8.4 面向对象应用实例

目录

CONTENTS

例：采用扑克牌类设计扑克牌发牌程序。

4名牌手打牌，电脑随机将52张牌（不含大小王）发给4名牌手，在屏幕上显示每位牌手的牌。程序运行效果如图所示：

```
This is a module with classes for playing cards.
牌手 1:梅7    黑5    梅K    黑2    方A    梅Q    黑6    黑9    黑10
红K    红Q    红A    方10
牌手 2:黑A    方6    方7    方K    黑J    黑3    红5    红8    黑8
方2    黑4    梅2    梅6
牌手 3:梅10   黑K    梅8    黑7    方4    红2    红3    方J    红9
方9    红J    方3    梅9
牌手 4:梅3    梅A    红6    梅4    梅J    方5    红4    方8    黑Q
方Q    梅5    红7    红10

Press the enter key to exit.
```

分析：这个程序中主要解决的问题有三个：

- 如何洗牌
- 如何表示不同花色的牌面，如：梅花10 红桃9 方块K等
- 如何分别显示牌手接收的牌

8.4.1 random库概述

使用random库主要目的是生成随机数。这个库提供了不同类型的随机数函数，其中最基本的函数是random.random()，所有其他随机函数都是基于这个函数扩展而来。

- random.random

random.random()用于生成一个[0.0, 1.0)的随机小数：

```
>>> import random
>>> random.random()
0.4868479203171543
```

- random.uniform

random.uniform(a,b),用于生成一个指定范围内的随机小数，两个参数其中一个是上线，一个是下限。如果a<b，则生成的随机数n的范围为:a<=n<=b.如果a>b,则b<=n<=a.

```
>>> import random
>>> print(random.uniform(10,20))
16.022248415108532
>>> print(random.uniform(20,10))
15.588893238574 94
```

- random.randint

random.randint(a,b),用于随机生成一个指定范围内的整数。其中参数a是下限，参数b是上限，生成的随机数n: a<=n<=b.

```
>>> import random
>>> print(random.randint(12,20))          #生成一个随机数n:12<=n<=20
18
>>> print(random.randint(20,20))          #结果永远是20
20
>>> print(random.randint(20,10))          #该句是错误的，下限必须小于上限
Traceback (most recent call last):
  ......
```

- random.randrange

random.randrange([start],stop[,step]),从指定范围内，按指定基数递增的集合中获取一个随机数。如：random.randrange(10,100,2)结果相当于从[10，12，14，16……96，98]序列中获取一个随机数。

```
>>> import random
>>> random.randrange(10,100,2)
74
```

- random.choice

random.choice（sequence）从序列中获取一个随机元素。参数sequence表示一个有序类型。即sequence在此不是一种特定类型，而是泛指序列数据结构。列表List，元组tuple以及字符串都属于sequence.

```
>>> import random
>>> print(random.choice("学习python"))  #字符串中随机取一个字符
t
>>> print(random.choice(["JGood","is","a","handsome","boy"]))  #list列表中随机取一个元素
boy
>>> print(random.choice(("tuple","list","dict")))  #tuple元组中随机取
list
```

- random.shuffle

random.shuffle(x[,random]),用于将一个列表中的元素顺序打乱。例如：

```
>>> p=["Pyhon","is","powerful","simple","and so on"]
>>> random.shuffle(p)
>>> print(p)
['powerful', 'and so on', 'Pyhon', 'simple', 'is']
```

注：在发牌游戏中我们将利用此方法打乱牌的顺序，实现洗牌功能。

- random.sample

random.sample(sequence,k),从指定序列中随机获取指定长度的片段。sample函数不会修改原有序列。

```
>>> list=[1,2,3,4,5,6,7,8,9,10]
>>> slice=random.sample(list,5)  #从list中随机获取5个元素，作为一个片段返回
>>> print(slice)
[3, 9, 4, 10, 6]
>>> print(list)      #原有序列并没有改变
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

8.4.2 程序设计思路

分析：这个程序中主要解决的问题有三个：

- 如何洗牌
- 如何表示不同花色的牌面，如：梅花10 红桃9 方块K等
- 如何分别显示牌手接收的牌

设计出3个类：Card类、Hand类、Poke类。

- Card类

Card类代表一张牌，我们用FaceNum字段代表牌面数字，Suit字段代表花色，值“梅”为梅花，“方”为放块，“红”为红桃，“黑”为黑桃。

```
# Cards Module
# Basic classes for a game with playing cards
class Card():
    """ A playing card. """
    RANKS = ["A", "2", "3", "4", "5", "6", "7",
             "8", "9", "10", "J", "Q", "K"]          #牌面数字 1--13
    SUITS = ["梅", "方", "红", "黑"]                  #梅为梅花，方为方钻，红为红心，黑为黑桃

    def __init__(self, rank, suit, face_up = True):
        self.rank = rank                             #指的是牌面数字 1--13
        self.suit = suit                             #suit指的是花色
        self.is_face_up = face_up                    #是否显示牌正面，True为正面，False为牌背面

    def __str__(self):
        if self.is_face_up:
            rep = self.suit + self.rank #+" "+ str(self.pic_order())
        else:
            rep = "XX"
        return rep

    def flip(self):                                  #翻牌方法
        self.is_face_up = not self.is_face_up

    def pic_order(self):                             #牌的顺序号
        if self.rank=="A":
            FaceNum=1
        elif self.rank=="J":
            FaceNum=11
        elif self.rank=="Q":
            FaceNum=12
        elif self.rank=="K":
            FaceNum=13
        else:
            FaceNum=int(self.rank)
        if self.suit=="梅":
            Suit=1
        elif self.suit=="方":
            Suit=2
        elif self.suit=="红":
            Suit=3
        else:
            Suit=4
        return (Suit - 1) * 13 + FaceNum
```

其中：

- Card构造函数根据参数初始化封装的成员变量，实现牌面大小和花色的初始化，以及是否显示牌面，默认Ture为显示牌正面。
- _str_()方法用来输出牌面大小和花色。
- pic_order()方法获取牌的顺序号，牌面按梅花1~13，方块14~26，红桃27~39，黑桃40~52顺序编号（未洗牌之前）。也就是说梅花2顺序号为2，方块A顺序号为14，方块K顺序号为26.这个方法图形化显示牌面预留的方法。
- flip()是翻牌方法，改变牌面是否显示的属性值。

Hand类

```
class Hand( ):
    """ A hand of playing cards. """
    def __init__(self):
        self.cards = []

    def __str__(self):                          #重写print()方法
        if self.cards:
            rep = ""
            for card in self.cards:
                rep += str(card) + "\t"
        else:
            rep = "无牌"
        return rep

    def clear(self):
        self.cards = []

    def add(self, card):
        self.cards.append(card)

    def give(self, card, other_hand):
        self.cards.remove(card)
        other_hand.add(card)
```

Hand类代表一手牌，也就是一个玩家手里拿的牌，可以认为是一位牌手手里的牌，其中cards列表变量存储牌手手里的牌。可以增加牌、清空牌手手里的牌、或者把一张牌给别的牌手。

Poke类

```
class Poke(Hand):
    """ A deck of playing cards. """
    def populate(self):                                  #生成一副牌
        for suit in Card.SUITS:
            for rank in Card.RANKS:
                self.add(Card(rank, suit))

    def shuffle(self):                                   #洗牌
        import random
        random.shuffle(self.cards)                       #打乱牌的顺序

    def deal(self, hands, per_hand = 13):                #发牌，发给玩家，每人默认13张牌
        for rounds in range(per_hand):
            for hand in hands:
                if self.cards:
                    top_card = self.cards[0]
                    self.cards.remove(top_card)
                    hand.add(top_card)
                    #self.give(top_card, hand)
                else:
                    print("Can't continue deal. Out of cards!")
```

主程序

```
#主程序

if __name__ == "__main__":
    print("This is a module with classes for playing cards.")
    #四个玩家
    players = [Hand(),Hand(),Hand(),Hand()]
    poke1 = Poke()
    poke1.populate()                    #生成一副牌
    poke1.shuffle()                     #洗牌
    poke1.deal(players,13)              #发给玩家每人13张牌
    #显示4位牌手的牌
    n=1
    for hand in players:
        print("牌手",n ,end=":")
        print(hand)
        n=n+1
    input("\nPress the enter key to exit.")
```

总结与拓展

做什么？

01 本章总结

- 封装性、继承性、多态性
- 类的定义和使用、构造函数、析构函数、实例属性和类属性、私有成员与公有成员、方法
- 类的继承、类的多继承、方法重写、多态、运算符重载

总结与拓展

做什么？

02 拓展作业

用python语言设计斗地主发牌程序。3名牌手打牌，计算机随机留下3张底牌，然后将剩余51张牌（含大小王）发给3名牌手，在屏幕上按照由小到大、黑、红、梅、方、小王、大王（♠♥♣♦♕♔）的顺序显示每位牌手的牌。程序的运行效果参考下图：

```
玩家 1的手牌为:  3♠  4♥  5♥  6♦  6♥  7♦  7♥  7♠  8♣  9♣  9♠  K♥  K♠  A♦  A♥  2♦  2♣
玩家 2的手牌为:  3♣  5♣  5♠  6♠  7♣  8♦  9♦  9♥  J♣  J♥  Q♣  Q♥  Q♠  K♣  2♥  2♠  ♕
玩家 3的手牌为:  3♦  3♥  4♣  4♠  5♦  6♣  8♥  8♠ 10♦ 10♣ 10♥ 10♠  J♦  J♠  Q♦  A♣  ♔
剩余的3张底牌为:  4♦  K♦  A♠
```

THANKS