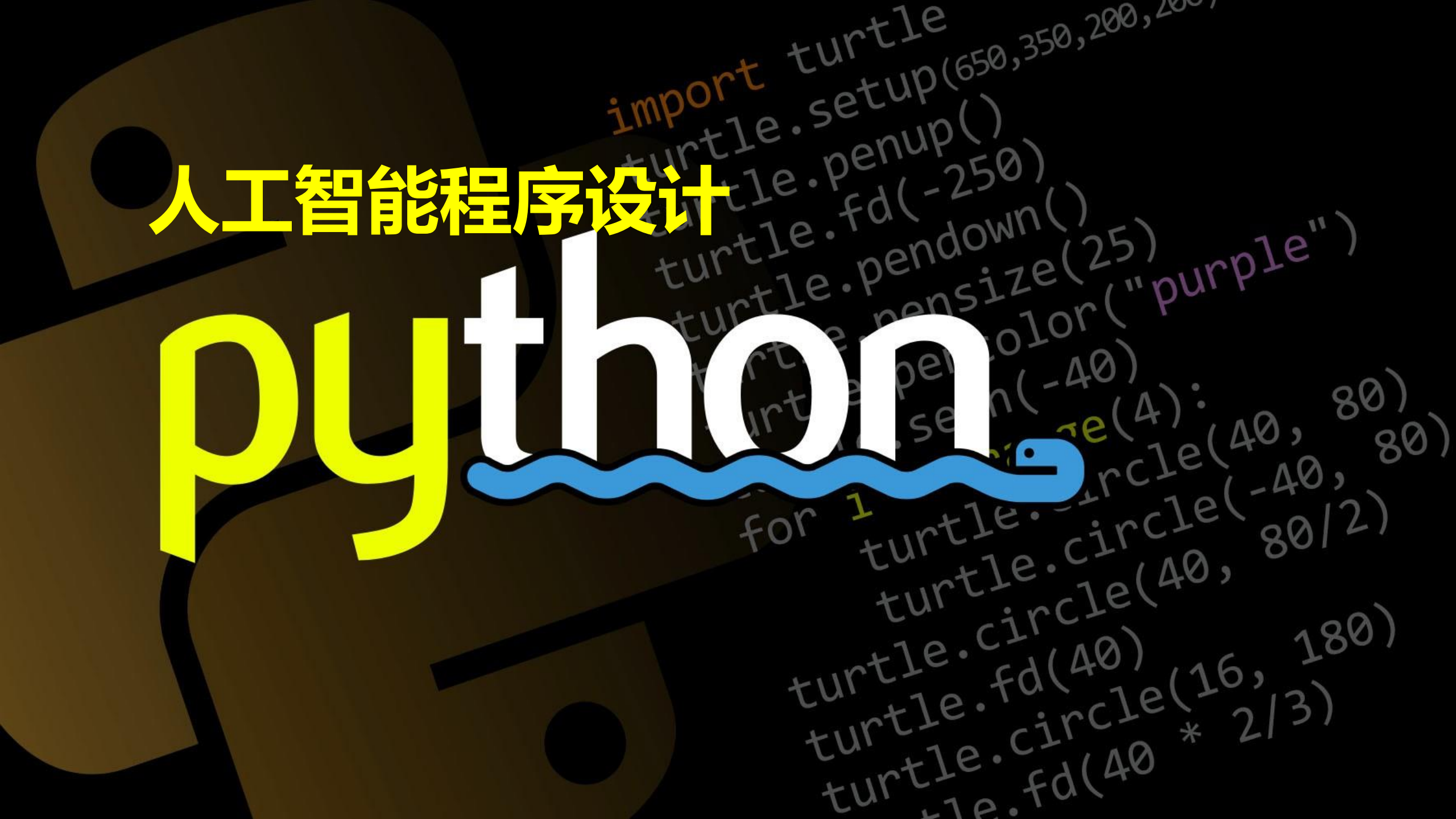
人工智能程序设计


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 1 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

3.4 列表推导式

北京石油化工学院 人工智能研究院

刘 强

3.4 列表推导式

列表推导式（List Comprehension）是Python中创建列表的一种简洁而优雅的方式。

它允许我们用一行代码完成原本需要多行for循环才能实现的功能，使代码更加Pythonic。

（Ask AI：什么是Pythonic？）

3.4 列表推导式

语法结构

列表推导式的基本语法结构为：

[expression for item in iterable]

3.4.1 基本语法

简单示例

将普通的for循环转换为列表推导式：

```
## 传统方法：使用for循环
squares = []
for x in range(1, 6):
    squares.append(x ** 2)
print(squares)  # 输出：[1, 4, 9, 16, 25]
## 列表推导式方法
squares = [x ** 2 for x in range(1, 6)]
print(squares)  # 输出：[1, 4, 9, 16, 25]
```

3.4.1 基本语法

字符串处理

```
## 将字符串列表转换为大写
    names = ["alice", "bob", "charlie"]
    upper_names = [name.upper() for name in names]
    print(upper_names)  # 输出：['ALICE', 'BOB', 'CHARLIE']

    ## 获取每个字符串的长度
    lengths = [len(name) for name in names]
    print(lengths)  # 输出：[5, 3, 7]
```

3.4.1 基本语法

数学运算

```
## 价格转换：人民币转美元（汇率7.2）
rmb_prices = [100, 200, 300, 400]
usd_prices = [price / 7.2 for price in rmb_prices]
print(usd_prices)  # 输出：[13.89, 27.78, 41.67, 55.56]
```

3.4.1 基本语法

数学运算

```
## 计算圆的面积
import math
radii = [1, 2, 3, 4, 5]
areas = [math.pi * r ** 2 for r in radii]
print([round(area, 2) for area in areas])  # 输出：[3.14, 12.57, 28.27,
50.27, 78.54]
```

3.4.2 条件过滤

基本条件过滤

在列表推导式中添加if条件进行过滤：

```
## 筛选偶数
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_numbers = [n for n in numbers if n % 2 == 0]
print(even_numbers)  # 输出：[2, 4, 6, 8, 10]

## 筛选正数
mixed_numbers = [-3, -1, 0, 2, 5, -8, 7]
positive = [n for n in mixed_numbers if n > 0]
print(positive)  # 输出：[2, 5, 7]
```

3.4.2 条件过滤

字符串过滤

```
## 筛选长度大于3的单词
words = ["cat", "dog", "elephant", "ant", "tiger"]
long_words = [word for word in words if len(word) > 3]
print(long_words)  # 输出：['elephant', 'tiger']

## 筛选包含特定字母的单词
words_with_a = [word for word in words if 'a' in word]
print(words_with_a)  # 输出：['cat', 'elephant']
```

3.4.2 条件过滤

条件表达式

结合条件表达式进行更复杂的处理：

```
## 将负数转换为0，正数保持不变
numbers = [-2, 3, -1, 4, -5, 6]
processed = [n if n > 0 else 0 for n in numbers]
print(processed)  # 输出：[0, 3, 0, 4, 0, 6]

## 成绩等级转换
scores = [85, 92, 78, 96, 73, 88]
grades = ['优秀' if score >= 90 else '良好' if score >= 80 else '及格' for score in scores]
print(grades)  # 输出：['良好', '优秀', '及格', '优秀', '及格', '良好']
```

3.4.3 Ask AI：高级列表推导式

列表推导式还有更多高级用法。

当需要处理复杂数据或提升性能时，可以向AI助手询问：

- "如何使用嵌套列表推导式处理二维列表？"
- "列表推导式与for循环的性能区别是什么？"
- "如何用列表推导式处理文件数据？"

实践练习

练习 3.4.1：数据处理

给定一个包含产品信息的列表：["苹果:12.5", "香蕉:8.0", "橙子:15.2", "葡萄:25.8"]

使用列表推导式提取：

1. 所有产品的名称
2. 所有产品的价格（转换为浮点数）
3. 价格大于15的产品名称

练习 3.4.2：文本分析

给定一个句子列表，使用列表推导式：

1. 统计每个句子的单词数
2. 提取所有长度超过5个字符的单词
3. 将所有单词转换为小写并去重

实践练习

练习 3.4.3：数学计算

使用列表推导式：

1. 生成1到20中所有3的倍数
2. 计算1到10每个数字的立方
3. 找出1到100中所有能被7整除但不能被14整除的数