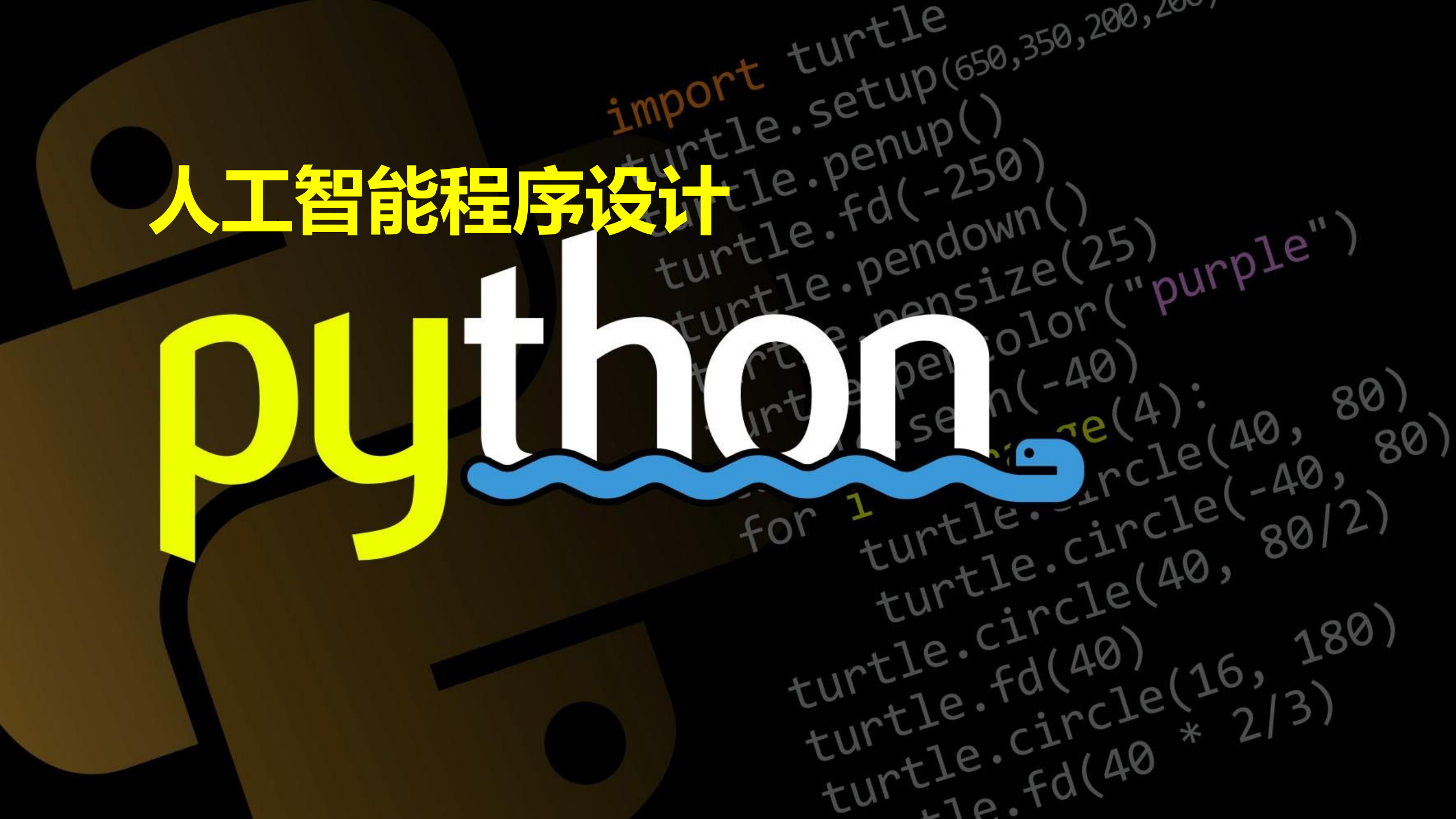
人工智能程序设计


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

第8章 PYTHON WEB应用开发

北京石油化工学院 人工智能研究院

刘 强

第8章 Python Web应用开发

Web应用开发是现代软件开发的核心技能。

无论是数据科学项目的展示、业务系统的构建，还是API服务的开发，Web技术都是连接程序与用户的重要桥梁。

Python在Web开发领域拥有Streamlit、Flask、Django、FastAPI等多种成熟框架，从快速原型开发到企业级应用，都有完善的解决方案。

传统桌面应用与Web应用的差异

传统桌面应用需要在每台计算机上安装配置，更新升级需要重新分发安装包，跨平台支持困难。用户获取和使用应用的门槛较高，数据共享和协作也面临诸多挑战。

Web应用则通过浏览器访问，无需安装即可使用，自动获取最新版本。天然支持跨平台，数据集中存储便于协作，通过网络可以触达全球用户。Web应用还能够轻松集成各种在线服务和API，实现数据分析、支付处理、地图服务等多种功能。

Python在Web开发中的优势在于：拥有Streamlit、Flask、Django、FastAPI等成熟框架，简洁的语法提高开发效率，Pandas、NumPy、Matplotlib等数据处理库便于开发数据驱动的Web应用，与前端技术良好集成。

8.1 Python Web编程概述

在开始具体的Web开发学习之前，我们需要建立对Web编程的整体认识。

本节将介绍Web应用的基本概念、HTTP协议的工作原理、Python在Web开发中的优势，以及主要的技术选择。

8.1.1 Web应用基本概念

什么是Web应用

Web应用（Web Application）是运行在Web服务器上，通过浏览器访问和使用的软件程序。

与传统的桌面应用不同，Web应用具有以下特点：

- 跨平台访问：用户只需要浏览器就能使用，无需安装特定软件
- 实时更新：应用更新后用户立即可以使用新功能
- 数据集中管理：数据存储在服务器端，便于管理和备份
- 多用户协作：支持多个用户同时访问和操作

8.1.1 Web应用基本概念

客户端-服务器模型

Web应用基于客户端-服务器（Client-Server）架构。在这种架构中，客户端（通常是浏览器）负责向服务器发送请求并显示响应结果，而服务器则负责接收请求、处理业务逻辑并返回响应。

客户端主要承担用户界面的展示和交互处理工作。当用户在浏览器中点击链接、提交表单或执行其他操作时，客户端会将这些操作转换为HTTP请求发送给服务器，然后接收服务器的响应并将结果渲染为用户可以看到的网页内容。

服务器端则专注于业务逻辑的处理和数据管理。它接收来自客户端的HTTP请求，根据请求的内容执行相应的程序逻辑，可能涉及数据库查询、文件操作、计算处理等，最后将处理结果组织成HTTP响应发送回客户端。

8.1.1 Web应用基本概念

静态网页 vs 动态网页

静态网页是指内容固定不变的网页，服务器直接将预先编写好的HTML文件发送给客户端。这类网页的内容在创建后就确定了，每次访问都显示相同的信息，主要适用于公司介绍、产品展示等不需要频繁更新的展示类网站。

动态网页则是根据用户请求实时生成内容的网页。服务器接收到请求后，会运行相应的程序代码，可能从数据库中查询数据，根据用户的不同情况或输入参数生成不同的页面内容。这类网页能够提供个性化的用户体验和交互功能，适用于电商网站、社交平台、在线服务等需要用户交互的应用。

Python主要用于开发动态Web应用，能够处理复杂的业务逻辑和数据操作。

8.1.2 HTTP协议基础

HTTP协议简介

HTTP（HyperText Transfer Protocol）是Web通信的基础协议，它规定了客户端（通常是浏览器）与服务器之间如何进行数据交换。

HTTP协议采用请求-响应模式，客户端发送请求到服务器，服务器处理请求后返回响应。

HTTP协议基于 TCP/IP协议栈，通常使用80端口（HTTP）或443端口（HTTPS），支持多种数据格式的传输，包括 HTML、JSON、XML、图片、视频等。

8.1.2 HTTP协议基础

HTTP请求与响应

HTTP通信由请求和响应两部分组成，每次交互都遵循这个基本模式。

HTTP请求结构：

HTTP请求由请求行、请求头和请求体三部分组成。请求行包含 HTTP方法、请求路径和协议版本；请求头提供关于请求的元数据信息，如客户端类型、接受的数据格式等；请求体包含要发送给服务器的数据（如表单数据、JSON数据）。

GET /api/users HTTP/1.1

Host: example.com

User-Agent: Mozilla/5.0

Accept: application/json

[请求体]

8.1.2 HTTP协议基础

HTTP响应结构：

HTTP响应同样由状态行、响应头和响应体三部分组成。状态行包含协议版本、状态码和状态描述；响应头提供关于响应的元数据信息，如内容类型、内容长度等；响应体包含服务器返回给客户端的实际数据。

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: 123

[响应体]

8.1.2 HTTP协议基础

常用HTTP方法

HTTP方法（也称为 HTTP动词）定义了对资源执行的操作类型。不同的方法有不同的语义和用途，遵循 RESTful设计原则可以让 API更加直观和易于理解。

方法	用途	特点	示例
GET	获取资源	安全、幂等	获取用户列表
POST	创建资源	非幂等	注册新用户
PUT	更新资源	幂等	修改用户信息
DELETE	删除资源	幂等	删除用户账号

RESTful 架构风格

RESTful设计的核心思想：

RESTful（Representational State Transfer，表述性状态转移）是一种Web服务的架构风格，由Roy Fielding在2000年提出。它将网络中的一切内容视为"资源"，每个资源都有唯一的URL地址。

使用标准的HTTP方法对资源进行操作（GET获取、POST创建、PUT更新、DELETE删除），通过URL定位资源，通过HTTP状态码表示操作结果。这种设计让API具有良好的可读性和统一性，例如GET /users/123表示获取ID为123的用户信息，DELETE /users/123表示删除该用户。

8.1.3 核心技术栈

Web开发涉及多个技术层面，需要前端和后端技术的协同配合。

前端技术主要负责用户界面的展示和交互，包括 HTML用于构建页面结构，CSS用于设计样式和布局，JavaScript用于实现动态交互功能。

后端技术则专注于业务逻辑处理和数据管理。Python Web开发主要使用 Flask、Django、FastAPI等框架来处理 HTTP请求和响应。数据存储采用SQLite、MySQL、PostgreSQL等关系型数据库，或者 MongoDB、Redis等非关系型数据库。

8.1.4 Python Web框架对比

Python在Web开发中具有显著优势，其简洁易读的语法让开发更加高效，完善的生态系统提供了 Flask、Django、FastAPI等多种框架选择。

Streamlit：专为数据科学应用设计，让开发者仅用Python代码就能创建交互式Web应用。

Gradio：专注于机器学习模型的快速部署和演示，让开发者能够为任何机器学习模型创建交互式Web界面。

Flask：轻量级微框架，适合小到中型Web应用和 API服务开发

Django：功能完整的全栈框架，内置 ORM、模板系统、用户认证、管理界面等功能，特别适合大型Web应用、内容管理系统和企业级项目

FastAPI：专为构建高性能 API设计，异步特性使其在高并发场景下表现出色，特别适合 API服务开发和微服务架构。

框架选择指南

框架	特点	适用场景	学习难度	性能表现
Streamlit	数据科学专用	数据可视化、原型	简单	中等
Gradio	模型部署专用	AI模型演示、学术展示	简单	中等
Flask	轻量级、灵活	小型应用、API	简单	中等
Django	功能完整、全栈	大型应用、企业级	中等	良好
FastAPI	现代化、高性能	API服务、微服务	中等	优秀

实践练习

练习 8.1.1：Web概念理解

1. 解释客户端-服务器模型的工作原理
2. 说明静态网页和动态网页的区别
3. 列举Web应用相比桌面应用的优势

练习 8.1.2：HTTP协议实践

使用浏览器开发者工具：

1. 观察一次完整的 HTTP请求和响应
2. 识别不同的 HTTP方法和状态码
3. 分析 URL的各个组成部分

Ask AI：Firefox浏览器的开发者工具怎样使用?

实践练习

练习 8.1.3：技术栈认知

分析Web开发技术栈：

1. 列举前端开发需要掌握的基础技术
2. 说明后端技术在Web应用中的作用
3. 解释开发工具和部署工具的重要性

练习 8.1.4：框架选择分析

针对以下场景，选择最适合的Python Web框架并说明理由：

1. 为公司制作销售数据分析仪表板
2. 开发个人博客网站
3. 构建 RESTful API服务
4. 创建在线教育平台