data：



//开启金手指，获取全图视野：

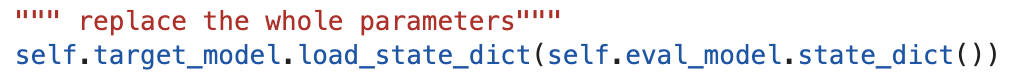


Build network：

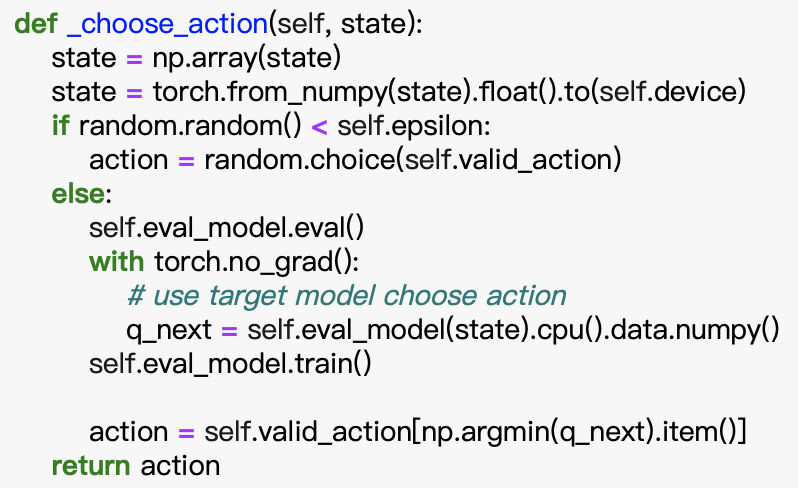


Soft update the target model parameters.

θ\_target = τ\*θ\_local + (1 - τ)\*θ\_target

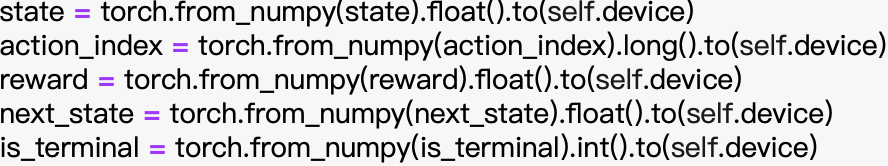


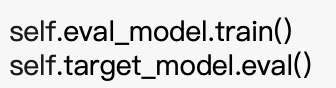
* choose\_action：



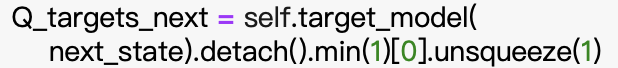
* Learn：

--convert the data to tensor type：

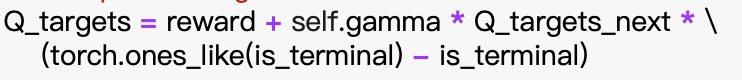




--Get max predicted Q values (for next states) from target model：



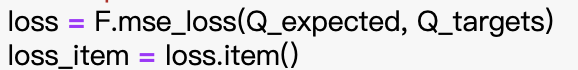
--Compute Q targets for current states：



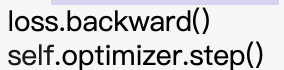
--Get expected Q values from local model：



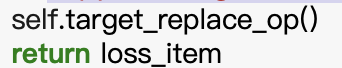
--compute loss：



--Minimize the loss：



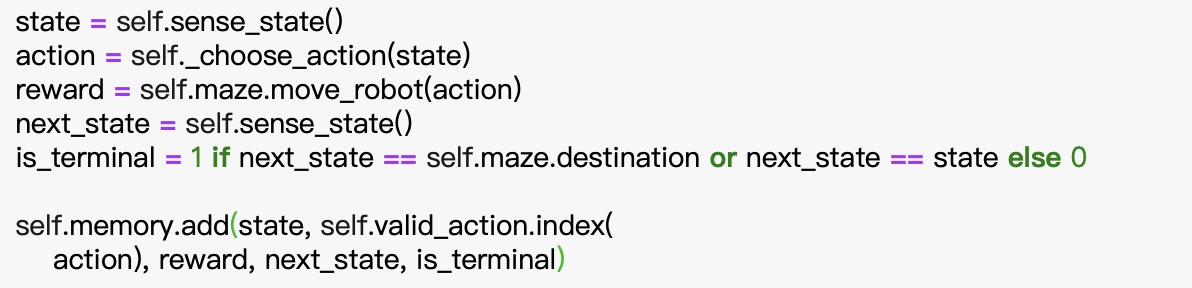
--copy the weights of eval\_model to the target\_model：



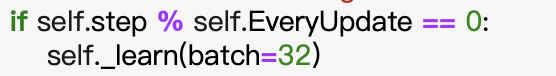
* **train\_update：**

训练状态选择动作并更新Deep Q network的相关参数

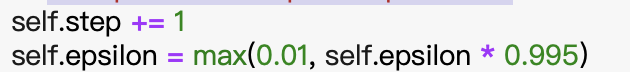
--获取状态信息：



--间隔一段时间更新target network权重：

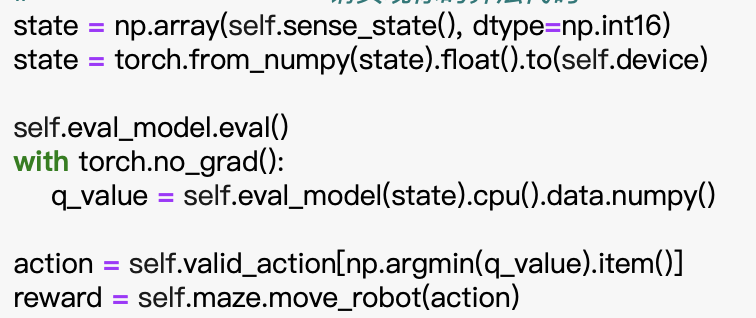


-----update the step and epsilon---：



* **test\_update：**

以测试状态选择动作并更新Deep Q network的相关参数：



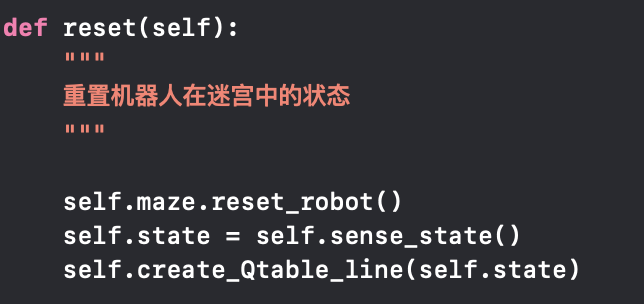
**2.4 实现 Q-Learning 算法**

**由于DQN算法可能对于走迷宫类游戏在有限训练步长内很难发挥其优势。所以进行选择提交Q-learning算法(QRobot）的选择**

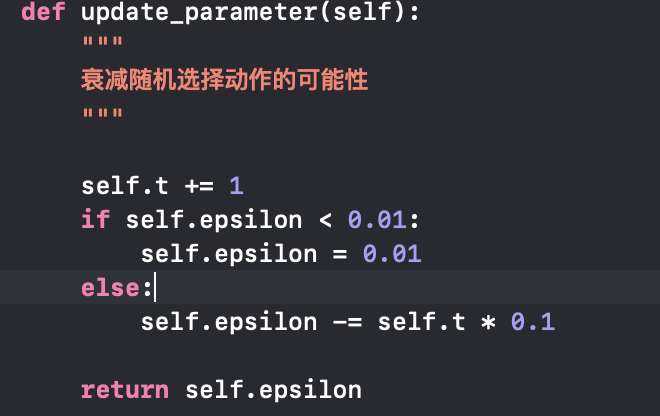
**初始化参数：**



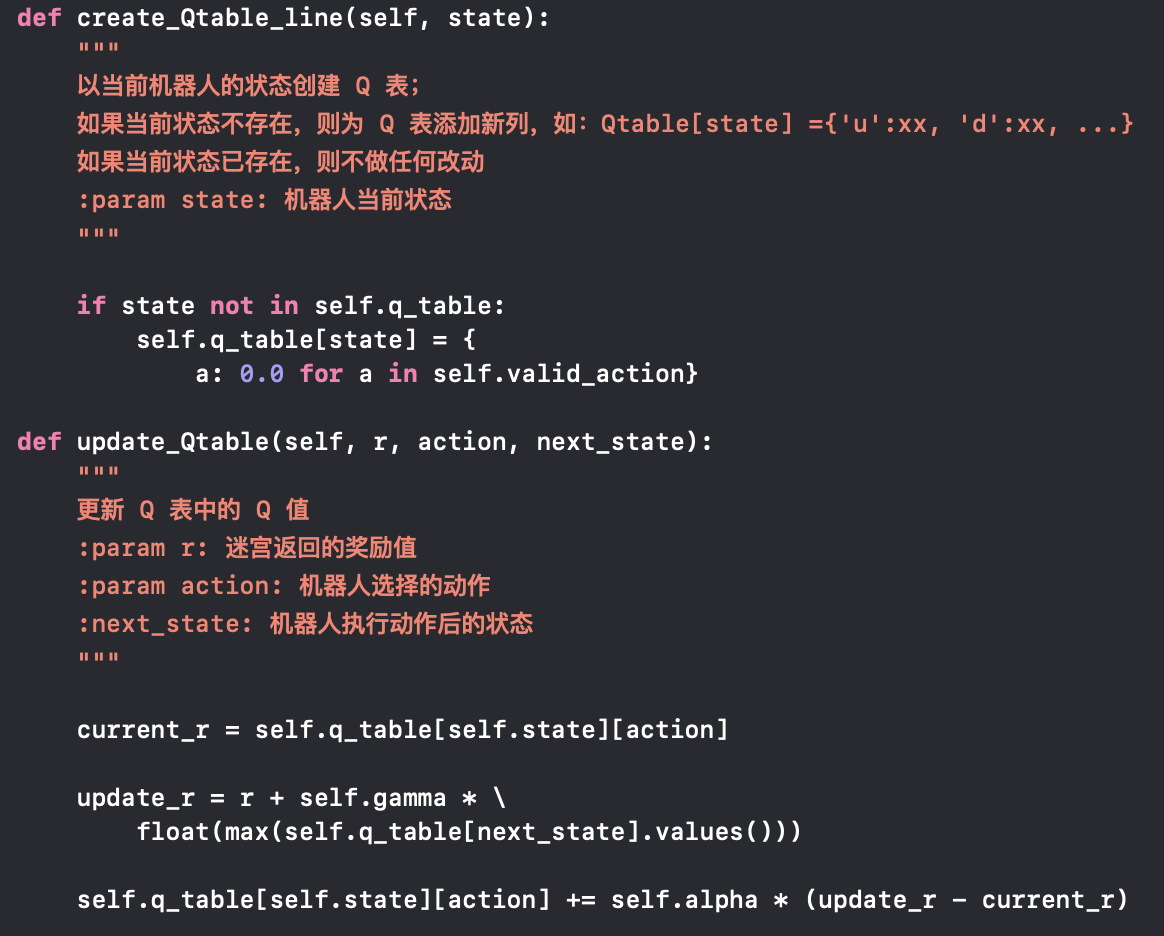
**重置状态：**



**衰减随机选择动作的可能性：**



**建立和更新Q表：**



训练动作：

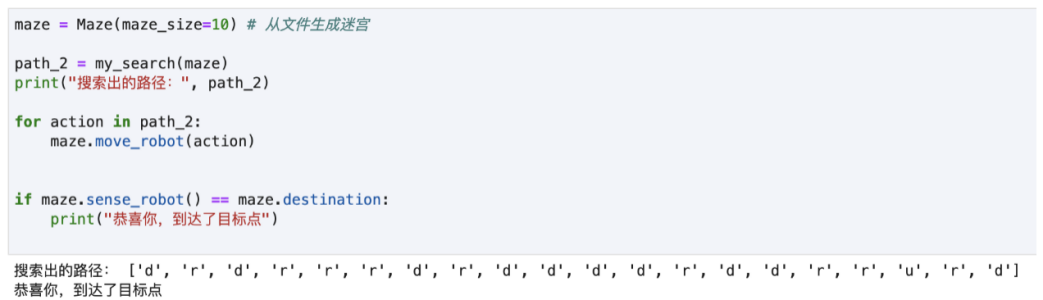


测试状态选择动作：



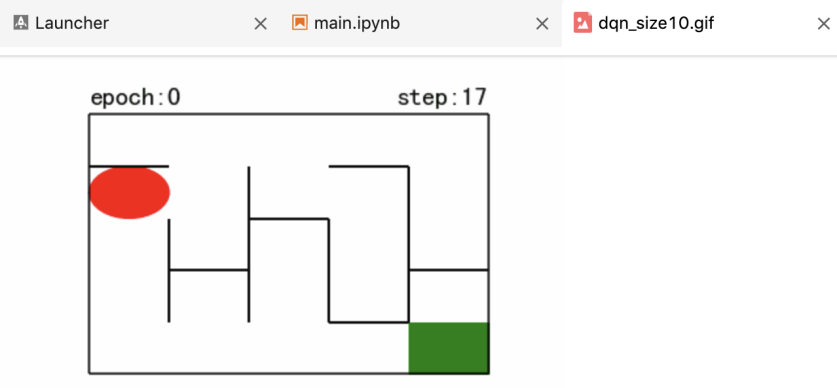
1. 实验结果

3.1. 测试我的深度优先搜索算法:

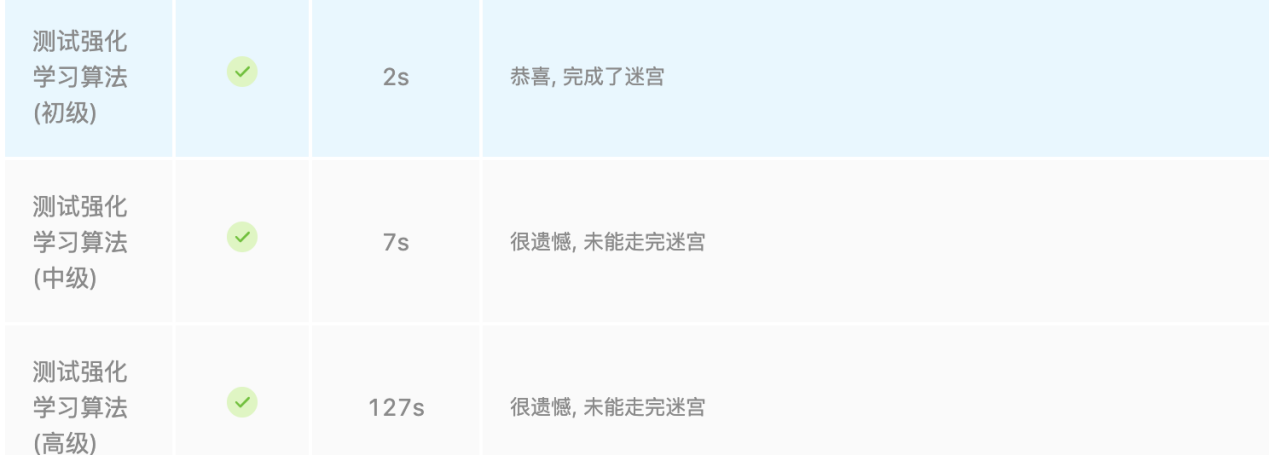


3.2. 测试我的 DQN 算法:





测试我的结果:



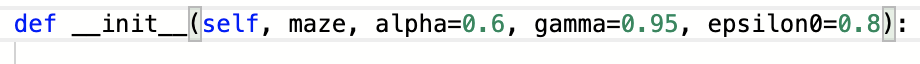
3.3. 测试我的 Q-learning 算法:

一开始：



更改了参数后，使其随机性更大，对未来的反馈更高：





**0.7的时候完全成功的案例还是比较少，当升到0.8的时候就比较多了**



1. 总结

4.1. 忘了将time import：

4.2. 没有将要用的searchtree放里面：



4.3. 调参时要理解参数的意义，不能乱调整，根据题意来选择参数，比如在这题中，明显需要加大机器人的随机性