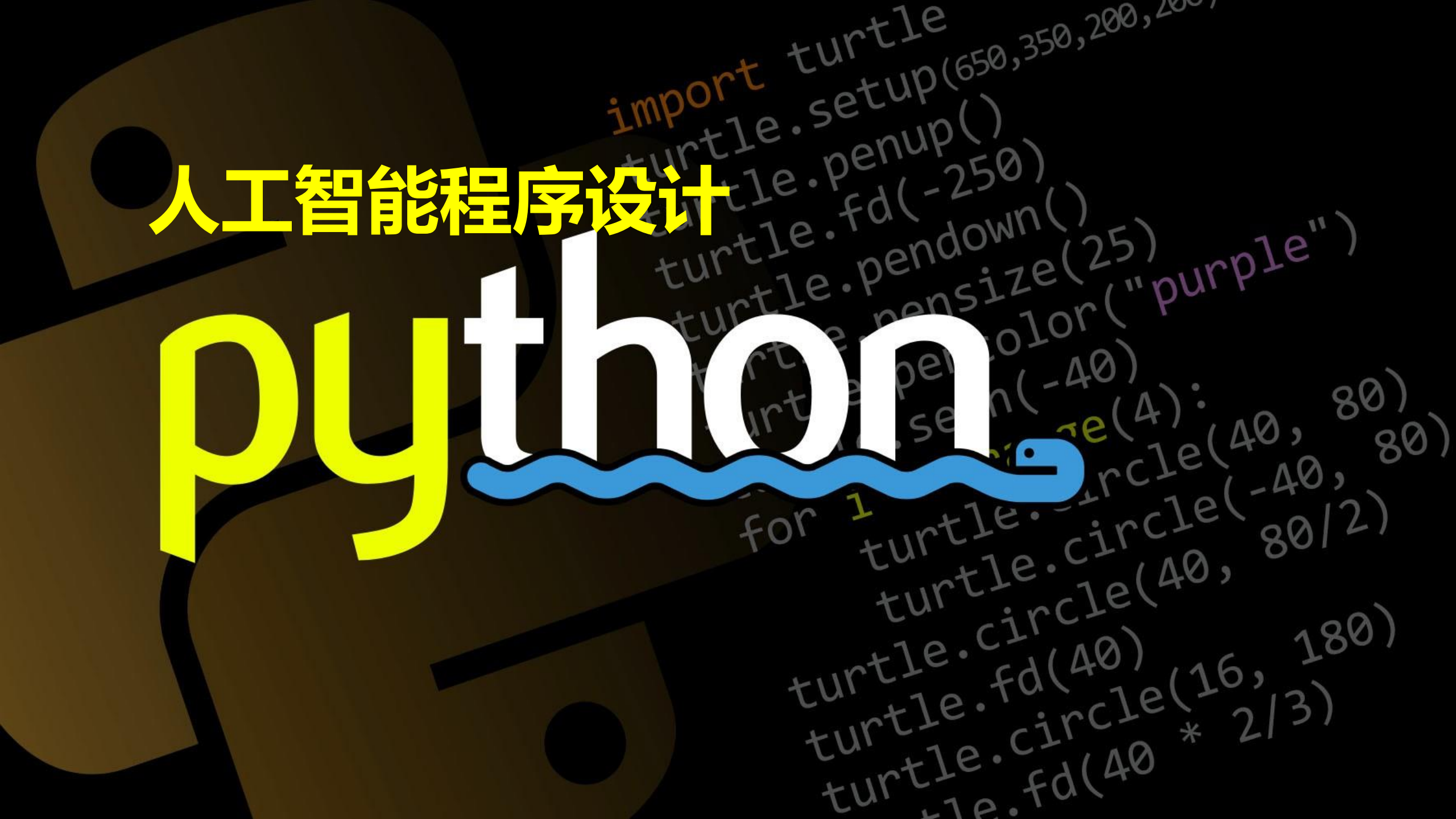
人工智能程序设计


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

17.4 智能体应用实践

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
- 企业知识管理智能体
- 智能客服团队系统
- 智能内容创作工作室

17.4.1 智能学习助手

学习内容：

- 学习路径规划
- 智能答疑系统
- 个性化学习指导

学习路径规划

学习助手根据学习者的基础和目标制定个性化的学习计划。

```
def create_learning_plan(topic, level, duration, llm):
    prompt = """
    为学习者制定{}的学习计划:
    当前水平: {}
    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. 推荐的学习资源
    4. 评估方式
    """.format(topic, level, duration)

    plan = llm.generate(prompt)
    return {"plan": plan, "topic": topic, "level": level}
```

学习计划使用示例

根据学习者的具体情况生成个性化的学习路径。

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

智能答疑系统

学习助手能够回答学习过程中的各种问题。

```
def answer_question(question, context, knowledge_base, llm):
    relevant_info = knowledge_base.search(question)

    prompt = """
    基于以下信息回答问题:
    问题: {}
    上下文: {}
    相关知识: {}

    请提供清晰、准确的答案。
    """.format(question, context, relevant_info)

    answer = llm.generate(prompt)
    return {"answer": answer, "confidence": 0.9}
```

17.4.2 企业知识管理智能体

学习内容：

- 知识收集与整理
- 智能知识问答
- 知识库管理

知识收集与整理

智能体自动从各种来源收集和整理知识。

```
def collect_knowledge(sources, categories, llm):
    collected_data = []

    for source in sources:
        content = extract_content(source)
        category = classify_content(content, categories, llm)

        structured_info = {
            "content": content,
            "category": category,
            "source": source,
            "timestamp": time.time()
        }
        collected_data.append(structured_info)

    return organized_knowledge(collected_data)
```

内容分类

自动从多个源头获取信息并进行分类整理。

```
def classify_content(content, categories, llm):
    prompt = """
    将以下内容分类到合适的类别：
    内容：{}...
    可选类别：{}

    返回最合适的类别。
    """.format(content[:200], categories)

    category = llm.generate(prompt)
    return category.strip()
```

智能知识问答

企业员工可以通过自然语言查询企业知识库。

```
def enterprise_qa(question, knowledge_base, llm):
    # 检索相关文档
    relevant_docs = knowledge_base.similarity_search(question, k=3)

    # 生成答案
    context = "\n".join([doc.content for doc in relevant_docs])
    prompt = """
    基于企业知识库回答问题:
    问题: {}
    相关文档: {}

    请提供准确、实用的答案。
    """.format(question, context)

    answer = llm.generate(prompt)
    return {"answer": answer, "sources": [doc.source for doc in relevant_docs]}
```

17.4.3 智能客服团队系统

学习内容：

- 客户意图识别
- 协作处理流程
- 质量保障机制

客户意图识别

准确识别客户的意图和需求。

```
def identify_intent(customer_message, llm):
    prompt = """
    分析客户消息的意图:
    客户消息: {}

    可能的意图类别:
    - 产品咨询
    - 技术支持
    - 订单查询
    - 投诉建议
    - 退换货

    返回最可能的意图类别。
    """.format(customer_message)

    intent = llm.generate(prompt)
    confidence = calculate_confidence(customer_message, intent)

    return {"intent": intent.strip(), "confidence": confidence}
```

路由到专业智能体

确保客户咨询被路由到最合适的专业智能体。

```
def route_to_specialist(intent, message):
    specialists = {
        "产品咨询": "product_agent",
        "技术支持": "tech_agent",
        "订单查询": "order_agent"
    }

    return specialists.get(intent, "general_agent")
```

协作处理流程

多个专业智能体协作处理复杂的客户问题。

```
def collaborative_service(customer_query, agents_team):
    # 初步处理
    intent = identify_intent(customer_query, agents_team.classifier)
    specialist = agents_team.get_specialist(intent)

    # 专业处理
    initial_response = specialist.handle(customer_query)

    # 质量检查
    if initial_response["needs_review"]:
        supervisor = agents_team.get_supervisor()
        final_response = supervisor.review_and_improve(initial_response)
    else:
        final_response = initial_response

    return {"response": final_response["content"],
            "handled_by": specialist.name}
```

17.4.4 智能内容创作工作室

学习内容：

- 内容策划与规划
- 多维度内容创作
- 跨平台适配

内容策划与规划

基于用户需求和市场趋势制定创作计划。

```
def plan_content_strategy(topic, target_audience, platform, llm):
    prompt = """
    为以下需求制定内容策略:
    主题: {}
    目标受众: {}
    发布平台: {}

    请提供:
    1. 内容角度和重点
    2. 内容结构规划
    3. 发布时间建议
    4. 预期效果评估
    """.format(topic, target_audience, platform)

    strategy = llm.generate(prompt)
    return {"strategy": strategy, "topic": topic}
```

多维度内容创作

智能体团队分工处理各种内容形式。

```python
def create_multimedia_content(topic, content_types, agents_team):
    results = {}

    for content_type in content_types:
        if content_type == "article":
            results["article"] = agents_team.writer.create_article(topic)
        elif content_type == "summary":
            results["summary"] = agents_team.summarizer.create_summary(topic)
        elif content_type == "social_post":
            results["social_post"] = agents_team.social_writer.create_post(topic)

    # 统一风格和质量检查
    final_content = agents_team.editor.unify_and_review(results)

    return final_content
```

跨平台内容适配

满足不同平台和受众的需求。

```
def generate_content_variations(base_content, platforms, llm):
    variations = {}

    for platform in platforms:
        prompt = """
        将以下内容改写为适合{}的版本:
        原内容: {}

        考虑{}的特点和用户习惯。
        """.format(platform, base_content, platform)

        variation = llm.generate(prompt)
        variations[platform] = variation

    return variations
```

实践练习

练习 17.4.1：学习助手开发

基于LangGraph开发一个智能学习助手，实现学习计划制定和智能答疑功能。

实践练习

练习 17.4.2：知识管理系统

使用多智能体架构构建企业知识管理系统，支持知识收集、整理和问答。

实践练习

练习 17.4.3：客服系统优化

设计一个多智能体客服系统，实现意图识别、专业分工和协作处理。