人工智能的现代方法实验一：斑马问题

实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 封兆欣 | 学号 | 202011010083 | 专业 | 教育技术学 |

# 斑马问题

1. 求解此类问题时，首先对问题中出现的名词进行分类，然后寻找各类名词之间对应的关系，请将以下关系补充完整：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 国家 | 工作 | 饮料 | 宠物 | 颜色 |
| 英国  西班牙  日本  意大利  挪威 | 油漆工  摄影师  外交官  小提琴家  医生 | 茶  牛奶  咖啡  橘子汁  水 | 狗  蜗牛  狐狸  马  斑马 | 红色  绿色  白色  黄色  蓝色 |

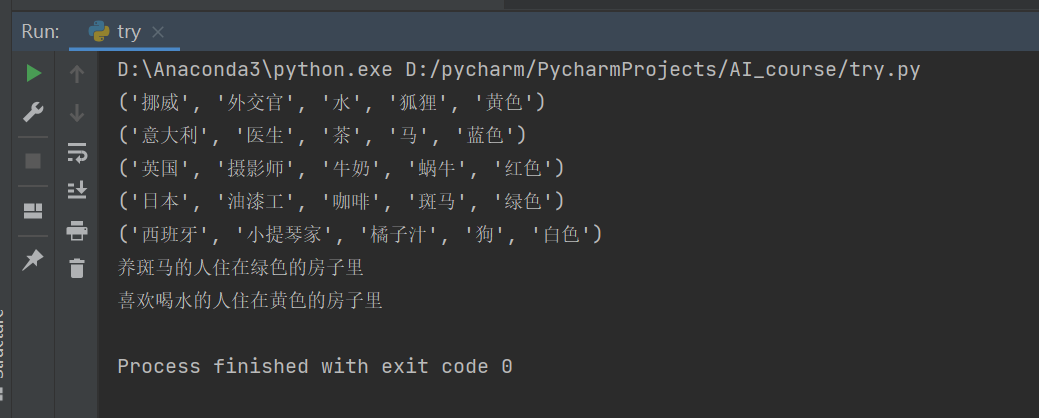
按照斑马问题的条件，手动推理完成下面的表格

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 国家 | 工作 | 饮料 | 宠物 | 颜色 |
| 挪威 | 外交官 | 水 | 狐狸 | 黄色 |
| 意大利 | 医生 | 茶 | 马 | 蓝色 |
| 英国 | 摄影师 | 牛奶 | 蜗牛 | 红色 |
| 日本 | 油漆工 | 咖啡 | 斑马 | 绿色 |
| 西班牙 | 小提琴家 | 橘子汁 | 狗 | 白色 |

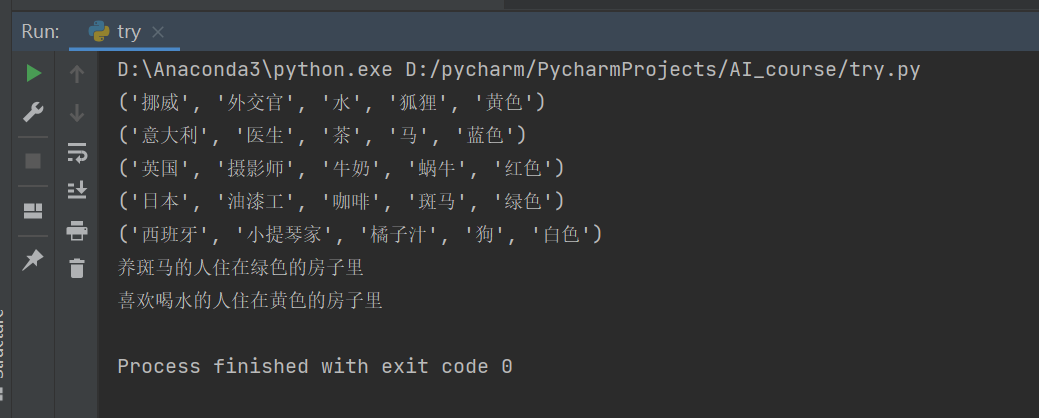
1. 给出斑马问题的逻辑编程实现(适当增加注释)：

from kanren import run, eq, membero, var, conde # kanren一个描述性Python逻辑编程系统  
from kanren.core import lall # lall包用于定义规则  
import time  
  
  
class Agent:  
 *"""  
 推理智能体.  
 """* def \_\_init\_\_(self):  
 *"""  
 智能体初始化.  
 """* self.units = var() # 单个unit变量指代一座房子的信息(国家，工作，饮料，宠物，颜色)  
 self.rules\_zebraproblem = None # 用lall包定义逻辑规则  
 self.solutions = None # 存储结果  
  
 def define\_rules(self):  
 *"""  
 定义逻辑规则  
 """* # left()表示在左边,next()表示在旁边  
 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)])  
  
 # 定义规则  
 self.rules\_zebraproblem = lall(  
 eq((var(), var(), var(), var(), var()), self.units),  
 # 英国人住在红房子里  
 (membero, ('英国', var(), var(), var(), '红色'), self.units),  
 # 西班牙人养狗  
 (membero, ('西班牙', var(), var(), '狗', var()), self.units),  
 # 日本人是一名油漆工  
 (membero, ('日本', '油漆工', var(), var(), var()), self.units),  
 # 意大利人喝茶  
 (membero, ('意大利', var(), '茶', var(), var()), self.units),  
 # 挪威人住在左边的第一个房子里  
 eq((('挪威', var(), var(), var(), var()), var(), var(), var(), var()), self.units),  
 # 绿房子在白房子的右边  
 (left,  
 (var(), var(), var(), var(), '绿色'),  
 (var(), var(), var(), var(), '白色'),  
 self.units),  
 # 摄影师养了一只蜗牛  
 (membero, (var(), '摄影师', var(), '蜗牛', var()), self.units),  
 # 外交官住在黄房子里  
 (membero, (var(), '外交官', var(), var(), '黄色'), self.units),  
 # 中间那个房子的人喜欢喝牛奶  
 eq((var(), var(), (var(), var(), '牛奶', var(), var()), var(), var()), self.units),  
 # 喜欢喝咖啡的人住在绿色的房子里  
 (membero, (var(), var(), '咖啡', var(), '绿色'), self.units),  
 # 挪威人住在蓝房子旁边  
 (next, ('挪威', var(), var(), var(), var()),  
 (var(), var(), var(), var(), '蓝色'), self.units),  
 # 小提琴家喜欢喝橘子汁  
 (membero, (var(), '小提琴家', '橘子汁', var(), var()), self.units),  
 # 养狐狸的人所住的房子与医生的房子相邻  
 (next, (var(), '医生', var(), var(), var()),  
 (var(), var(), var(), '狐狸', var()), self.units),  
 # 养马的人所住的房子与外交官的房子相邻  
 (next, (var(), '外交官', var(), var(), var()),  
 (var(), var(), var(), '马', var()), self.units),  
 # 将水和斑马加进去  
 (membero, (var(), var(), var(), '斑马', var()), self.units),  
 (membero, (var(), var(), '水', var(), var()), self.units))  
  
 def solve(self):  
 self.define\_rules()  
 self.solutions = run(0, self.units, self.rules\_zebraproblem)  
 return self.solutions  
  
  
agent = Agent()  
solutions = agent.solve()  
  
for j in range(0,5):  
 print(solutions[0][j])  
  
# 输出养斑马的人  
for i in range(0,5):  
 list = solutions[0][i]  
 if list[3] == '斑马':  
 print('养斑马的人住在{}的房子里'.format(list[4]))  
# 输出喝水的人  
for i in range(0,5):  
 list = solutions[0][i]  
 if list[2] == '水':  
 print('喜欢喝水的人住在{}的房子里'.format(list[4]))

1. 斑马问题答案截图：



1. 斑马问题中所有信息的输出截图：



# 实验总结

本实验的关键是使用membero构建成员关系表达式，使用eq构建等价关系表达式，同时定义left与next关系表示在左邻近和邻近关系，然后依次表示出14个已知条件，并加上题目中未知的“马”和“水”，最后使用run函数解出答案。

# 描述Kanren包中run函数的运行原理（选做）

kanren的逻辑推理通过run函数实验，函数参数为run(n, var(), rules,[rules, ...])，该函数能够根据指定的规则运行得出符合规则的变量结果。