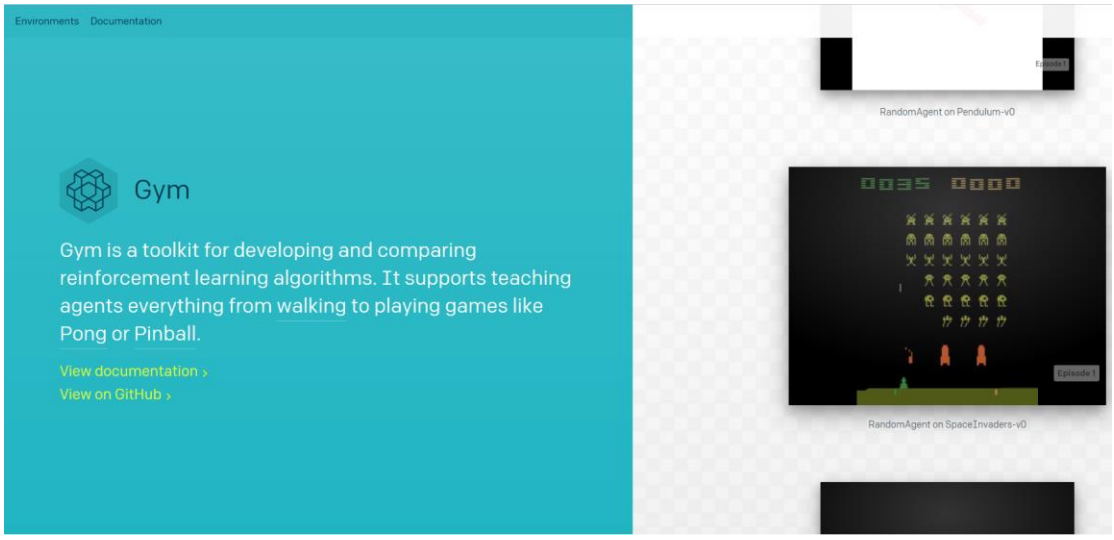
学习 OpenAI Gym

官方网址：gym.openai.com/



先学习文档 documentation，你要做以下工作：

1. 安装 gym 以及相关库比如 atari(需要什么装什么)
2. 运行'CartPole-v0'示例理解流程：

   载入相应游戏：　　env = gym.make('游戏名字')

   重置环境：　　env.reset()

   展示环境（渲染）：env.render()

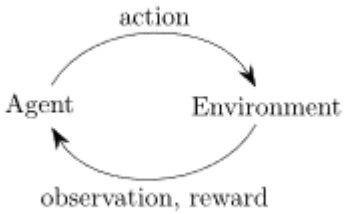
   环境中智能体执行动作：env.step()

   环境的关闭：　　env.close()

   标红函数为核心方法，其中 reset 代表重新开始任务，将环境设置成任务初始状态 s0；step(a)中参数 a 代表智能体此刻需要执行的动作，环境会依照行动模拟生成一系列的信息——下一时刻的观测状态、即时获得的奖励、此次任务是否结束以及其他信息(因此我们一般会这样写：

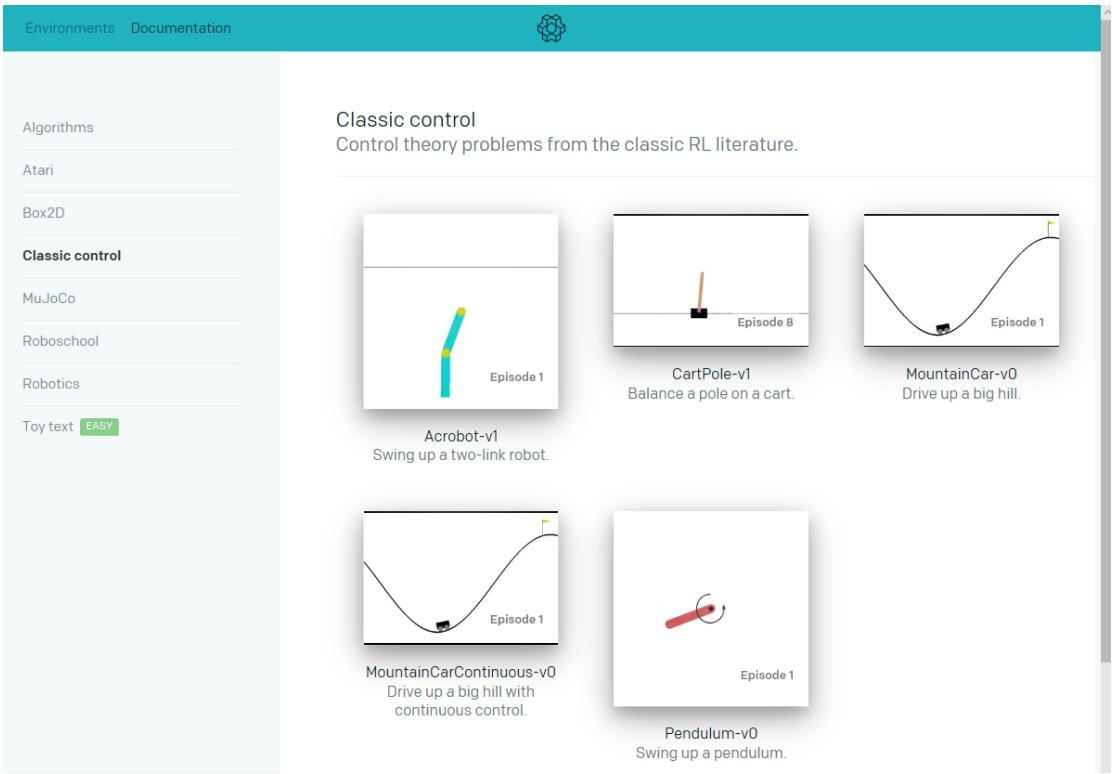
ob,reard,done,info = env.step(action…))；render()函数代表输出当前的状态，你在 OpenAI Gym 中看到的图片（一帧一帧的游戏）都是靠这个函数生成的。

3. 理解强化学习的基本流程：

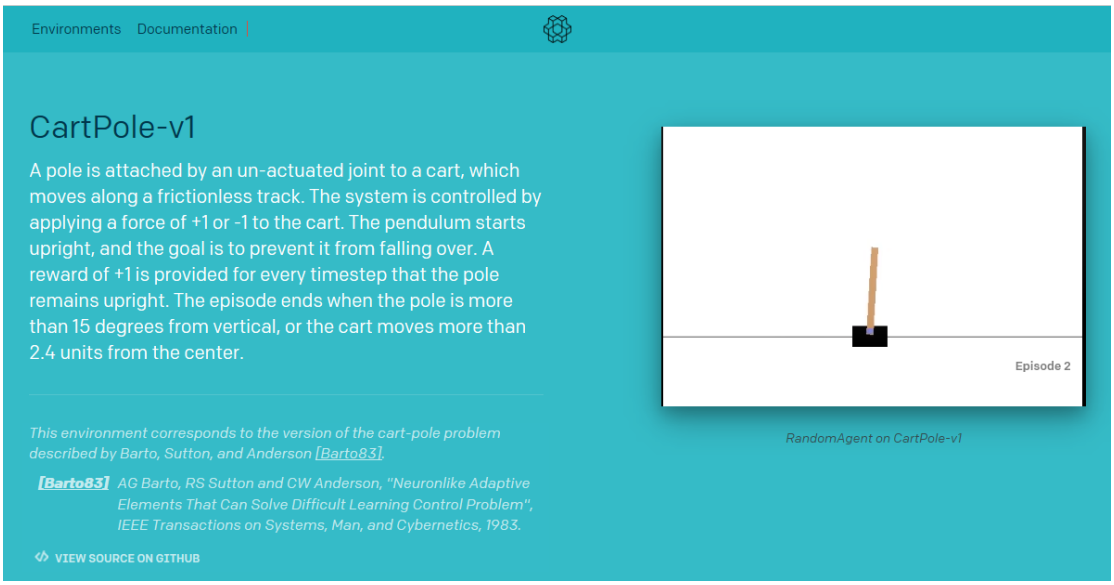


4. 深入理解 Observation 以及 Spaces 的概念(联系 github):

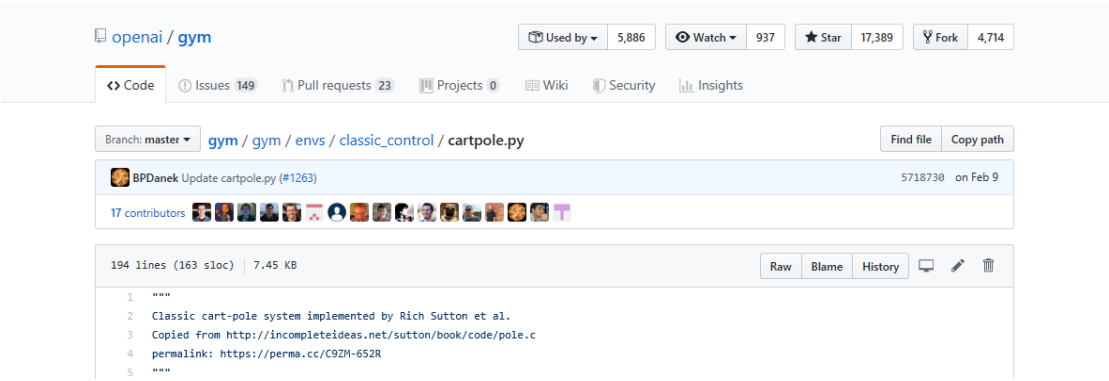
举个例子：



在左上角选中环境后你可以看到不同类型的游戏（有的库想要 run 需要安装相应依赖），点击一个环境，以‘CartPole-v1’为例：



你可以在左下角的接口中点击进入 Github 中查看其源码：



看代码理解 Observation 以及 Spaces：

```
Observation:
    Type: Box(4)
    Num     Observation               Min         Max
    0       Cart Position             -4.8            4.8
    1       Cart Velocity             -Inf            Inf
    2       Pole Angle                 -24 deg        24 deg
    3       Pole Velocity At Tip      -Inf            Inf

        Actions:
        Type: Discrete(2)
        Num     Action
        0       Push cart to the left
        1       Push cart to the right
```

明确其 Observation 不是图像的像素信息，而是小车位置、速

度、角度等信息；而 Actions Spaces 只有两个。

最后理解游戏的终止条件：

```
Episode Termination:
    Pole Angle is more than 12 degrees
    Cart Position is more than 2.4 (center of the cart reaches the edge of the display)
    Episode length is greater than 200
    Solved Requirements
    Considered solved when the average reward is greater than or equal to 195.0 over 100 consecutive trials.
"""
```

5. 在 env 中手写 RL 算法

代码样例：Policy gradient

介绍：

其与 Value-based 方法 (Q learning, Sarsa 等等)有很多不同，但它也要接受环境信息 (observation)，不同的是它要输出不是 action 的 价值（value），而是具体的哪一个 action（选择 action 的概率），这样 policy gradient 就跳过了 value 这个阶段. 而且个人认为 Policy gradient 最大的一个优势是: 输出的这个 action 可以是一个连续的值，之前我们说到的 value-based 方法输出的都是不连续的值，然后再选择值最大的 action. 而 policy gradient 可以在一个连续分布上选取 action.

run_CartPole.py 是 PG 算法更新（与环境相关）

RL_brain.py 是 PG 算法的思维决策（算法本身）

针对代码中理解不了 vt 的参考资料：

karpathy.github.io/2016/05/31/rl/

cs231n.github.io/neural-networks-2/#losses

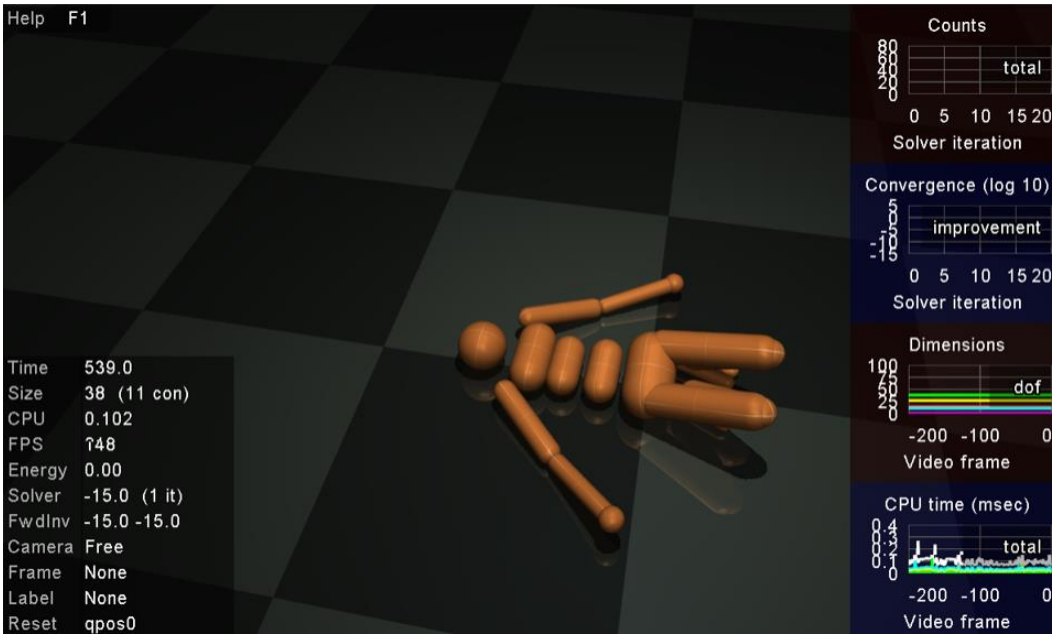
www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching_files/pg.pdf

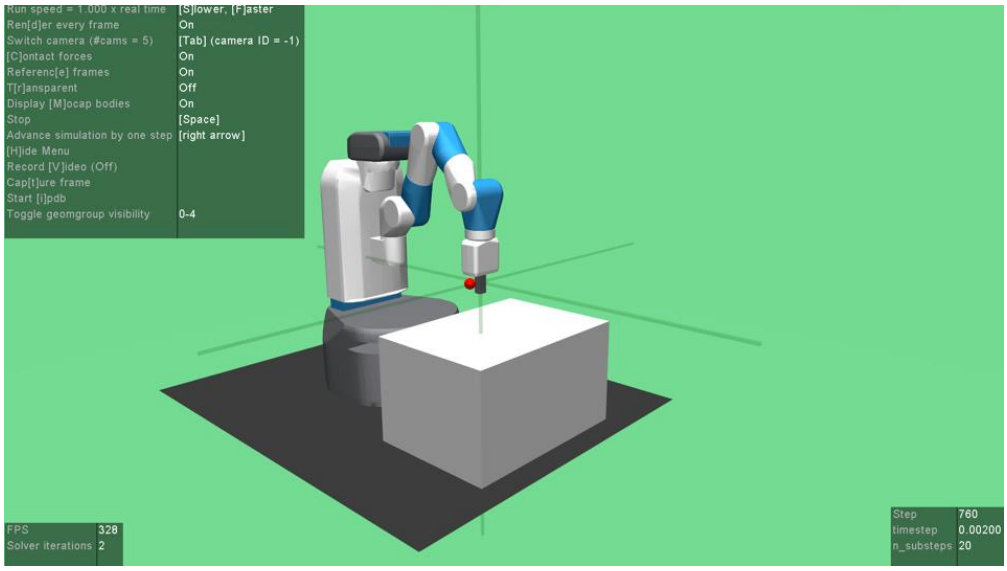
你还可以选择其他的游戏环境，不局限于 Gym

比如——**Mujoco（更加高大上）**

安装方法百度，尤其是 Windows 版本

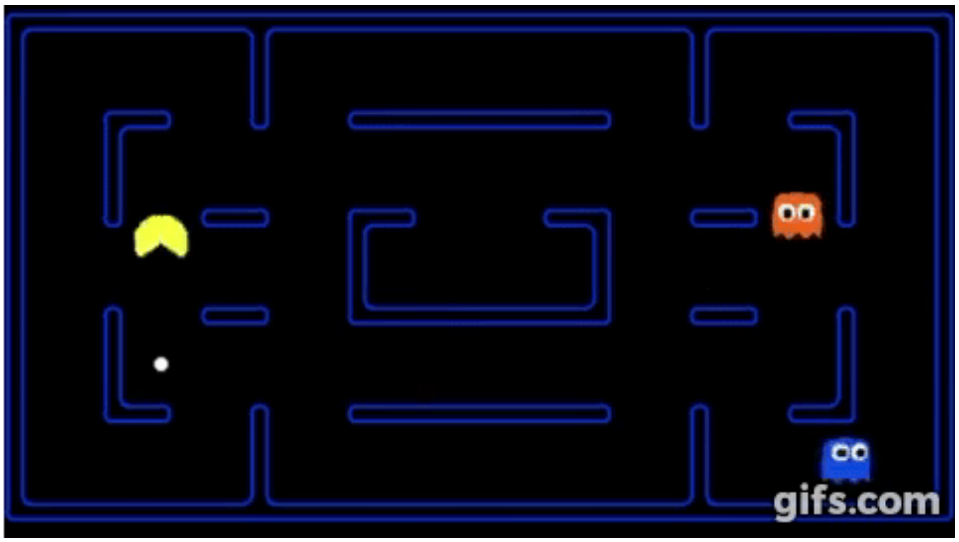
用学校邮箱可以申请使用权限，否则是有 30 天试用





比如：Github 上有很多别人开源的游戏环境

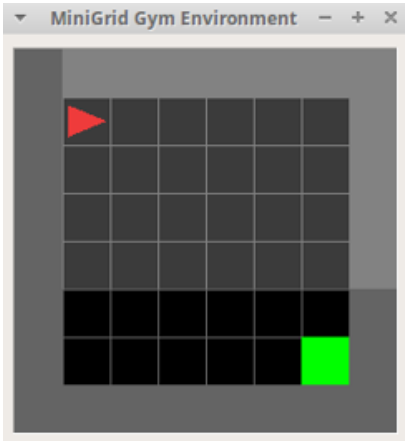
Pacman: https://github.com/tychovdo/PacmanDQN



Gomoku: https://github.com/junxiaosong/AlphaZero_Gomoku

```
AI move: 5,4

Player 1 with X
Player 2 with O

      0       1       2       3       4       5       6       7

   7  _       _       _       _       _       _       _       _

   6  _       _       _       _       _       _       _       _

   5  _       _       _       _       X       _       _       _

   4  _       _       X       _       O       _       _       _

   3  _       _       X       _       O       _       _       _

   2  _       _       _       _       _       _       _       _

   1  _       _       _       _       _       _       _       _

   0  _       _       _       _       _       _       _       _
```

Gridworld:



等等……

你们要做的是：选择一个环境（Gym 或者是更高级的也好），实现一个上课讲过的强化学习算法

如果选择最简单的 cartpole，或者是差不多的 MountainCar，就需要实现除了样例中 PG 算法中的其他强化学习算法（因为这样太简单，改改输入参数就可以了）

（单智能体）强化学习算法难度等级如下：

1. 最简单级别：DQN/PG/Actor-Critic
2. 中等级别：DQN 的各种变体（double DQN、dueling DQN…）、A2C
3. 比较难的级别：A3C、DDPG

给分：以环境、实现算法的难度等级、实现算法的个数给基础分和额外分（当然还有报告）

举例：如果用一个最简单级别的算法实现 **Mujoco**，只要程序可以 run 动，大作业部分直接满分、如果用 cartpole，实现了 DDPG 或者是 A3C 也可以获得满分

提交：一段视频（你们小组 run 项目的视频）

一个工程代码（压缩包。可能有各种算法，好好命名）

一份报告（写清楚组号、成员学号、姓名）

报告中要有：

1. 介绍你的游戏环境，如果用的比较难的，可以贴出参考博客的网址（安装细节什么的）
2. 介绍你的 RL 算法，比如算法中的数学公式对应的代码，或者是代码中特别的地方（算法的亮点）、或者是你添加了什么可以加快收敛，提高效率的方法，请介绍
3. 如果实现了多种算法，请比较（收敛速度、计算效率等等）
4. 注意，代码可以参考 blog 或者是 github 中，但要有一些自己的注释，证明小组确实阅读过并理解，而不是直接 copy，同时在报告中贴出参考的网址