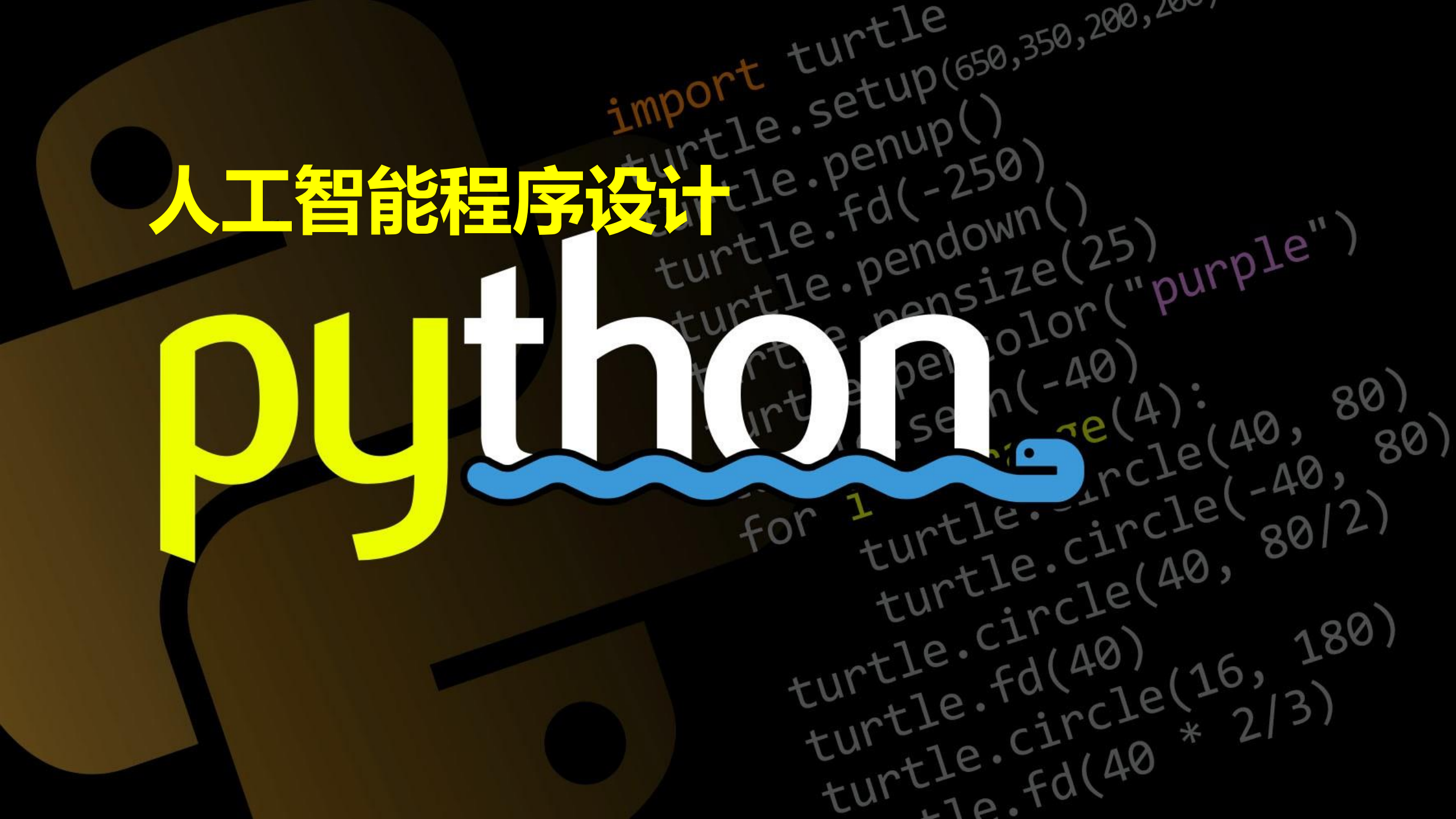
人工智能程序设计


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

15.2 语音识别

北京石油化工学院 人工智能研究院

刘 强

学习内容

- 语音识别系统架构
- 声学建模与语言建模技术
- Python语音识别实现

语音识别概述

语音识别技术将人类的语音信号转换为可理解的文本，是语音处理领域最重要的应用之一。掌握语音识别的基本原理和Python实现方法，是构建智能语音应用的核心技能。

15.2.1 语音识别基本原理

学习内容：

- 传统与端到端系统架构
- 声学建模技术
- 语言建模技术

传统语音识别系统架构

传统语音识别系统采用模块化架构，包括四个主要组件：

组件	功能
特征提取	从语音信号提取声学特征（如MFCC）
声学模型	建立特征与音素的映射关系
语言模型	提供语言学约束，提高识别准确率
解码器	搜索最优的文本输出序列

端到端语音识别系统

端到端语音识别系统使用深度神经网络直接建立语音信号到文本的映射关系。

优势：

- 简化系统架构
- 联合优化所有组件
- 提高识别性能

代表模型： Whisper、DeepSpeech、Conformer

声学建模技术

深度神经网络声学模型使用多层神经网络学习语音特征到音素的映射关系。

常用架构：

- 深度前馈网络（DNN）
- 循环神经网络（RNN）
- 卷积神经网络（CNN）
- Transformer

注意力机制与CTC算法

注意力机制：

- 允许模型动态关注输入序列的不同部分
- 提高长序列语音的识别准确率

CTC算法（Connectionist Temporal Classification）：

- 解决语音序列和文本序列长度不匹配问题
- 使端到端训练成为可能

语言建模技术

类型	特点
统计语言模型	基于n-gram统计，计算词汇出现概率
神经语言模型	使用RNN/Transformer，捕获长距离依赖
领域自适应	在特定领域语料上微调，提高场景识别率

15.2.2 Python语音识别实现

学习内容：

- SpeechRecognition库基础
- 多种识别引擎对比
- 实时语音识别

SpeechRecognition库

SpeechRecognition是Python中最常用的语音识别库，支持多种识别引擎。

安装方式：

```
pip install SpeechRecognition
pip install pyaudio  # 麦克风支持
```

示例 15.2.1：基础语音识别系统

使用SpeechRecognition库实现从麦克风录音到文本转换的完整流程。

```
import speech_recognition as sr

recognizer = sr.Recognizer()

with sr.Microphone() as source:
    audio = recognizer.listen(source)

text = recognizer.recognize_google(audio, language='zh-CN')
print("识别结果: {}".format(text))
```

多种识别引擎对比

引擎	特点	适用场景
Google Speech	高质量在线服务，多语言支持	通用场景
百度语音	中文优化，准确率高	中文应用
讯飞语音	中文表现优秀	中文应用
PocketSphinx	离线识别，无需网络	隐私要求高的场景

音频文件识别

从本地音频文件中提取语音内容并转换为文本，支持WAV、FLAC等无损格式。

```
with sr.AudioFile('speech.wav') as source:
    audio = recognizer.record(source)
    text = recognizer.recognize_google(audio, language='zh-CN')
    print("识别结果: {}".format(text))
```

实时语音识别

通过循环录音和识别实现连续的语音转文字，包含环境噪声适应和超时处理机制。

```
def real_time_recognition():
    with sr.Microphone() as source:
        recognizer.adjust_for_ambient_noise(source)

    while True:
        with sr.Microphone() as source:
            audio = recognizer.listen(source, timeout=1, phrase_time_limit=5)
        text = recognizer.recognize_google(audio, language='zh-CN')
        print("实时识别: {}".format(text))
```

环境噪声处理

adjust_for_ambient_noise() 方法用于适应环境噪声：

```
with sr.Microphone() as source:
    # 采集1秒环境噪声样本
    recognizer.adjust_for_ambient_noise(source, duration=1)
    # 开始录音
    audio = recognizer.listen(source)
```

错误处理机制

语音识别常见的异常处理：

```
try:
    text = recognizer.recognize_google(audio, language='zh-CN')
except sr.UnknownValueError:
    print("无法识别语音内容")
except sr.RequestError as e:
    print("识别服务请求失败: {}".format(e))
```

实践练习

练习 15.2.1：基础语音识别

搭建基本的语音识别环境，实现从麦克风录音并转换为文字的功能。

实践练习

练习 15.2.2：音频文件批处理

编写程序批量处理音频文件，将多个语音文件转换为对应的文本文件。

实践练习

练习 15.2.3：多语言识别对比

比较不同语音识别引擎在中英文识别上的准确率和响应速度差异。

实践练习

练习 15.2.4：实时识别应用

开发一个实时语音识别应用，支持连续识别和结果显示，包含基本的错误处理机制。