类：

Tractor：游戏的主类，维护游戏进程，让游戏执行起来的方法都在这里面

Player：游戏玩家的类，代表主类中包含的四个玩家，

Action：动作类，玩家使用的牌都会以Action形式给出而非用列表，这是为了方便处理。

### 让游戏执行起来的必须方法：

reset\_game()：

重置所有游戏参数：级牌是2，每个玩家等级是2，卡组和每个玩家手牌，底牌为初始状态。

返回值：无

calcGameScore()：

结算一局游戏分数，重置部分游戏参数：卡组和每个玩家手牌，底牌为初始状态。但每个玩家等级不变

返回值：winPlayer, dealer,grade

winPlayer代表赢的玩家编号，dealer代表下一局的庄家，grade代表赢者升的等级

dealCards(beginDeckList,bidFun,(setDecor=-1,setNum=-1,setDealer=-1)):

发牌函数，必须发牌后才能执行游戏

参数：beginDeckList代表牌堆列表，如果不传入就按照初始状态洗牌，否则就用传入牌并且不洗牌。bidFun代表叫牌函数，需要使用者自己去实现，如果发完牌没人叫牌，则重新发牌。setDecor, setNum , setDealer代表设置主牌颜色,数字，庄家编号，-1代表不设置。

返回值：无

bidFun(env,p,round,allActList)

代表叫牌函数的回调，需要使用者自己去实现

参数：env是游戏环境，p代表当前玩家的对象，round发牌时的轮数，allActList是动作空间，包括：不叫，以及每张级牌，如果已经有人叫，则需要对子来反主。对王可以反成无主。

返回值：act\_id代表动作在allActList中编号

step(ActionList, firstPlayerId,):

比较一个轮次的4组牌的大小

参数： ActionList是4个玩家得动作列表，firstPlayerId是先出得玩家编号,

返回值：playerId,sc,isTer,Info

其中，playerId是较大的人编号；sc是本轮牌中的分数；isTer为是否游戏结束；endSc是游戏结束时结算底牌的分数。只有isTer==True时，endSc才会为非0

firstPolicy (firstPlayerId,firstPolicyFun)

代表作为先手使用一个动作

参数：firstPlayerId是先出得玩家编号, firstPolicyFun是出牌策略，需要使用者自己去定义

返回值：一个action，代表动作

firstPolicyFun(env,p,usedAct,allActList)

一个出牌策略回调，需要使用者自己去定义

参数：env是游戏环境的对象，p是先出得玩家对象, usedAct是当前已经决定的牌的动作，allActList是动作空间的列表。具体来说，当第一次执行firstPolicyFun时，allActList会调用getFirstAllAction，allActList中包含所有非甩牌动作和甩牌的类别。如果玩家选择了非甩牌动作则firstPolicy执行结束；如果玩家选择了甩牌动作，则继续执行firstPolicyFun，每次得到的allActList的allActList[0]代表动作结束标识，其余时单张牌动作。

返回值：act\_Id代表选取的动作在allActList中的编号

otherPolicy(act4,firstPlayerID,nowPlayerId,otherPolicyFun):

代表作为非先手使用一个动作

参数：act4代表4个人对于玩家id的动作，没出的人动作是None，firstPlayerID代表先出的人id, nowPlayerId代表当前要出牌的玩家的id, otherPolicyFun是出牌策略，需要使用者自己去定义

返回值：一个action，代表动作

otherPolicyFun(env, p, usedAct, allActList)

一个出牌策略回调，需要使用者自己去定义

参数：env是游戏环境的对象，p是先出得玩家对象, usedAct是当前已经决定的牌的动作，allActList是动作空间的列表。具体来说，当第一次执行firstPolicyFun时，allActList会调用getFirstAllAction，allActList中包含所有非甩牌动作和甩牌的类别。如果玩家选择了非甩牌动作则firstPolicy执行结束；如果玩家选择了甩牌动作，则继续执行firstPolicyFun，每次得到的allActList的allActList[0]代表动作结束标识，其余时单张牌动作。

返回值：act\_Id代表选取的动作在allActList中的编号

useAction(act,updateINFFun):

使用一组牌

参数：act是玩家的动作(其中包含玩家编号)，updateINFFun是一个函数，用来确定该动作内部的INF用什么牌来代表，updateINFFun需要使用者自己来定义

updateINFFun(p,usedCards,allActList)

参数：p是当前玩家的对象，usedCards是这组牌内，已经用了的牌的列表，allActList是可能出的单张牌。

返回：act\_id代表动作在allActList中编号

### 其他可能用到的方法

一下函数以及被包含在必须方法中，不需要单独使用

getFirstAllAction(playerId)

返回当前状态下如果该玩家先出牌的动作空间

参数：playerId是玩家编号;

返回值：返回一个Action和list组成的列表，具体来说，所有单牌，对子，拖拉机，会返回明确的Action，然后包含最多3个由数字组成的list，长度为1，数字代表甩牌的类别。

getAllAction(p, firstAct)

返回当前状态下如果该玩家不先出牌的动作空间

参数：p是当前玩家; firstAct代表当前第一个人的动作

返回值：返回2个Action组成的列表upAns和downAns。upAns代表比firstAct大的明确动作列表；downAns代表比firstAct小的动作列表，如果firstAct是单张，downAns包含的动作都是明确值。如果downAns是非单张，则返回和firstAct相同数量的INF

judgeSeqUse(act)：

判断一个甩牌动作能不能甩,如果不能甩，就返回最小的牌

参数：act是这个动作，act中包含了所属玩家。

返回值：如果该牌是甩牌且不能甩，返回最小的子动作。否则返回act本身

printAllInfo()

打印环境和每个玩家当前状态的全部信息

Tractor\_action文件中的方法：

print()

主要是调试过程中打印动作

dfsPrintActList(li)

该方法在Action类外部，主要是调试过程中深度优先输出卡牌列表，且列表和Action都能打印

发现该环境有什么错误，可以联系我github或邮箱

<https://github.com/meiyoumingzile>

2428578442@qq.com

目前的先手玩家出牌方法中，如果玩家选择甩牌并且甩牌失败会直接扣分并且强制出最小的牌。环境中还缺少一个最保守的甩牌策略，告诉当前玩家一定可以甩的牌。