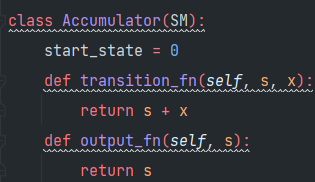
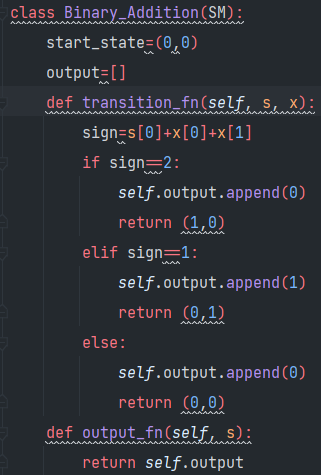
1. 依照材料要求声明父类SM，同时补充tranduce函数



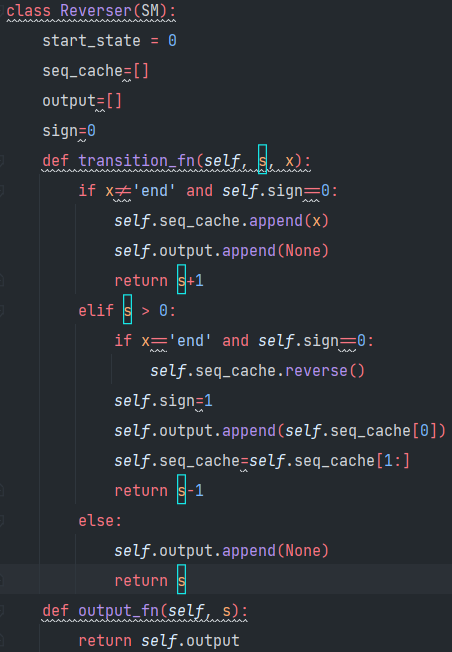
遍历输入列表中的元素，对于每个元素和上一次的状态调用transition函数，再将状态更新。遍历结束后调用output函数进行材料要求的输出



子类累加器，以当前求出的元素和为状态



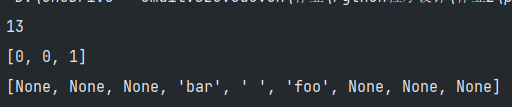
二进制加法，用元组形式存储状态，列表output存储结果。状态的第一个元素标志是否进位，第二个元素表示结果中当前位置的值，需要添加到output列表中



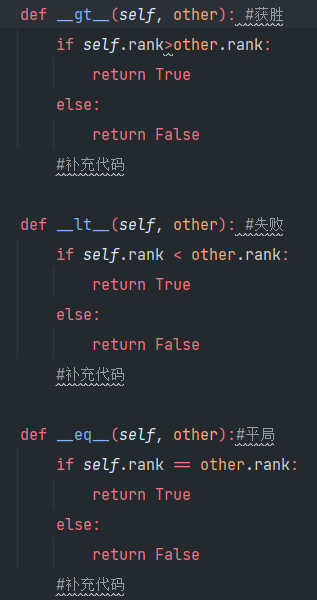
反相器，状态用于存储读入的第一个列表大小，利用一个列表seq\_cach存储第一个列表的数据，output存储输出结果，sign表示第一个列表读入是否结束。

还未读入‘end’之前，将读入的元素存入seq\_cach列表，同时状态加1，output中添加None；第一次读入‘end’时，将缓冲列表元素对调，sign改变标志第一个列表读入结束，在output中添加缓冲列表中的第一个元素，再删除缓冲列表中的第一个元素；缓冲列表为空之后在output中添加None

结果：

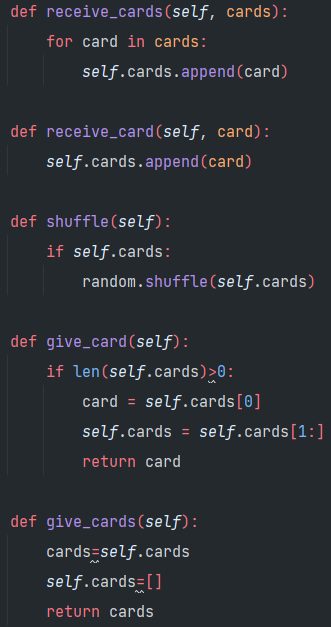


1. 在Card类中添加对胜负情况判断函数的代码：

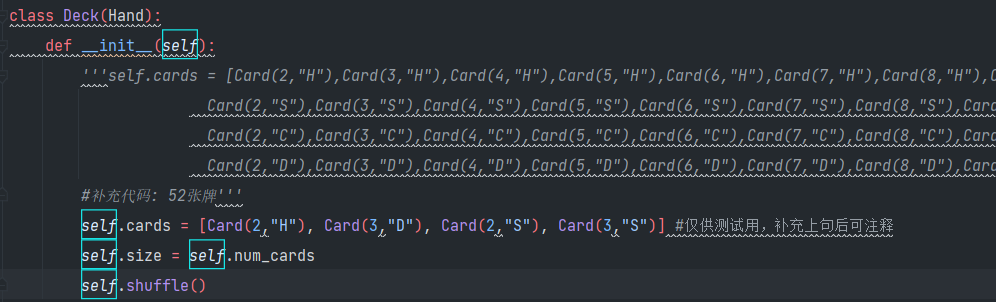


摸牌函数中，摸一张简单利用append‘函数；摸多张需要遍历列表中的每个元素，对每个元素用append函数。

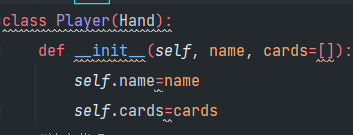
出牌函数中，出一张简单利用切片操作；出多张直接利用clear函数。



在Deck类中添加52张牌的初始化



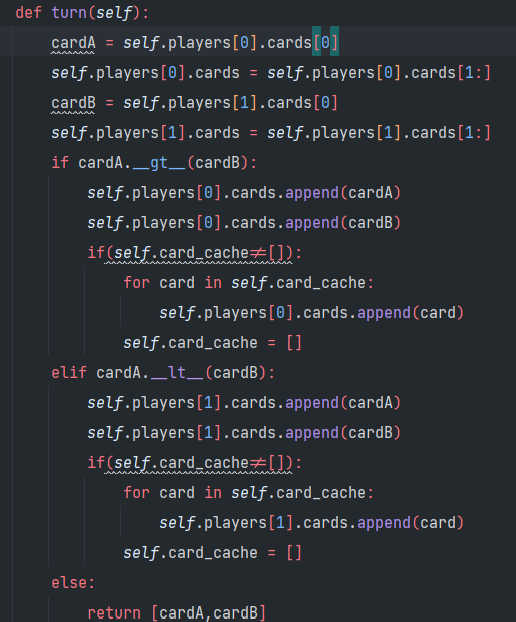
在Player类中补充初始化函数



发牌函数，以牌堆大小的一半为循环次数



回合函数，调用前文胜负判断函数并进行对应处理，平局时利用一个缓冲列表存储打出的两张牌

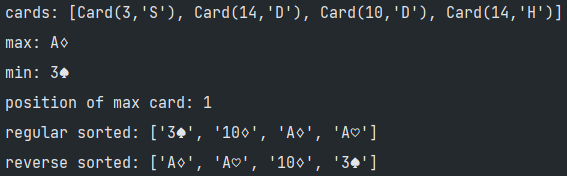


游戏进行函数，发牌后在一方手牌为0之前循环调用上述回合函数，如果非平局则对每个玩家都进行一次洗牌。最后判断赢家



结果：

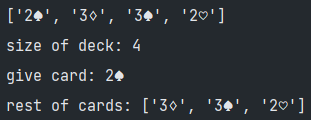
Test\_card



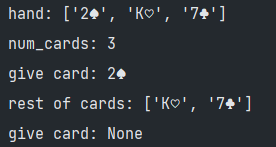
Test\_player



Test\_deck



Test\_hand

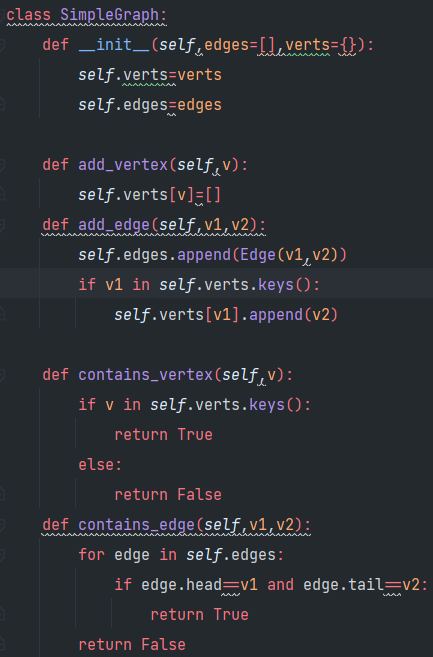


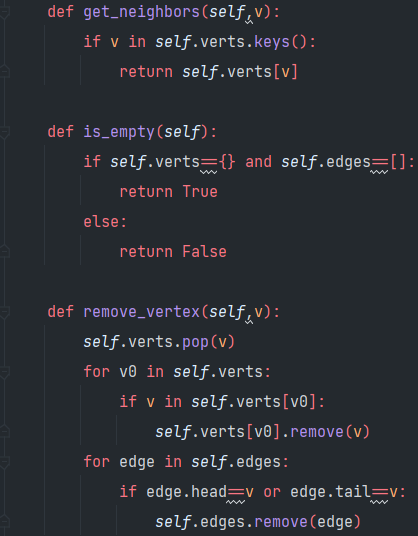
Test\_game

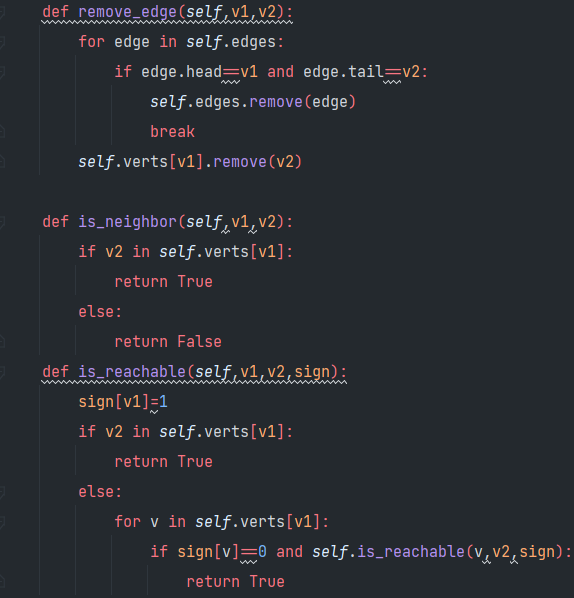


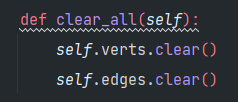
1. 如图，定义材料要求的类











对如图测试数据，有以下结果：

