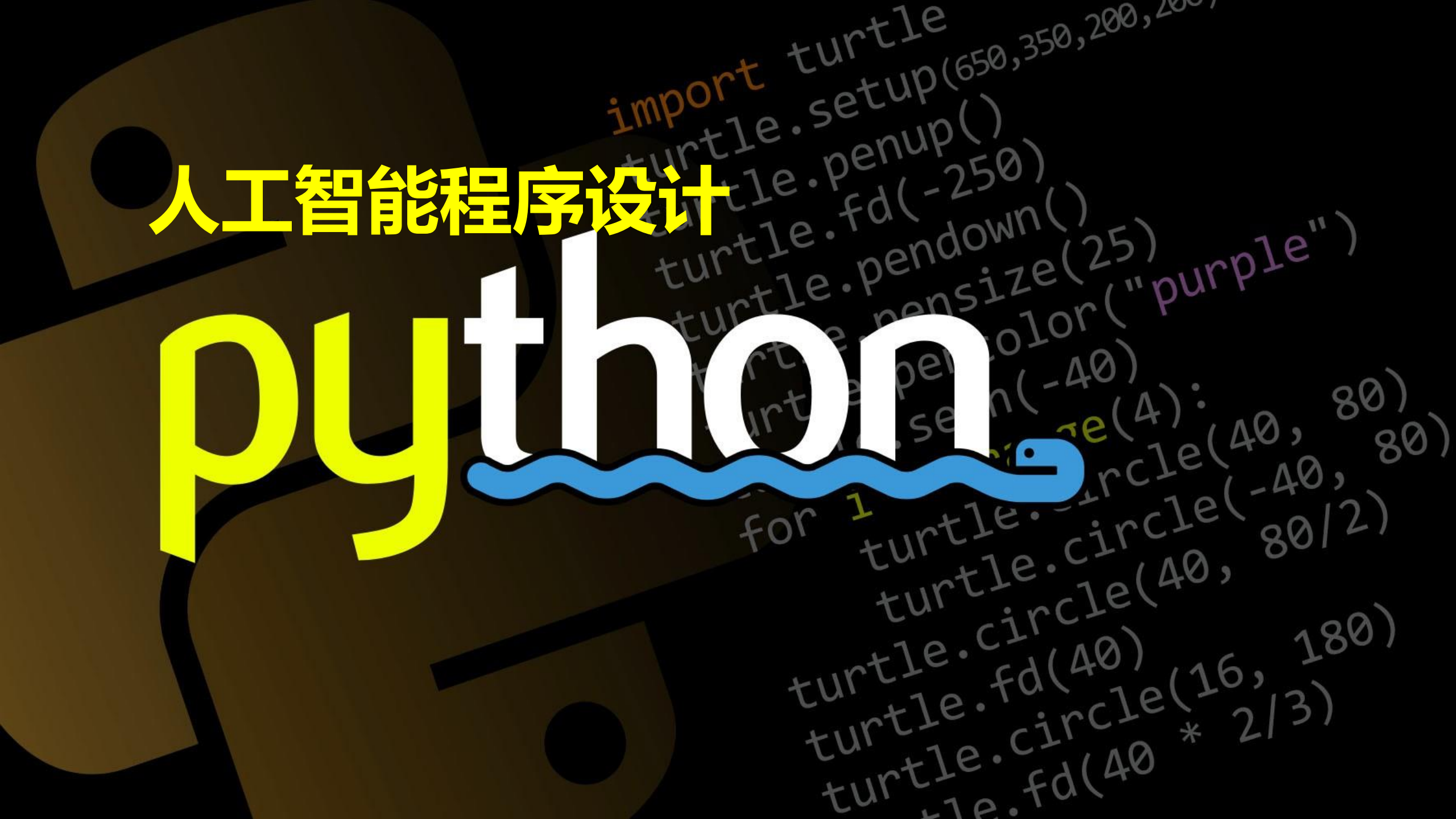
人工智能程序设计


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

8.3 WEB框架：FLASK

北京石油化工学院 人工智能研究院

刘 强

8.3 Web框架：Flask

Flask是一个轻量级的Python Web框架，以其简洁性和灵活性著称。

它采用"微框架"的设计理念，提供了Web开发的核心功能，同时保持了高度的可扩展性。本节将学习如何使用Flask构建基础的Web应用。

8.3.1 Flask简介与环境搭建

Flask的设计理念

Flask遵循"简单胜于复杂"的设计哲学，具有以下特点：

- 微框架架构：核心功能精简，通过扩展提供额外功能
- 灵活性高：开发者可以自由选择组件和架构
- 学习曲线平缓：易于理解和掌握
- 社区活跃：完善的官方文档和大量第三方扩展

8.3.1 Flask简介与环境搭建

环境搭建：安装Flask和相关依赖：

```
## 安装Flask
pip install Flask

## 验证安装
python -c "import flask; print(flask.__version__)"
```

示例 8.3.1：Hello Flask Web应用

```
## app.py
from flask import Flask

## 创建Flask应用实例
app = Flask(__name__)

## 定义路由和视图函数
@app.route('/')
def hello():
    return '<h1>欢迎使用Flask！</h1>'

@app.route('/user/<name>')
def user(name):
    return f'<h1>你好，{name}！</h1>'

## 运行应用
if __name__ == '__main__':
    app.run()
```

运行应用：

python app.py

浏览器会自动打开，显示我们的第一个Flask应用，访问 http://localhost:5000 查看效果。

一个基本的Flask应用包含三个核心要素：创建应用实例、定义路由函数、启动应用服务。

8.3.2 核心功能与基础应用

路由组件

路由是Flask应用的核心，用于将URL映射到Python函数。在掌握基础路由后，我们需要学习更高级的路由功能：

动态路由与类型转换：Flask支持在URL中指定参数类型，如 <int:id>、<float:price>、<path:filepath>等，系统会自动进行类型转换和验证。

多参数路由：路由可以包含多个动态参数，用于构建更复杂的URL结构。

HTTP方法：可以通过methods参数指定路由支持的HTTP方法，如GET、POST等。

8.3.2 核心功能与基础应用

路由组件

```
## 动态路由与类型转换
@app.route('/post/<int:post_id>')
def show_post(post_id):
    return f'文章ID：{post_id}（类型：{type(post_id).__name__}）'

## 多参数路由
@app.route('/user/<username>/post/<int:post_id>')
def user_post(username, post_id):
    return f'用户{username}的第{post_id}篇文章'

## 指定HTTP方法
@app.route('/login', methods=['GET', 'POST'])
def login():
    from flask import request
    if request.method == 'POST':
        return '处理登录'
    return '显示登录页面'
```

8.3.2 核心功能与基础应用

模板组件

模板系统让我们能够分离HTML和Python代码，创建动态网页：

模板渲染使用 render_template()函数将数据传递给HTML模板，实现动态内容生成。

变量传递可以将Python变量传递给模板，在HTML中使用 {{ 变量名 }}显示。

8.3.2 核心功能与基础应用

模板组件

```
from flask import render_template

@app.route('/welcome/<name>')
def welcome(name):
    return render_template('welcome.html', username=name)
```

创建模板文件 templates/welcome.html：

```
<!DOCTYPE html>
<html>
<head>
    <title>欢迎页面</title>
</head>
<body>
    <h1>欢迎，{{ username }}！</h1>
    <p>这是一个Flask模板示例。</p>
</body>
</html>
```

8.3.2 核心功能与基础应用

表单组件

表单处理是Web应用的重要功能，用于接收用户输入：

表单数据获取通过 request.form获取POST数据，request.args获取GET参数。

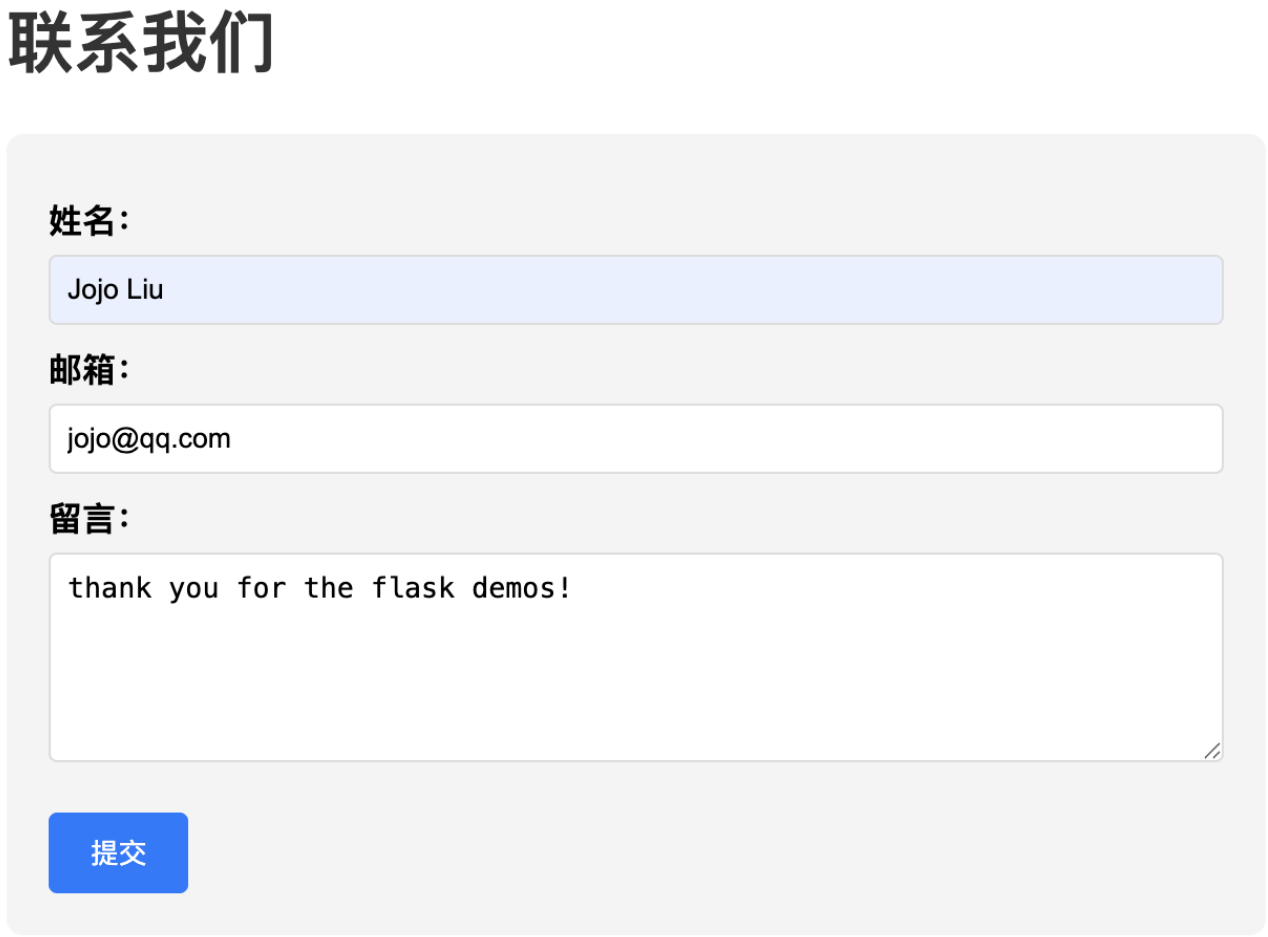
表单验证可以对用户输入进行基本的验证和处理。

8.3.2 核心功能与基础应用

表单组件

```
from flask import request, redirect, url_for

@app.route('/contact', methods=['GET', 'POST'])
def contact():
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        message = request.form['message']

        # 简单处理表单数据
        print(f'收到留言：{name} ({email}): {message}')
        return redirect(url_for('thank_you'))

    return render_template('contact.html')

@app.route('/thank-you')
def thank_you():
    return '<h1>感谢您的留言！</h1>'
```

8.3.2 核心功能与基础应用



Flask的表单组件

8.3.2 核心功能与基础应用

数据组件

简单的数据存储和处理：

内存存储使用Python数据结构临时存储数据，适合简单应用和学习。

JSON响应可以返回JSON格式的数据，用于API接口。

8.3.2 核心功能与基础应用

数据组件

```
from flask import jsonify

## 简单的内存数据存储
users = [
    {'id': 1, 'name': '张三', 'email': 'zhangsan@example.com'},
    {'id': 2, 'name': '李四', 'email': 'lisi@example.com'}
]

@app.route('/api/users')
def get_users():
    return jsonify(users)

@app.route('/api/users/<int:user_id>')
def get_user(user_id):
    user = next((u for u in users if u['id'] == user_id), None)
    if user:
        return jsonify(user)
    return jsonify({'error': '用户不存在'}), 404
```

8.3.3 运行Flask应用

在开发阶段，Flask提供了便捷的内置开发服务器。最简单的运行方式是直接执行Python文件：

python app.py

应用启动后，会在终端显示运行信息：

```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
```

访问http://localhost:5000即可查看应用效果。终端会实时显示每个HTTP请求的信息，包括请求方法、路径和响应状态码。

8.3.4 Ask AI：深入学习Flask

掌握了Flask基础后，可以向AI助手询问更多高级特性：

- "如何在Flask中实现用户认证和会话管理？"
- "如何使用Flask-SQLAlchemy进行数据库操作？"
- "如何在Flask中处理文件上传和下载？"
- "如何在Flask中实现RESTful API？"

实践练习

练习 8.3.1：个人主页

创建一个简单的个人主页应用，包含：

1. 首页显示个人信息
2. 关于页面介绍个人经历
3. 联系页面显示联系方式
4. 使用模板实现页面布局

实践练习

练习 8.3.2：留言板应用

开发一个简单的留言板，功能包括：

1. 显示所有留言列表
2. 添加新留言的表单
3. 留言的基本信息展示（姓名、内容、时间）
4. 简单的留言验证

实践练习

练习 8.3.3：简单API接口

构建一个基础的API服务，包含：

1. 获取用户列表的API接口
2. 根据ID获取单个用户信息
3. 添加新用户的POST接口
4. 返回JSON格式的响应数据