人工智能程序设计


python
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
turtle.seth(-40)
for i in range(4):
turtle.circle(40, 80)
turtle.circle(-40, 80)
turtle.fd(40)
turtle.circle(40, 80/2)
turtle.fd(40)
turtle.circle(16, 180)
turtle.fd(40 * 2/3)



人工智能程序设计

4.2 模式匹配语句MATCH

北京石油化工学院 人工智能研究院

刘 强

4.2.1 match语句基本语法

```
def process_grade(letter_grade):
    """根据字母成绩给出评价"""
    if letter_grade == 'A’:
            return "优秀！继续保持！"
    elif letter_grade == 'B’:
            return "良好，还有提升空间"
    elif letter_grade == 'C’:
            return "中等，需要更加努力"
    elif letter_grade == 'D’:
            return "及格，但需要大幅提升"
    elif letter_grade == 'F’:
            return "不及格，需要重修" else: return "无效的成绩等级"
```

4.2.1 match语句基本语法

match语句的基本结构由一个match关键字和多个case分支组成。

程序会按顺序检查每个case分支，当找到第一个匹配的模式时，执行对应的代码块，然后退出整个match语句。

这与if-elif-else的执行逻辑相似，但语法更加简洁。

```
match value:
    case pattern1:
        # 当value匹配pattern1时执行
        statement1
    case pattern2:
        # 当value匹配pattern2时执行
        statement2
    case _:
        # 默认情况（可选）
        default_statement
```

4.2.2 简单值匹配

```
def process_grade(letter_grade):
        """根据字母成绩给出评价"""
        match letter_grade:
            case 'A':
                return "优秀！继续保持！"
            case 'B':
                return "良好，还有提升空间"
            case 'C':
                return "中等，需要更加努力"
            case 'D':
                return "及格，但需要大幅提升"
            case 'F':
                return "不及格，需要重修"
            case _:
                return "无效的成绩等级"
```

基础值匹配

将一个变量与一系列具体的值进行比较。

每个case后面跟一个字面值（如数字、字符串、布尔值等），当被匹配的值与某个case的字面值相等时，执行该分支的代码。

```
## 测试
print(process_grade('A'))  # 优秀！继续保持！
print(process_grade('D'))  # 及格，但需要大幅提升
print(process_grade('X'))  # 无效的成绩等级
```

4.2.3 多值匹配

多值匹配：允许在一个case分支中使用竖线符号|连接多个模式，表示"或"的关系。只要被匹配的值与其中任何一个模式相等，就会执行该分支的代码。

```
def get_season(month):
    """根据月份判断季节"""
    match month:
        case 12 | 1 | 2:  # 使用 | 表示"或"
            return "冬季"
        case 3 | 4 | 5:
            return "春季"
        case 6 | 7 | 8:
            return "夏季"
        case 9 | 10 | 11:
            return "秋季"
        case _:
            return "无效月份"
```

```
## 测试
for month in [1, 4, 7, 10, 13]:
    print(f"{month}月是{get_season(month)}")
```

Match语句的特点

match语句的主要特点：

- 简洁清晰：相比多层if-elif，代码结构更清晰
- 模式匹配：不仅可以匹配值，还可以匹配结构
- 性能优化：在某些情况下比if-elif链更高效

4.2.4 Ask AI：探索 match语句的高级用法

当你想要深入探索 match语句的更多功能时，可以向AI助手提出以下问题：

- "match语句与 if-elif语句有什么区别和优势？"
- "如何使用 match语句处理复杂的数据结构？"
- "什么是守卫条件（Guard），如何在 match中使用？"
- "如何用 match语句实现状态机或处理JSON数据？"
- "match语句的最佳实践和常见陷阱有哪些？"

实践练习

练习 4.2.1：交通信号灯

使用 match语句实现交通信号灯逻辑，根据不同颜色（红、黄、绿）输出对应的行动指令。

练习 4.2.2：文件类型识别

根据文件扩展名，使用 match语句判断文件类型（如：图片、文档、视频、音频等）。

练习 4.2.3：星期几判断

编写程序使用 match语句，根据输入的数字（1-7）返回对应的星期几，并说明是工作日还是周末。