1. 状态机
2. 累加器Accumulator

Translation\_fn函数：依次传入数组的元素，根据状态机的下一状态等于上一状态加上传入参数的参数的特性编写函数即可完成状态累加器。

Output\_fn函数：返回累加器的当前状态即可。

1. 二进制加法器Binary\_Addition

Translation\_fn函数：采取一个元组表示当前状态值，元组的第一个元素表示进位值，即记录低一位是否产生进位。元组的第二个元素表示已经计算的当前结果，由于二进制是逆序存储的，因此对于每次加法，只需将计算出来的结果插入到列表的结尾即可。对于每次加法，一共有4种可能情况，判断是否产生进位和当前位是0还是1。

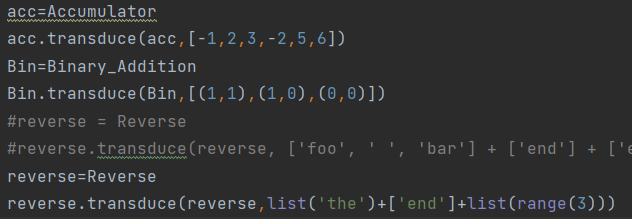
Output\_fn函数：需要再对最终是否产生进位进行一次判断，如果不产生进位，则直接返回元组的第二个参数即可；如果产生进位，则先插入元素1后，再返回元组的第二个元素即可。

1. 反向器Reverse

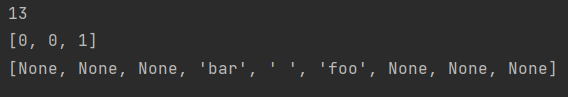
Translation\_fn函数：记录‘end’的位置，最后统一进行计算，使用一个思源元组分别表示状态值是否结束，len的位置，seq1的长度和已输入的列表。

Output\_fn函数：首先输出seq1长度个None，然后依次进行反向输入，最后补齐None即可。

1. 输入：

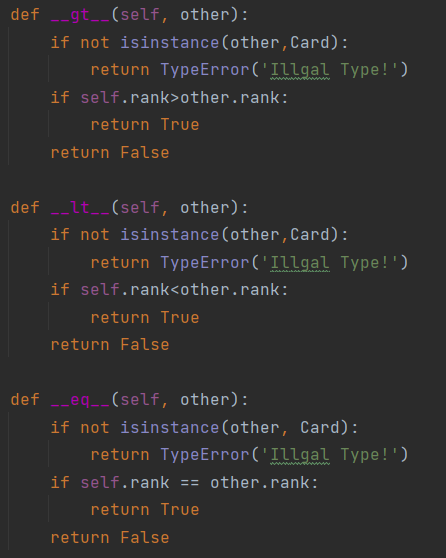


1. 输出：



1. 扑克游戏
2. Card()类

Card()类内为比较函数，因此将比较函数代码补全即可，核心代码：



1. Hand()

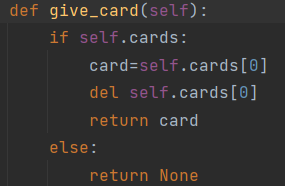
当收到多张牌的时候，直接将收到的牌补充在玩家手牌列表即可



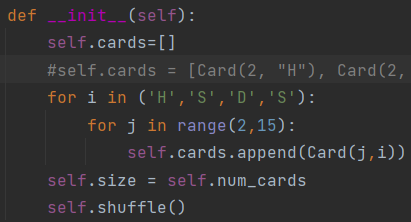
当收到一张牌时，直接插入当前player的手牌列表中即可



补充单张卡牌时，首先进行合法判断，如果为空则返回None，若不为空，则将第一张牌删除，并返回

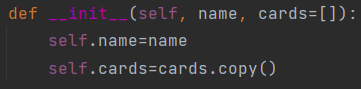


分配52张牌的时候，先循环遍历花色，然后再遍历点数并依次加入牌堆中即可



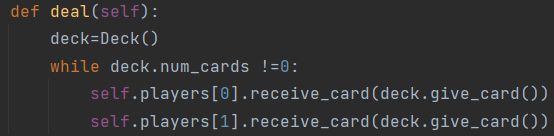
1. Player()

初始化



1. Game()

发牌函数需要实现两个人的轮流取牌，因此只需要让牌堆轮流给两个人一张牌直至排队为空为止



当两方的牌数一直大于0的时候一直进入循环，当得到胜利者，则先将card中的牌给胜利者，并返回，若为得到结果，则进行下一次比较。此外，如果在取牌过程中，尽管当前为平局，但一方的牌被取空，则被取空的一方输；若两方都同时被取空，则为平局。

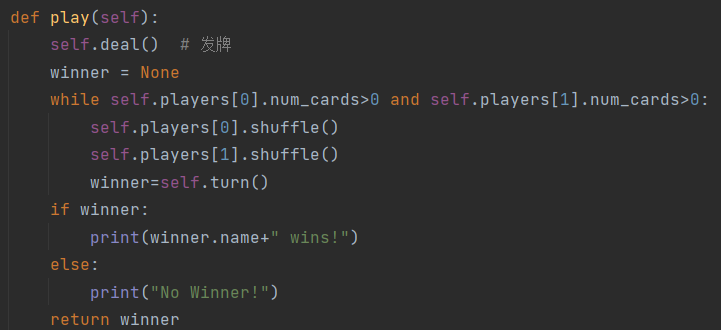
核心代码：



1. Play()

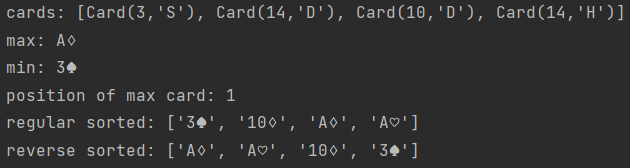
游戏开始时先发牌然后使用while循环判断两方牌的数量并记录最后一轮的获胜者，最终获胜者为最后一轮的获胜者，将结果输出即可。

核心代码：

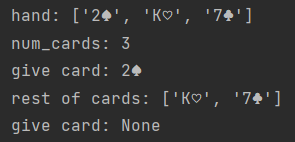


1. 运行结果

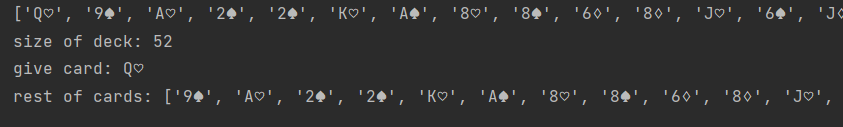
Test\_Card():



Test\_Hand():



Test\_Deck():

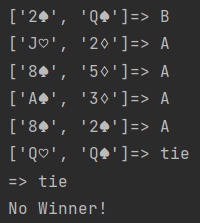


Test\_Player():

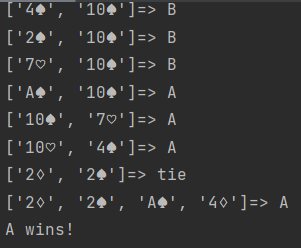


Test\_Game()

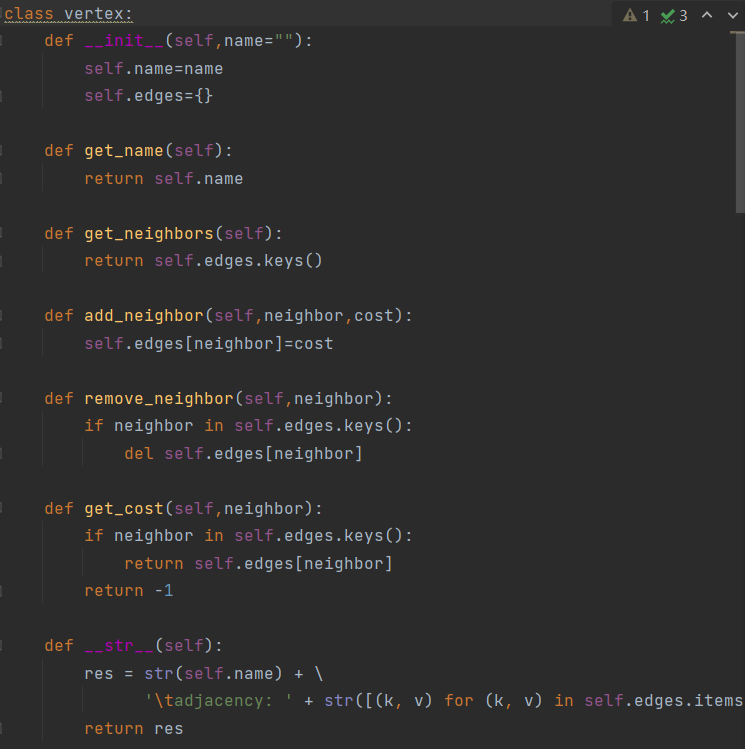
平局：



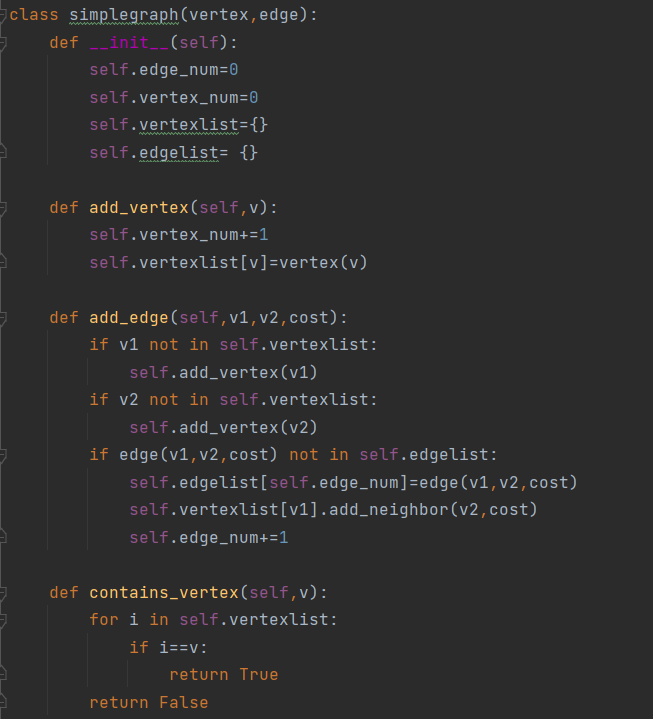
非平局：

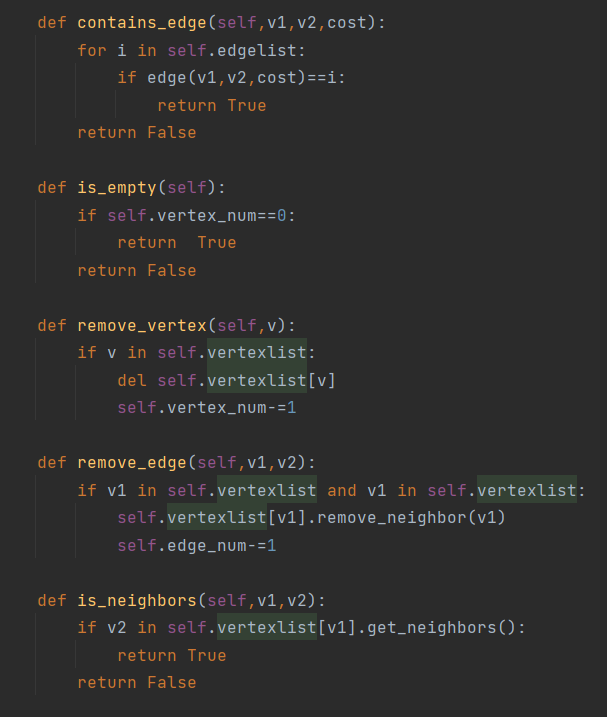


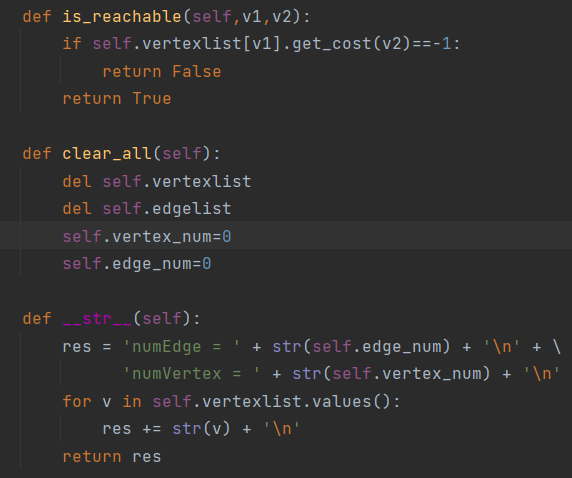
1. SimpleGraph
2. Vertex():



1. Simplegraph():



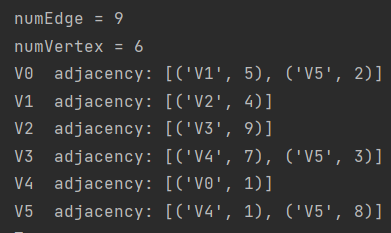




1. 建图

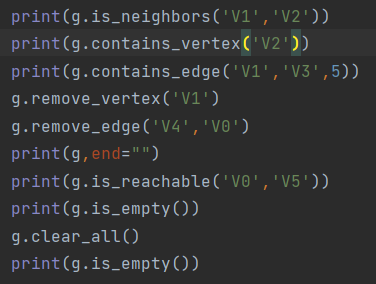


输出：



1. 功能函数的实现

输入：



输出：

