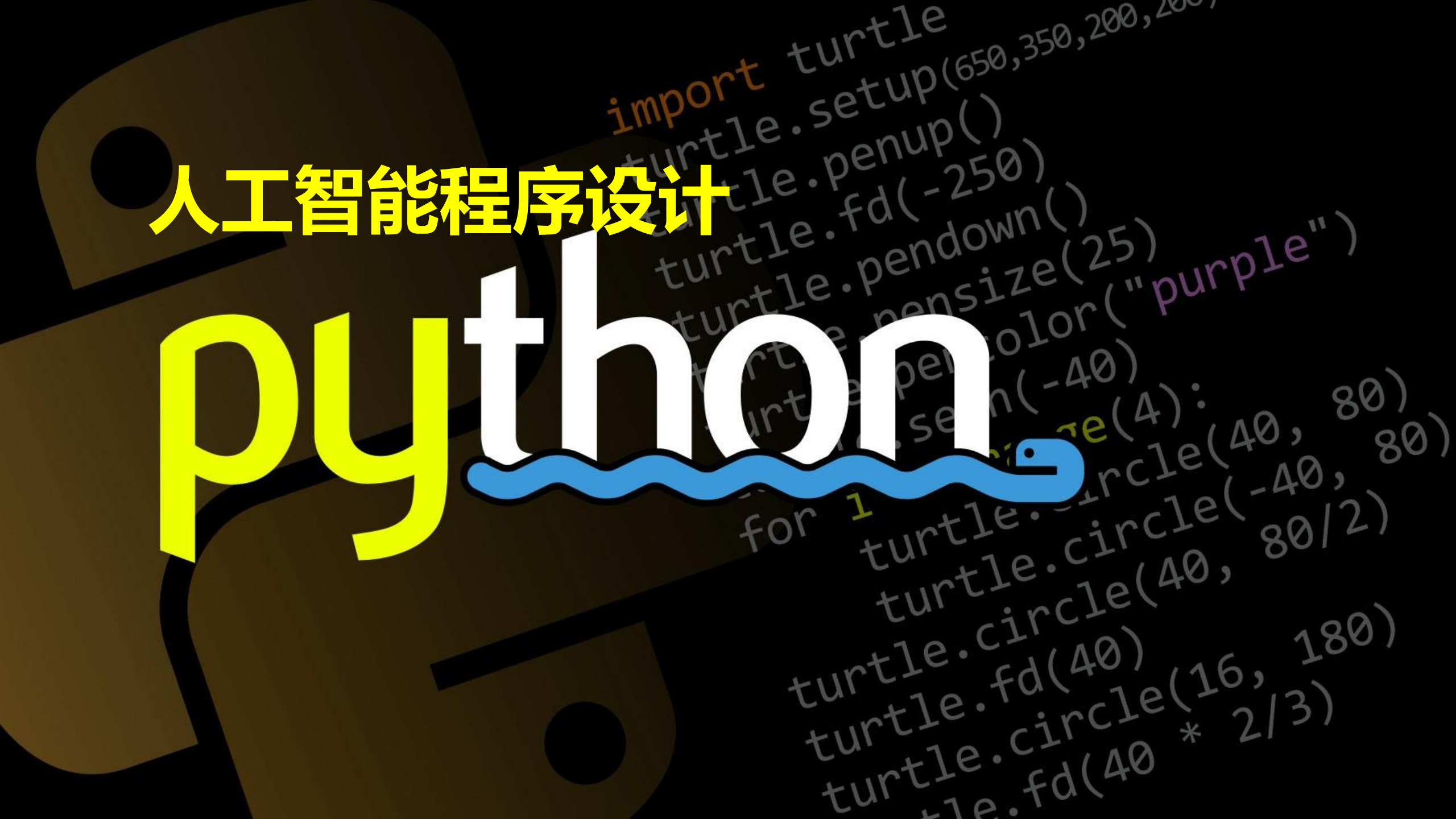
人工智能程序设计


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

16.4 大模型应用实践

北京石油化工学院 人工智能研究院

刘 强

学习内容

- 智能学习助手开发
- 企业知识管理系统
- 多模态内容创作平台
- 智能客服系统

16.4.1 智能学习助手

学习内容：

- 学习计划生成
- 学习进度跟踪
- 个性化学习建议

学习计划生成

根据学习目标和时间安排，自动生成个性化的学习计划。

```
def create_study_plan(subject, level, duration, chat_model):
    prompt = """
    请为以下学习需求制定详细的学习计划:

    学科: {}
    水平: {}
    学习时长: {}

    请提供:
    1. 学习阶段划分
    2. 每个阶段的重点内容
    3. 具体的时间安排
    4. 学习资源建议
    """.format(subject, level, duration)

    return chat_model.predict(prompt)
```

学习计划使用示例

该函数能够根据用户的具体情况生成量身定制的学习路径。

```
# 使用示例
plan = create_study_plan(
    "Python编程",
    "初学者",
    "3个月",
    chat_model
)
print("学习计划:", plan)
```

学习进度跟踪

跟踪学习进度，提供学习建议和调整方案。

```
def assess_progress(completed_topics, quiz_scores, study_time, chat_model):
    progress_data = {
        "已完成主题": completed_topics,
        "测验成绩": quiz_scores,
        "学习时间": study_time
    }

    prompt = """
    基于以下学习数据，评估学习进度并提供建议：
    {}
    请分析当前进度、需要加强的方面和学习方法调整建议。
    """.format(progress_data)

    return chat_model.predict(prompt)
```

16.4.2 企业知识管理系统

学习内容：

- 知识库构建
- 知识提取与结构化
- 智能问答服务

知识提取

自动处理企业文档，提取结构化的知识点。

```
def extract_knowledge(documents, chat_model):
    knowledge_base = []

    for doc in documents:
        prompt = """
        从以下文档中提取关键知识点：
        {}
        请提取：核心概念、重要流程、关键数据、注意事项
        """.format(doc[:2000])

        knowledge = chat_model.predict(prompt)
        knowledge_base.append({
            "document": doc["name"],
            "knowledge": knowledge
        })

    return knowledge_base
```

智能问答服务

基于企业知识库提供准确的问答服务。

```
def enterprise_qa(question, knowledge_base, chat_model):
    # 搜索相关知识
    relevant_knowledge = search_knowledge(question, knowledge_base)

    prompt = """
    基于企业知识库回答问题:

    相关知识: {}
    问题: {}

    请提供准确答案、相关政策和操作步骤。
    """.format(relevant_knowledge, question)

    return chat_model.predict(prompt)
```

16.4.3 多模态内容创作平台

学习内容：

- 内容策划
- 多格式内容生成
- 跨平台适配

内容策划

根据主题和受众，生成内容创作策略。

```
def plan_content(topic, audience, platform, chat_model):
    prompt = """
    为以下内容需求制定创作策略:

    主题: {}
    目标受众: {}
    发布平台: {}

    请提供: 内容角度、结构建议、关键信息、互动元素
    """.format(topic, audience, platform)

    return chat_model.predict(prompt)

# 使用示例
strategy = plan_content("人工智能在教育中的应用",
                        "教育工作者", "微信公众号", chat_model)
```

多格式内容生成

根据不同平台需求，生成适配的内容格式。

```
def generate_multi_format(content_brief, formats, chat_model):
    results = {}

    for format_type in formats:
        if format_type == "长文章":
            prompt = "将以下内容要点展开成详细文章：\n{}".format(content_brief)
        elif format_type == "社交媒体":
            prompt = "将以下内容改写成社交媒体文案：\n{}".format(content_brief)
        elif format_type == "演讲稿":
            prompt = "将以下内容整理成演讲稿格式：\n{}".format(content_brief)

        results[format_type] = chat_model.predict(prompt)

    return results
```

16.4.4 智能客服系统

学习内容：

- 客户意图分析
- 情感分析
- 智能回复生成

客户意图分析

分析客户咨询的真实意图，提供精准的服务响应。

```
def analyze_intent(customer_message, chat_model):
    prompt = """
    分析客户咨询的意图类型:

    客户消息: {}

    可能的意图类型:
    1. 产品咨询
    2. 技术支持
    3. 售后服务
    4. 账户问题
    5. 其他咨询

    请判断意图类型并说明理由。
    """.format(customer_message)

    return chat_model.predict(prompt)
```

情感分析

分析客户消息的情感倾向，调整服务策略。

```
def analyze_sentiment(customer_message, chat_model):
    prompt = """
    分析客户消息的情感倾向：

    消息：{}

    请判断情感类型（积极/中性/消极）并给出处理建议。
    """.format(customer_message)

    return chat_model.predict(prompt)
```

智能回复生成

综合意图、情感和知识库，生成个性化回复。

```
def generate_response(customer_query, intent, sentiment,
                      knowledge_base, chat_model):
    relevant_info = search_relevant_info(intent, knowledge_base)
    tone = "亲切耐心" if sentiment == "消极" else "专业友好"

    prompt = """
    为客户生成回复:
    客户问题: {}
    意图类型: {}
    回复语调: {}
    相关信息: {}
    """.format(customer_query, intent, tone, relevant_info)

    return chat_model.predict(prompt)
```

实践练习

练习 16.4.1：学习助手原型

基于16.4.1的学习计划生成和进度跟踪功能，开发一个简单的学习助手原型，支持个性化学习建议。

实践练习

练习 16.4.2：企业问答机器人

结合16.4.2的知识提取和问答功能，为一个小型企业构建内部知识问答机器人。

实践练习

练习 16.4.3：智能客服助手

实现16.4.4的意图识别和智能回复功能，创建一个能够处理常见客户咨询的智能助手。