# 面向对象编程指南

## 前言

因为本书是基于21点扑克游戏的，所以介绍21点游戏的简单规则

1.A 可以是1点或10点，如果是当作1点，则手牌为硬牌，否则为软牌

2.Face牌为10点

3.超过21点叫爆牌

4.玩家可以分牌，算是额外的下注

5.开局时发两张牌，庄家的牌一张朝上，一张朝下，玩家的都朝上

6.

## \_\_init\_\_() 方法

### 隐式的基类-object

class X:

pass

>>> X.\_\_class\_\_

<class ‘type’>

>>> X.\_\_class\_\_.\_\_base\_\_

<class ‘object’>

ps.本书是基于Python3.x的，每个类的定义都会隐式继承自object类，Python2.x有新旧两种定义类的方式，新式是显式的继承Object类

### 1.2基类中的\_\_init\_\_()方法

1.尽量避免延迟赋值的实现方式,因为这种多态机制不但给程序带来了隐藏的不确定性，也会相应产生很多令人费解的if语句

2.Zen of python poem 曾提过建议：显式而非隐式

3.对于每个\_\_init\_\_()方法，都应当显式地指定要初始化的变量

### 1.3 在基类中实现\_\_init\_\_()方法

创建对象：

1. 先创建一个空对象
2. 调用对象的\_\_init\_\_()函数(提供了对象内部变量以及其他一些一次性过程的初始化操作)

最普通的多态：

class Card(object):

def \_\_init\_\_(self, rank, suit):

self.suit = suit

self.rank = rank

self.hard, self.soft = self.\_points()

class NumberCard(Card):

def \_points():

return int(self.rank), int(self.rank)

class AceCard(Card):

def \_points():

return 1, 11

class FaceCard(Card):

def \_points():

return 10,10

每个子类为\_points()方法提供特有的实现，所有子类有相同的方法名和属性

### 1.4 使用\_\_init\_\_()方法创建常量清单

1.静态常量构成策略(Strategy)或状态(State)模式的一部分

2.python中没有提供简单而直接的方式来定义一个不可变对象

3.创建策略或状态模式对象的集合，把创建好的花色对象做缓存，构成一个缓存池，使得在调用时对象可被重用，那么性能将得到显著提升。

### 1.5 通过工厂函数调用\_\_init\_\_()

实现工厂有两种途径：

1. 定义一个函数，返回不同类的对象

（Python 更倾向于定义一个函数就能做到的事情没必要去定义类层次结构）

1. 定义一个类，包含了创建对象的方法。这是完整的工厂设计模式(java)