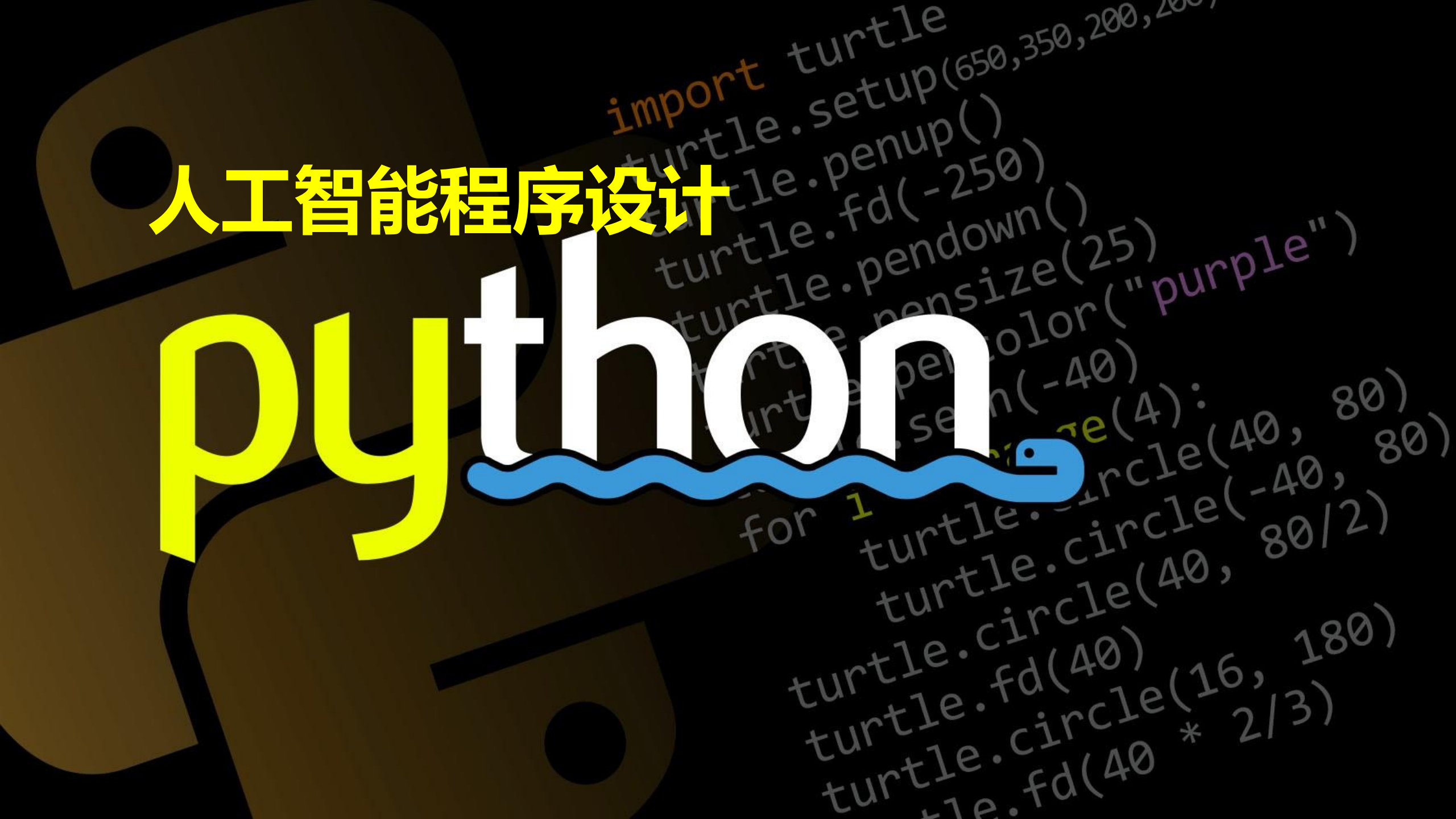
人工智能程序设计


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

15.4 语音交互

北京石油化工学院 人工智能研究院

刘 强

学习内容

- 语音唤醒技术
- 语音情感识别
- 语音交互应用开发

语音交互概述

语音交互技术包括语音唤醒和情感识别等核心功能，这些技术使语音系统能够提供更自然、更智能的交互体验。

掌握这些核心技术是构建语音交互应用的关键。

15.4.1 语音唤醒技术

学习内容：

- 语音唤醒原理
- 唤醒词检测算法
- 实时唤醒系统实现

语音唤醒原理

关键词检测（Keyword Spotting）技术能够在连续的音频流中检测特定的唤醒词。

常见唤醒词：

- "小爱同学"
- "Hey Siri"
- "Alexa"
- "OK Google"

语音唤醒设计要点

设计要点	说明
低功耗设计	持续监听音频，需要低功耗算法和硬件
误唤醒控制	设置合适阈值，减少环境噪声造成的误唤醒
响应速度	快速响应，提升用户体验

唤醒词检测算法

使用MFCC特征提取和余弦相似度比较实现唤醒词检测。

```
import librosa
from sklearn.metrics.pairwise import cosine_similarity

# 提取MFCC特征
mfcc = librosa.feature.mfcc(y=audio_data, sr=16000, n_mfcc=13)
features = mfcc.T
```

滑动窗口匹配

通过滑动窗口与模板进行相似度匹配。

```
# 滑动窗口匹配
max_similarity = 0
for i in range(len(features) - len(template) + 1):
    window = features[i:i+len(template)]
    similarity = cosine_similarity(window.flatten().reshape(1, -1),
                                   template.flatten().reshape(1, -1))[0][0]
    max_similarity = max(max_similarity, similarity)

# 判断是否检测到唤醒词
wake_detected = max_similarity > 0.8
```

实时唤醒系统实现

使用pyaudio实现连续音频流监听和实时唤醒词检测。

```
import pyaudio
import numpy as np

# 初始化音频流
p = pyaudio.PyAudio()
stream = p.open(format=pyaudio.paFloat32, channels=1,
                rate=16000, input=True)
```

音频缓冲与实时监听

实现音频缓冲区和实时监听循环。

```
# 音频缓冲区
audio_buffer = np.array([])
buffer_size = 32000  # 2秒音频

# 实时监听循环
while True:
    chunk = np.frombuffer(stream.read(1024), dtype=np.float32)
    audio_buffer = np.append(audio_buffer, chunk)

    if len(audio_buffer) >= buffer_size:
        current_audio = audio_buffer[-buffer_size:]
        # 调用唤醒词检测逻辑
        print("检测到唤醒词！")
        break
```

15.4.2 语音情感识别

学习内容：

- 情感识别基础
- 特征提取方法
- 情感分类模型

语音情感特征

语音情感特征包括多种声学特征，不同情感状态下呈现不同模式。

特征类型	说明
基频变化	情感激动时基频升高
语速	焦虑时语速加快
音量	愤怒时音量增大
频谱特征	MFCC等频谱特征

情感分类体系

常用的情感分类方式：

基本情感分类：

- 高兴、悲伤、愤怒、恐惧、惊讶、厌恶

二维情感模型：

- 激活度（Arousal）：情感强度
- 效价（Valence）：积极/消极

特征提取方法

提取基频、MFCC等声学特征用于情感识别。

```
import librosa
import numpy as np

# 提取基频特征
f0, _, _ = librosa.pyin(audio, fmin=80, fmax=400)
f0_mean = np.nanmean(f0)

# 提取MFCC特征
mfcc = librosa.feature.mfcc(y=audio, sr=16000, n_mfcc=13)
mfcc_mean = np.mean(mfcc, axis=1)
```

组合特征向量

提取多种特征并组合为特征向量。

```
# 提取频谱质心
spectral_centroid = np.mean(
    librosa.feature.spectral_centroid(y=audio, sr=16000))

# 组合特征向量
features = np.concatenate([
    [f0_mean, spectral_centroid],
    mfcc_mean
])
```

情感分类模型

使用随机森林算法对语音特征进行情感分类。

```
# 特征标准化
features_scaled = scaler.transform(features.reshape(1, -1))

# 情感分类
emotion_labels = ['happy', 'sad', 'angry', 'neutral']
emotion_idx = classifier.predict(features_scaled)[0]
confidence = max(classifier.predict_proba(features_scaled)[0])

# 输出结果
predicted_emotion = emotion_labels[emotion_idx]
print("预测情感: {}, 置信度: {:.2f}".format(predicted_emotion, confidence))
```

情感识别应用场景

应用场景	说明
智能客服	检测用户情绪，调整服务策略
心理健康	监测情绪变化，及时预警
教育培训	了解学习者状态，个性化教学
人机交互	情感感知的智能助手

实践练习

练习 15.4.1：唤醒词检测器

实现一个简单的唤醒词检测系统，能够识别自定义的唤醒词。

实践练习

练习 15.4.2：情感识别系统

构建语音情感识别系统，能够识别基本的情感状态。

实践练习

练习 15.4.3：语音交互应用

结合唤醒词检测和情感识别技术，开发一个完整的语音交互应用。

实践练习

练习 15.4.4：系统性能测试

测试语音交互系统的准确率和响应速度，分析影响性能的关键因素。