**程序报告**

学号：2213924 姓名：申宗尚

1. **问题重述**

5 个不同国家，工作各不相同的人分别住在一条街上的 5 所房子里，每所房子的颜色不同，每个人都有自己养的不同宠物，喜欢喝不同的饮料。

根据题目给出已知的条件，推理出正确的匹配

**1. 英国人住在红色的房子里**

**2. 西班牙人养了一条狗**

**3. 日本人是一个油漆工**

**4. 意大利人喜欢喝茶**

**5. 挪威人住在左边的第一个房子里**

**6. 绿房子在白房子的右边**

**7. 摄影师养了一只蜗牛**

**8. 外交官住在黄房子里**

**9. 中间那个房子的人喜欢喝牛奶**

**10. 喜欢喝咖啡的人住在绿房子里**

**11. 挪威人住在蓝色的房子旁边**

**12. 小提琴家喜欢喝橘子汁**

**13. 养狐狸的人所住的房子与医生的房子相邻**

**14. 养马的人所住的房子与外交官的房子相邻**

利用Python解决此问题时，与之前的命令式编程有别，解决该问题需要逻辑式编程，能够依据给定事实以及规则编写代码，解决逻辑约束问题。

1. **设计思想**

使用python自带的kanren库进行逻辑推理的实现。kanren库可以通过定义变量，约束条件和关系来解决这类问题。首先定义逻辑变量，定义变量的属性，然后建立约束条件，根据题目的要求，将其用代码表示，接着定义关系来表达变量之间的约束关系，比如left（）左边，next（）相邻，最后利用kanren提供的求解器unify，run等求解满足所有约束条件的解。

**流程：定义问题空间(var)🡪定义逻辑规则(lall)🡪求解器(run)🡪输出结果**

可能的局限性：本题提供的条件较小，然而对于实际逻辑求解问题，条件空间一般十分巨大，尽管kanren 是一个强大的逻辑编程系统，但是对于复杂问题，搜索空间可能会非常大，导致计算资源消耗较多，同时面临性能问题。

1. **代码内容**

def left(q, p, list):  
 return membero((q, p), zip(list, list[1:]))  
def next(q, p, list):  
 return conde([left(q, p, list)], [left(p, q, list)])  
self.rules\_zebraproblem = lall(  
 (eq, (var(), var(), var(), var(), var()), self.units),  
 (membero, ('英国人', var(), var(), var(), '红色'), self.units), # 英国人住在红房子里  
 (membero, ('西班牙人', var(), var(), '狗', var()), self.units), # 西班牙人养了一条狗  
 (membero, ('日本人', '油漆工', var(), var(), var()), self.units), # 日本人是一个油漆工  
 (membero, ('意大利人', var(), '茶', var(), var()), self.units), # 意大利人喝茶。  
 (eq, (('挪威人', var(), var(), var(), var()), var(), var(), var(), var()), self.units), # 挪威人住在左边的第一个房子里  
 (membero, (var(), '摄影师', var(), '蜗牛', var()), self.units), # 摄影师养了一只蜗牛  
 (membero, (var(), '外交官', var(), var(), '黄色'), self.units), # 外交官住在黄房子里  
 (eq, (var(), var(), (var(), var(), '牛奶', var(), var()), var(), var()), self.units),  
 (membero, (var(), var(), '咖啡', var(), '绿色'), self.units), # 喜欢喝咖啡的人住在绿房子里  
 (membero, (var(), '小提琴家', '橘子汁', var(), var()), self.units), # 小提琴家喜欢喝橘子汁  
 (membero, (var(), var(), var(), '斑马', var()), self.units),  
 (membero, (var(), var(), '矿泉水', var(), var()), self.units),  
 (left,  
 (var(), var(), var(), var(), '绿色'), # 绿房子在白房子的右边  
 (var(), var(), var(), var(), '白色'),  
 self.units),  
 (next, ('挪威人', var(), var(), var(), var()),  
 (var(), var(), var(), var(), '蓝色'), self.units), # 挪威人住在蓝房子旁边。  
 (next, (var(), '医生', var(), var(), var()), # 养狐狸的人所住的房子与医生的房子相邻  
 (var(), var(), var(), '狐狸', var()), self.units),  
 (next, (var(), '外交官', var(), var(), var()), # 养马的人所住的房子与外交官的房子相邻  
 (var(), var(), var(), '马', var()), self.units))

1. **实验结果**

五条正确的匹配信息，通过网站测试：

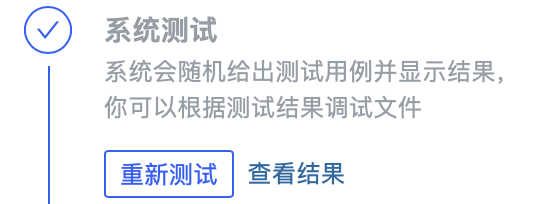
('挪威人', '外交官', '矿泉水', '狐狸', '黄色')

('意大利人', '医生', '茶', '马', '蓝色')

('英国人', '摄影师', '牛奶', '蜗牛', '红色')

('西班牙人', '小提琴家', '橘子汁', '狗', '白色')

('日本人', '油漆工', '咖啡', '斑马', '绿色')



1. **总结**

本次解答基本达到了目标预期，成功地使用了 kanren 库来解决斑马问题。然而，这种解决方法建立在输入条件已知的情况下进行推理，无法应对真实情况中输入条件未知或者不完整的情况。如果要解决实际问题，可能需要引入更大规模的模型，如分词、建立模型等模块。

可能改进的方向：

**1、更灵活的输入处理**： 考虑到真实情况中输入条件可能是动态变化的，可以设计更灵活的输入处理模块，使系统能够适应不同形式的输入条件。

**2、引入更多的知识**： 除了已有的逻辑规则外，可以考虑引入更多的领域知识来增强系统的推理能力，例如利用自然语言处理技术从文本中提取信息。

**3、优化算法设计**： 可以进一步优化算法设计，提高求解效率，如通过引入启发式搜索等技术来加速求解过程。

在求解斑马问题的过程中，最大的困难是对kanren库的理解与运用，但这次实验也让我学会了很多，对于未来python其他库的学习会更加得心应手，胸有成竹。