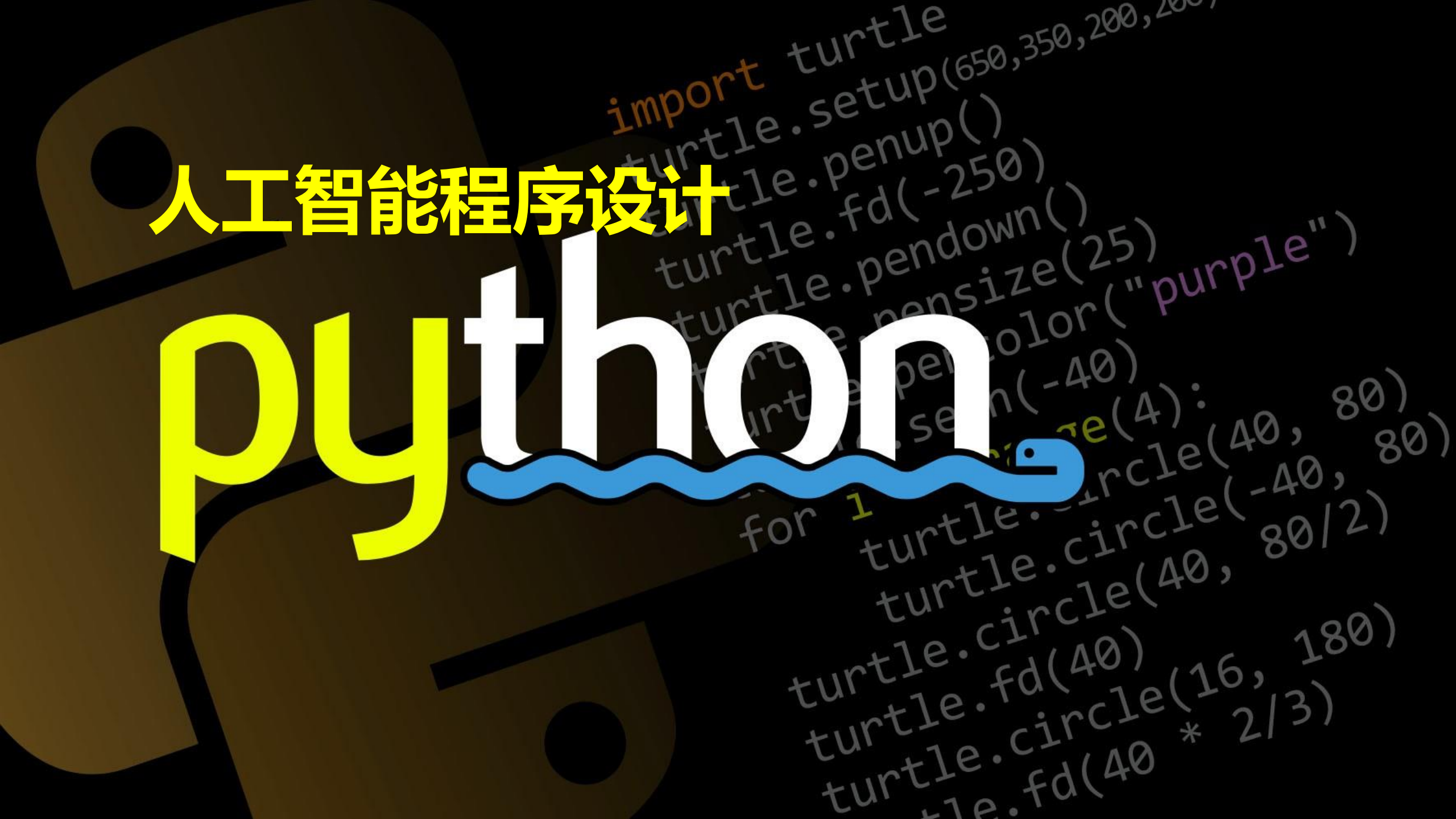
人工智能程序设计


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

10.4 数据交换格式：JSON

北京石油化工学院 人工智能研究院

刘 强

JSON 简介

JSON（JavaScript Object Notation）是一种轻量级的数据交换格式：

- 因简洁性和可读性被广泛应用于 Web API、配置文件和数据存储
- 在数据科学项目中，经常需要处理来自 API 的 JSON 数据
- 也常用于将分析结果保存为 JSON 格式

10.4.1 JSON基础语法

- **轻量级**：比 XML 更简洁，传输效率高
- **易读性**：人类可读的文本格式
- **跨平台**：支持几乎所有编程语言
- **结构化**：支持对象和数组的嵌套结构

JSON数据类型

JSON 支持字符串、数字、布尔值、null、数组和对象六种数据类型：

```
{
    "字符串": "Hello World",
    "数字": 42,
    "浮点数": 3.14,
    "布尔值": true,
    "空值": null,
    "数组": [1, 2, 3, "four"],
    "对象": {
        "嵌套键": "嵌套值",
        "数字": 100
    }
}
```

JSON语法规则

JSON 的语法规则简单明确：

- 数据以键值对形式存在
- 键必须是字符串，用双引号包围
- 值可以是字符串、数字、布尔值、null、对象或数组
- 字符串必须用双引号
- 对象用花括号 `{}` 包围
- 数组用方括号 `[]` 包围

10.4.2 Python JSON模块

Python 内置的 **json** 模块提供了 JSON 数据的处理功能。两个核心函数：

- **dumps**：将 Python 对象转换为 JSON 字符串
- **loads**：将 JSON 字符串转换为 Python 对象

```
import json

data = {
    "name": "张三",
    "age": 25,
    "city": "北京",
    "skills": ["Python", "JavaScript", "SQL"]
}
```

Python对象转JSON字符串

使用 **dumps** 函数将 Python 对象转换为 JSON 字符串：

```
import json

data = {
    "name": "张三",
    "age": 25,
    "city": "北京",
    "skills": ["Python", "JavaScript", "SQL"]
}

## ensure_ascii=False允许中文正常显示，indent=2设置缩进格式
json_string = json.dumps(data, ensure_ascii=False, indent=2)
print("Python对象转JSON: ")
print(json_string)
```

JSON字符串转Python对象

使用 **loads** 函数将 JSON 字符串转换为 Python 对象：

```
## json.loads函数将JSON字符串转换为Python对象
parsed_data = json.loads(json_string)
print("JSON转Python对象: ")
print(parsed_data)
print("类型: ", type(parsed_data))
```

文件操作：写入JSON文件

使用 **dump** 函数将 Python 对象直接写入 JSON 文件。与 **dumps** 不同，**dump** 直接操作文件对象：

```
employee_data = {
    "employees": [
        {"id": 1, "name": "张三", "department": "技术部", "salary": 8000},
        {"id": 2, "name": "李四", "department": "市场部", "salary": 7500},
        {"id": 3, "name": "王五", "department": "人事部", "salary": 6800}
    ],
    "company": "ABC科技有限公司",
    "total_employees": 3
}

with open('employees.json', 'w', encoding='utf-8') as f:
    json.dump(employee_data, f, ensure_ascii=False, indent=2)

print("数据已保存到employees.json文件")
```

文件操作：读取JSON文件

使用 **load** 函数从 JSON 文件读取数据并转换为 Python 对象：

```
with open('employees.json', 'r', encoding='utf-8') as f:
    loaded_data = json.load(f)

print("从文件读取的数据: ")
print("公司名称:", loaded_data['company'])
print("员工总数:", loaded_data['total_employees'])

for employee in loaded_data['employees']:
    print("员工:", employee['name'], "部门:", employee['department'])
```

JSON模块函数对比

函数	功能	操作对象
`dumps`	Python对象 → JSON字符串	字符串
`loads`	JSON字符串 → Python对象	字符串
`dump`	Python对象 → JSON文件	文件对象
`load`	JSON文件 → Python对象	文件对象

实践练习

练习 10.4.1：JSON数据处理

1. 创建一个包含商品信息的JSON文件，包含商品名、价格、分类、库存等字段
2. 读取JSON文件并转换为Pandas DataFrame
3. 对数据进行分析后，将结果保存为新的JSON文件

实践练习

练习 10.4.2：API数据模拟

模拟处理电商API返回的商品数据：

1. 创建包含商品信息的复杂JSON结构
2. 解析JSON数据并提取关键信息
3. 实现数据验证和错误处理机制

实践练习

练习 10.4.3：数据格式转换

1. 将CSV数据转换为JSON格式
2. 实现JSON数据的扁平化和嵌套化
3. 创建通用的数据格式转换工具函数

本节小结

- `JSON` 是轻量级数据交换格式
- Python 内置 `json` 模块处理 JSON 数据
- 核心函数：`dumps`/`loads`（字符串）、`dump`/`load`（文件）
- 常用参数：`ensure_ascii=False`、`indent`