田忌赛马

田忌赛马的故事

齐威王、田忌、孙膑

对战方式列出来

	第一场	第二场	第三场	获胜方
齐威王	上马	中马	下马	
田忌1	上马（s）	中马(s)	下马(s)	齐威王
田忌2	上（s）	下(s)	中(y)	齐威王
田忌3	中	上	下	
田忌4	中	下	上	
田忌5	下	中	上	
田忌6	下(s)	上(y)	中(y)	田忌

（每匹马只能出场1次）

如果问题变的复杂，比如30场比赛，50匹马，或者再加一些限制条件呢？ 所以有必要学会用计算机计算这些问题。

想想如果用计算机处理这样的问题，我们应该怎么是编写代码？

```
1. 田马的所有出场方式都列出来
2. 一个一个的去判断嫩故不能应
```

问题1：如何遍历田马的所有出场方式

田的上、中、下马都有一个共同点，即都是田的马，就像是上节课的螺旋爆炸的颜色。

我们使用一种“容器”----元组，来存储这些有着特殊关系的数据。

元组

Python的基本数据类型之一，属于**序列型数据**，是一种特殊的“容器”，可以有序存放多个数据，小括号包围起来和数据之间用逗号（英文状态下）隔开。

元组的创建

```
tHouse = ('上马', '中马', '下马') # 这里，元组中的数据是字符串
```

事实上，元组中的数据可以是任意类型，同一元组中数据可以相同，也可以不同。

```

```

元组中的数据也可以是元组（俄罗斯套娃），此时，我们称之为“二维元组”，例如，创建一个二维元组存放田的马和齐的马

```
house = (('上马', '中马', '下马'),('上马', '中马', '下马'))
```

tips:

```
序列型数据
是能够表示一组数据之间特殊关系的一种组合数据类型。序列型数据中的各个元素都是有顺序地进行存放。例如range()函数返回的数字序列、字符串、元组以及之后学的列表。
其实就是，前后之间有顺序，你知道谁的后面是谁，谁的前面是谁，谁排第几
```

访问元组中的值

```
houses = ('上马', '中马', '下马')
print(houses[1])
# '中马'

# 访问二维元组
houses2 = (('q上马', 'q中马', 'q下马'), ('t上马', 't中马', 't下马'))
# 访问田的所有马
print(houses2[0])
# ('q上马', 'q中马', 'q下马')
# 访问田的中马
print(houses2[0][1])
# q中马

# 遍历
houses = ('上马', '中马', '下马')
for i in range(3):
    print(houses[i])
print()
for i in houses:
    print(i)
```

元组的不可变性

```
houses = ('上马', '中马', '下马', 123, (1,2,3))
houses[0] = '下马'
out:
TypeError: 'tuple' object does not support item assignment
不能修改
```

元组的长度

```
houses = ('上马', '中马', '下马', 123, (1,2,3))
print(len(houses))
out:
5
用len()这个函数
```

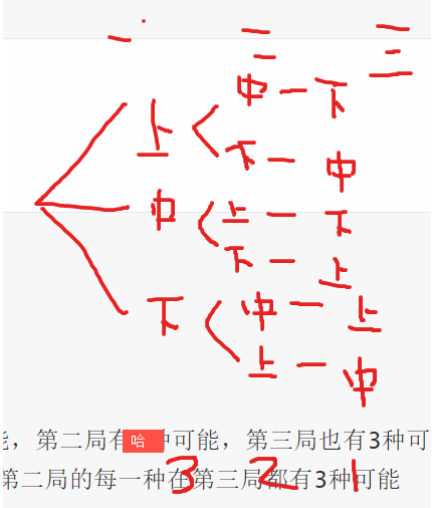
tips

```
有了元组，我们可以把一组有着特殊关系的数据组织起来，并结合for循环，遍历元组中的各个元组，相比一个一个输出，可以减少代码量，提高效率和可读性。
```

不限制马的出场次数

把所有出场顺序的可能打印出来

```
houses = ('上马', '中马', '下马')
# 第一局 有 3 种可能
# 第二局 有 3
# 第三局 有 3
# 3 * 3 * 3 = 27
# 为什么用 乘法， 因为第一局有3种可能，第二局有3种可能，第三局也有3种可能，
# 第一局的每1种在第二局都有3种可能，第二局的每一种在第三局都有3种可能
```



限制马的出场次数

把所有出场顺序的可能打印出来

```
# 第一局 有 3 种可能
# 第二局 有 2
# 第三局 有 1
# 3 * 2 * 1 = 6
# 为什么用 乘法， 因为第一局有3种可能，第二局有2种可能，第三局也有1种可能，
# 第一局的每1种在第二局都有2种可能，第二局的每一种在第三局都有1种可能
```

上，中，下

第一局，第二局，第三局

上 中 下

上 下 中

中 上 下

中 下 上

下 中 上

下 上 中

```
# 第一局比赛 用第i匹马
# 第二局比赛 用第j匹马
# 第三局比赛 用第k匹马
# 不允许同一匹马在三次比赛中出现两次以上
# 也就是说：第一局用的马和第二局不一样， 第二局用的马和第三局不一样（也可以不用我的方法）
```

问题2：判断每种出场方式中田马能否获胜

```
n = 0
houses = ('上马', '中马', '下马')
housesLevel = (3,2,1)
```

```
for i in range(3): # 第一局比赛 用第i匹马
    for j in range(3): # 第二局比赛 用第j匹马
        for k in range(3): # 第三局比赛 用第k匹马
            # 不允许同一匹马在三次比赛中出现两次以上
            # 第一局用的马和第二局不一样， 第二局用的马和第三局不一样
            if i != j and i != k and j!= k:
                # n = n + 1
                # print("出丧顺序{}: {} {} {}".format(n, houses[i],
houses[j], houses[k]))

                # 去判断，田忌的马的等级和齐威王的马的等级，谁的高，高的赢，赢
两局就赢了
                # 齐威王的马是按照顺序出场的，所以，第一局等级为3 第二局等级为
2 第三局等级为1
                win = 0
                if housesLevel[i] > 3:
                    win = win +1
                if housesLevel[j] > 2:
                    win = win + 1
                if housesLevel[k] > 1:
                    win = win + 1
                if win >= 2:
                    print("当出场次序为{},{},{}时，田忌
赢！！！".format(houses[i], houses[j], houses[k]))
                '''
                    win = 0
                    判断第一回合田忌是否获胜
                        win = win +1
                    判断第二回合田忌是否获胜
                        win = win +1
                    判断第三回合田忌是否获胜
                        win = win +1
                    判断win是否大于2，如果大于2那么就赢了
                         print("当出场次序为{}{}{}时，田忌
赢！！！".format(houses[i],houses[j],houses[k]
                    # 简单方法提示：用 or 连接，可以减少 if 的次数
                '''
```

枚举算法