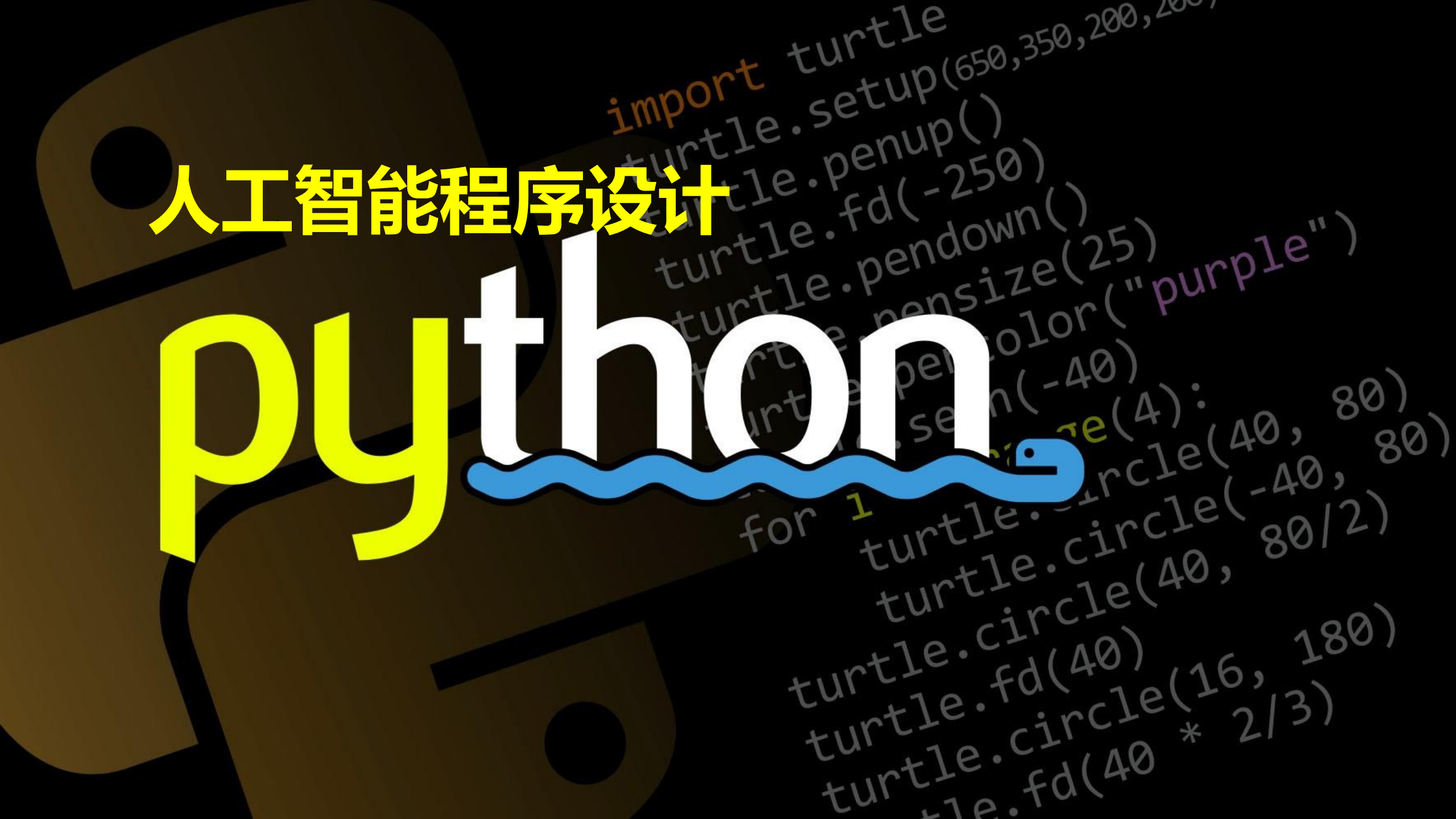
人工智能程序设计


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

10.2 数据处理工具：PANDAS

北京石油化工学院 人工智能研究院

刘 强

Pandas 简介

Pandas 是 Python 中最重要的数据分析和处理库：

- 建立在 NumPy 之上
- 为处理结构化数据提供强大而灵活的功能
- 是数据科学工作流的核心工具

10.2.1 Pandas简介与数据结构

- **强大的数据结构**：提供 `Series`（一维）和 `DataFrame`（二维）数据结构
- **灵活的数据处理**：支持数据清洗、转换、合并和重塑
- **多种数据格式支持**：可读取 `CSV`、`Excel`、`JSON`、`SQL` 等格式
- **高效的数据分析**：提供分组、聚合、统计等分析功能
- **时间序列处理**：专门的时间序列数据处理能力

环境搭建

安装 **Pandas** 库：

```
## 使用pip安装
pip install pandas

## 验证安装
python -c "import pandas as pd; print(pd.__version__)"
```

导入 **Pandas** 的标准方式，通常使用 **pd** 作为别名：

```
import pandas as pd
import numpy as np
```

核心数据结构：Series

Series 类似于带标签的数组，可以存储任何数据类型：

```
## 从列表创建Series
s1 = pd.Series([1, 2, 3, 4, 5])
print("Series:")
print(s1)

## 带索引的Series
s2 = pd.Series([85, 92, 78, 96], index=['张三', '李四', '王五', '赵六'])
print("带索引的Series:")
print(s2)
```

核心数据结构：DataFrame

DataFrame 是 Pandas 最重要的数据结构，类似于 Excel 表格。字典的键作为列名，值作为列数据：

```
## 从字典创建DataFrame
data = {
    '姓名': ['张三', '李四', '王五', '赵六'],
    '年龄': [25, 30, 35, 28],
    '城市': ['北京', '上海', '广州', '深圳'],
    '薪资': [8000, 12000, 10000, 9500]
}

df = pd.DataFrame(data)
print("DataFrame:")
print(df)
```

数据结构基本属性

了解数据结构的基本信息是数据分析的第一步：

```
## 查看基本信息
print("数据形状: ", df.shape)           # (4, 4)
print("列名: ", df.columns.tolist())    # ['姓名', '年龄', '城市', '薪资']
print("索引: ", df.index.tolist())      # [0, 1, 2, 3]
print("数据类型: ")
print(df.dtypes)

## 查看前几行数据
print("前2行数据: ")
print(df.head(2))

## 基本统计信息
print("数值列统计信息: ")
print(df.describe())
```

10.2.2 数据读取与基本操作

CSV 是最常用的数据交换格式，read_csv 函数会自动识别第一行作为列名：

```
## 创建示例CSV数据
sample_data = """姓名,年龄,城市,薪资
张三,25,北京,8000
李四,30,上海,12000
王五,35,广州,10000
赵六,28,深圳,9500"""

## 保存CSV文件
with open('employee_data.csv', 'w', encoding='utf-8') as f:
    f.write(sample_data)

## 读取CSV文件
df = pd.read_csv('employee_data.csv')
print("从CSV读取的数据: ")
print(df)
```

数据选择：选择列

列选择是数据分析的基础操作，使用方括号 [] 选择单列或多列：

```
## 选择单列
names = df['姓名']
print("姓名列: ")
print(names)

## 选择多列
subset = df[['姓名', '薪资']]
print("姓名和薪资: ")
print(subset)
```

数据选择：选择行

行选择使用 **loc**（按标签）和 **iloc**（按位置）两种方式：

```
## 使用loc按标签选择
first_row = df.loc[0]
print("第一行数据: ")
print(first_row)

## 使用iloc按位置选择
last_row = df.iloc[-1]
print("最后一行数据: ")
print(last_row)

## 选择多行
first_two = df.iloc[0:2]
print("前两行数据: ")
print(first_two)
```

条件筛选

条件筛选是数据分析中的常用操作，支持单条件和多条件筛选：

```
## 单条件筛选
high_salary = df[df['薪资'] > 9000]
print("薪资大于9000的员工：")
print(high_salary)

## 多条件筛选，使用&（与）和|（或）操作符
young_high_salary = df[(df['年龄'] < 30) & (df['薪资'] > 8000)]
print("年龄小于30且薪资大于8000的员工：")
print(young_high_salary)

## 使用isin方法进行多值匹配
beijing_shanghai = df[df['城市'].isin(['北京', '上海'])]
print("北京和上海的员工：")
print(beijing_shanghai)
```

数据修改：添加新列

数据修改包括添加新列和基于条件创建分类列：

```
## 基于现有列直接计算新列
df['年薪'] = df['薪资'] * 12
print("添加年薪列后：")
print(df)
```

Lambda函数简介

lambda函数 是 Python 中创建小型匿名函数的方式，语法为 **lambda 参数：表达式。**

下述代码段：

```
def salary_level(x):
    if x > 10000:
        return '高'
    elif x > 8000:
        return '中'
    else:
        return '低'
```

等价的 lambda 表达式：

```
lambda x: '高' if x > 10000 else '中' if x > 8000 else '低'
```

使用apply方法创建分类列

`apply` 方法会将函数应用到 `Series` 的每个元素上：

```
## 使用apply方法将lambda函数应用到'薪资'列的每个值
df['薪资等级'] = df['薪资'].apply(lambda x: '高' if x > 10000 else '中' if x > 8000 else '低')
print("添加薪资等级列后：")
print(df)
```

10.2.3 Ask AI：深入学习Pandas

掌握了 Pandas 基础后，可以向 AI 助手询问更多高级数据处理技术：

- "如何处理缺失值和数据清洗？"
- "如何使用 groupby 进行分组分析？"
- "如何进行数据合并和连接操作？"

实践练习

练习 10.2.1：用户行为数据处理

创建一个包含用户行为数据的CSV文件，包含用户ID、年龄、访问时长、点击次数、购买金额等字段：

1. 使用Pandas读取文件并查看基本信息
2. 筛选出购买金额大于500元的用户
3. 按年龄分组分析用户行为模式

实践练习

练习 10.2.2：数据清洗与转换

处理一个包含缺失值的销售数据：

1. 识别和处理缺失值
2. 将日期字符串转换为日期类型
3. 创建新的计算列（如销售额 = 数量 × 单价）

实践练习

练习 10.2.3：数据分组分析

分析员工数据：

1. 按部门和职位进行分组分析
2. 计算各组的平均薪资、最高薪资和人数
3. 找出薪资最高的前10%员工

本节小结

- `Pandas` 是 Python 数据分析的核心库
- 核心数据结构：`Series`（一维）和 `DataFrame`（二维）
- 数据读取：支持 CSV、Excel、JSON 等格式
- 数据操作：选择、筛选、修改、计算