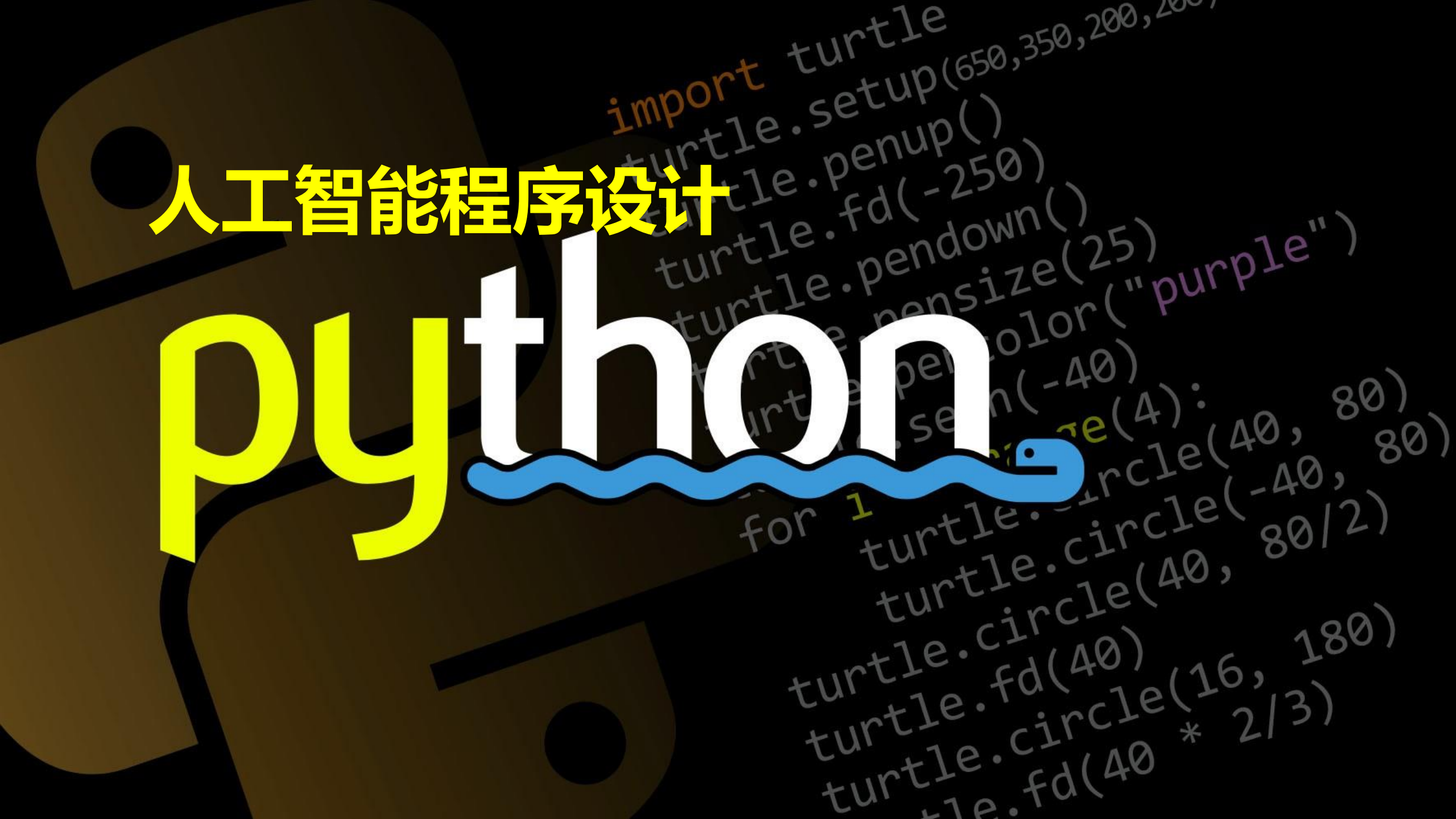
人工智能程序设计


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

10.5 PYTHON数据库编程

北京石油化工学院 人工智能研究院

刘 强

数据库简介

数据库是现代数据科学和应用开发的重要组成部分：

- 用于持久化存储和高效管理大量结构化数据
- 在数据科学项目中，经常需要从数据库读取数据进行分析
- 或将处理结果存储到数据库中

10.5.1 数据库基本概念

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库：

- **数据持久化**：数据存储在磁盘上，程序关闭后数据不会丢失
- **数据共享**：多个用户和程序可以同时访问数据
- **数据完整性**：通过约束确保数据的准确性和一致性
- **数据安全性**：提供访问控制和权限管理
- **高效查询**：通过索引和优化提供快速的数据检索

关系型数据库基本概念

关系型数据库由表、行、列等基本元素组成：

- **表（Table）**：存储数据的基本单位，由行和列组成
- **行（Row/Record）**：一条记录，包含一个实体的完整信息
- **列（Column/Field）**：一个字段，表示实体的一个属性
- **主键（Primary Key）**：唯一标识表中每一行的字段
- **外键（Foreign Key）**：引用其他表主键的字段，用于建立表间关系

SQL语言基础

SQL（Structured Query Language）是操作关系型数据库的标准语言：

- **DDL（数据定义语言）**：CREATE、ALTER、DROP 等，用于定义数据库结构
- **DML（数据操作语言）**：SELECT、INSERT、UPDATE、DELETE 等，用于操作数据
- **DCL（数据控制语言）**：GRANT、REVOKE 等，用于权限控制

10.5.2 SQLite数据库操作

`SQLite` 是一个轻量级的关系型数据库管理系统：

- **无服务器**：不需要单独的数据库服务器进程
- **零配置**：无需安装和配置
- **跨平台**：支持多种操作系统
- **自包含**：整个数据库存储在一个文件中
- **Python内置支持**：Python 标准库包含 `sqlite3` 模块

创建数据库和表

创建 SQLite 数据库文件和员工表，流程包括：连接数据库、创建游标、执行 SQL、提交事务：

```
import sqlite3

## 连接数据库，如果文件不存在会自动创建
conn = sqlite3.connect('company.db')
## 创建游标对象，用于执行SQL语句
cursor = conn.cursor()

## 创建员工表，IF NOT EXISTS避免重复创建错误
cursor.execute('''
CREATE TABLE IF NOT EXISTS employees (
    id INTEGER PRIMARY KEY,
    name TEXT,
    department TEXT,
    salary REAL
)
''')

conn.commit()
print("数据库创建成功")
```

插入数据

向员工表中批量插入数据，使用 **executemany** 函数提高效率：

```
## 准备要插入的员工数据
employees_data = [
    ('张三', '技术部', 8000),
    ('李四', '市场部', 7500),
    ('王五', '人事部', 6800)
]

## 使用executemany批量插入，?占位符防止SQL注入
cursor.executemany('''
INSERT INTO employees (name, department, salary)
VALUES (?, ?, ?)
''', employees_data)

conn.commit()
print("数据插入成功")
```

查询数据

从数据库中检索数据，使用 **fetchall** 获取结果集：

```
## 查询所有员工
cursor.execute('SELECT name, department, salary FROM employees')
employees = cursor.fetchall()

print("员工信息: ")
for emp in employees:
    print("姓名:", emp[0], "部门:", emp[1], "薪资:", emp[2])

## 条件查询：使用WHERE子句和参数化查询
cursor.execute('SELECT name, salary FROM employees WHERE department = ?',
               ('技术部',))
tech_employees = cursor.fetchall()

print("技术部员工: ")
for emp in tech_employees:
    print("姓名:", emp[0], "薪资:", emp[1])
```

更新和删除数据

使用 **UPDATE** 和 **DELETE** 语句修改和删除数据库中的记录：

```
## 使用UPDATE语句修改数据
cursor.execute('UPDATE employees SET salary = ? WHERE name = ?',
               (8800, '张三'))

## 使用DELETE语句删除数据
cursor.execute('DELETE FROM employees WHERE name = ?', ('王五',))

conn.commit()
print("数据更新成功")
```

使用上下文管理器

使用 Python 的 **with** 语句自动管理数据库连接，确保资源正确释放：

```
## with语句自动处理连接的打开、提交事务和关闭
with sqlite3.connect('company.db') as conn:
    cursor = conn.cursor()
    cursor.execute('SELECT name, salary FROM employees WHERE department = ?',
                   ('技术部',))
    employees = cursor.fetchall()

print("技术部员工: ", employees)
```

10.5.3 数据库操作封装

封装数据库操作为可复用的函数，提高代码的可维护性：

```
def query_database(query, params=None):
    """查询数据库的通用函数"""
    with sqlite3.connect('company.db') as conn:
        cursor = conn.cursor()
        if params:
            cursor.execute(query, params)
        else:
            cursor.execute(query)
        return cursor.fetchall()

## 使用通用函数执行查询
employees = query_database('SELECT name, salary FROM employees WHERE salary > ?',
                           (7000,))
print("高薪员工: ")
for emp in employees:
    print("姓名:", emp[0], "薪资:", emp[1])
```

更新操作封装

封装 INSERT、UPDATE、DELETE 等更新操作，与查询函数配合使用，构成完整的数据库操作工具：

```
def update_database(query, params=None):
    """更新数据库的通用函数（INSERT、UPDATE、DELETE）"""
    with sqlite3.connect('company.db') as conn:
        cursor = conn.cursor()
        if params:
            cursor.execute(query, params)
        else:
            cursor.execute(query)
        ## with语句自动提交事务
```

10.5.4 Ask AI：数据库进阶应用

掌握了数据库基础后，可以向 AI 助手询问更多高级数据库技术：

- "如何编写复杂的 SQL 连接查询？"
- "如何设计规范化的数据库表结构？"
- "如何连接和使用 MySQL、PostgreSQL 等数据库？"
- "什么是 NoSQL 数据库，何时使用？"

实践练习

练习 10.5.1：用户行为数据库

1. 创建一个用户行为分析数据库，包含用户表和行为记录表
2. 插入示例数据并实现基本的增删改查操作
3. 编写查询语句统计用户活跃度和行为偏好

实践练习

练习 10.5.2：数据分析应用

1. 将CSV文件数据导入SQLite数据库
2. 使用SQL查询进行数据分析和统计
3. 将查询结果可视化展示

实践练习

练习 10.5.3：数据库管理工具

1. 创建通用的数据库操作工具类
2. 实现数据的批量导入和导出功能
3. 添加数据验证和错误处理机制

本节小结

- 数据库用于持久化存储和管理结构化数据
- SQLite 是轻量级数据库，Python 内置支持
- SQL 基本操作：CREATE、INSERT、SELECT、UPDATE、DELETE
- 使用参数化查询防止 SQL 注入
- 使用上下文管理器自动管理连接