1. myTeam.py 是我们要实现的部分，里面最重要的是createTeam这个方法。这个方法有两个参数：first，second分别就是代表两个pacman，我们分别为他们设计策略（比如一个吃豆一个防守）图1

A close up of a sign

Description automatically generated

图1

比如这个样例里给的两个agent是一样的，那两个豆子的策略就是一样的

2. baselineTeam.py 是老师给我们的测试的。我不知道他这个是不是我们最重要战胜的，但是这两个表现感觉挺好的。

他在这里面是用 ReflexCaptureAgent作为父类。这里面实现了诸如getsuccessor等基本操作。然后子类DefensiveReflexAgent和OffensiveReflexAgent分别实现了防守和吃豆的两个agent。这两个agent没用重写很多父类的方法，只是把getWeight和getFeature给重写了。然后把这两个函数作为createTeam的参数进去（如图2，图3）

A picture containing object

Description automatically generated

图2

A screenshot of a cell phone

Description automatically generated

图3

3.最后返回部分是chooseAction方法，这个方法是传禁区当前的gameState，最终要返回我们当前agent下一步要走的路径的，只需要返回一步即可。所以说应该是每个回合都调用在createTeam传进去的agent，返回一个下一步的结果

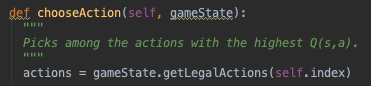


图4

4.下面是一些我用来表示状态的想法和理由

State：gameState，CaptureAgent 理由：gameState中是当前的局面，并不包括我们是哪一方的信息，所以必须要从CaptureAgent.red这个参数判断我们是哪一方并相应的获取到对方和我方的信息。获取信息的部分都是在gameState里的函数。但是这次跟上次不一样的是，你要获取哪一方的需要说明，比如说获取food是getRedFood方法

A close up of text on a black background

Description automatically generated

A screen shot of a social media post

Description automatically generated

图5

GoalState：gameState，CaptureAgent 理由：同上，要判断foodnum是不是为0，这一部分我已经写了，如图6

A screenshot of a cell phone

Description automatically generated

图6

Action：用CaptureAgent.getLegalActions 方法可以获得

其他的一些函数

getSuccessor： 在baselineTeam中已经有了，不过他是要有一个action后才能生成successor。如果我们想要实现DFS这种搜索算法的话就需要再写一个不依靠与action的getSuccessors。这个应该实现不难。（然而写起来发现有点麻烦）

evaluate： 一般是用来衡量这一步的价值的。之前我们写搜索算法，所以不会涉及到这个，不过之后要想提升效率的话难免要用到这个函数。

然后就没有然后了。基本上需要用到的暂时就是这一些，希望能帮助到大家。有什么错误的地方或者补充的地方大家可以直接继续往下写。