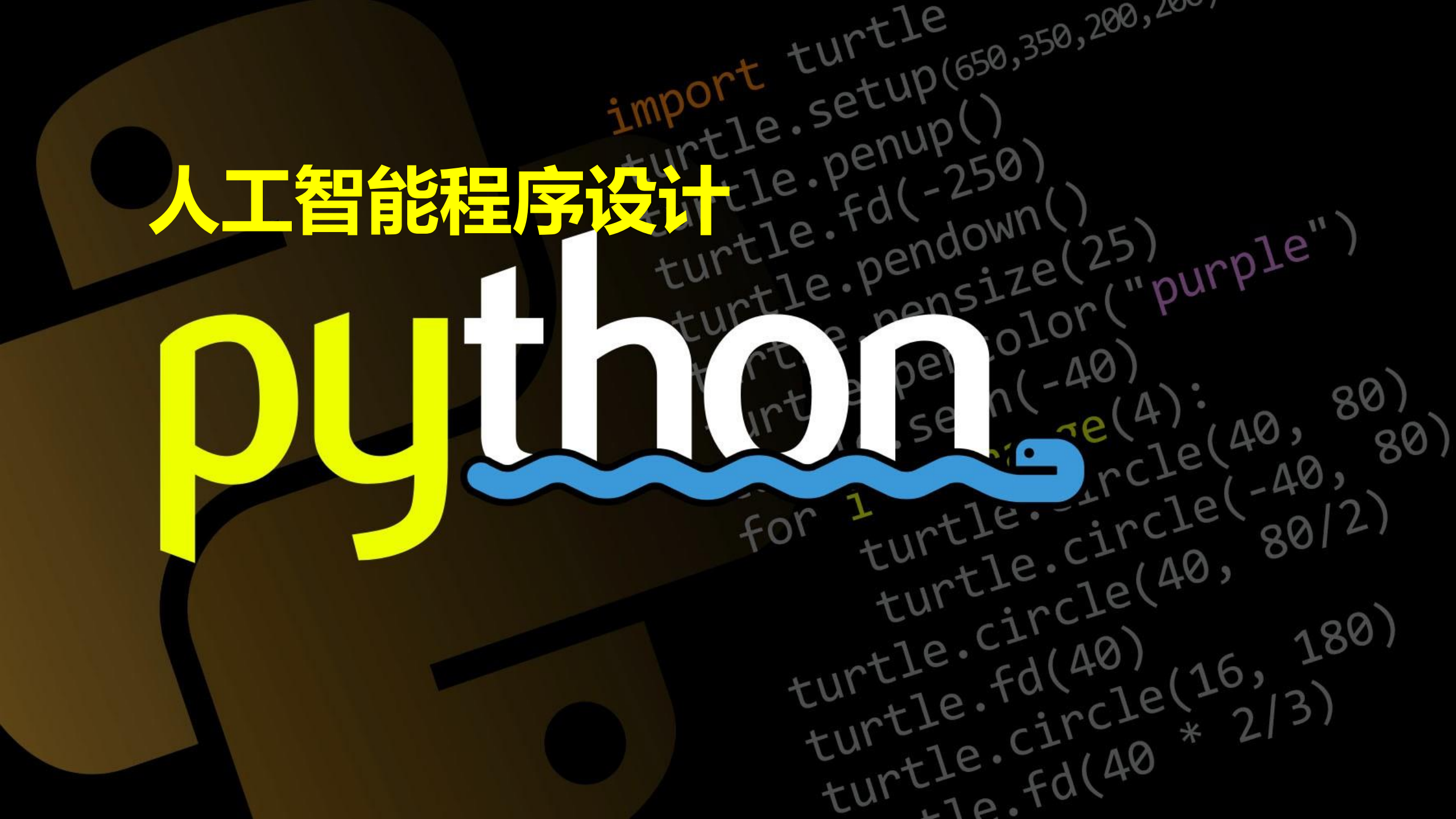
人工智能程序设计


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

4.5 函数

北京石油化工学院 人工智能研究院

刘 强

函数

从软件工程的角度看，**函数是模块化设计的基础。**

通过将复杂问题分解为多个独立的函数，每个函数完成一个特定的任务，可以降低程序的复杂度，提高代码的可读性和可维护性。

函数的独立性也使得测试更容易进行，团队成员可以并行开发不同的函数模块，大大提高开发效率。

掌握函数的使用，是从编写简单脚本到开发专业软件的重要转变。

Ask AI：什么是软件工程?

4.5.1 函数的定义与调用

函数是组织好的、可重复使用的代码块，用来实现单一或相关联功能。

函数能提高应用的模块性和代码的重复利用率。

Python提供了许多内建函数，同时也支持用户自定义函数。

函数定义的基本语法

在Python中，使用 **def**关键字来定义函数。函数定义包括函数名、参数列表、函数体和可选的返回值。函数名应该具有描述性，能够清楚地表达函数的功能。

```
def function_name(parameters):
    """文档字符串（可选）"""
    # 函数体
    statements
    return value  # 返回值（可选）
```

4.5.1 函数的定义与调用

语法要点：

- def：定义函数的关键字
- function_name：函数名，遵循Python变量命名规则
- parameters：参数列表，可以为空
- 冒号（:）：表示函数定义开始
- 缩进的代码块：函数体，包含具体的执行语句
- return：返回语句，可选，用于返回函数结果

```
def function_name(parameters):
    """文档字符串（可选）"""
    # 函数体
    statements
    return value  # 返回值（可选）
```

4.5.1 函数的定义与调用

简单函数示例

最简单的函数不需要参数，也不返回值，只是执行一些操作。

```
## 定义一个简单的问候函数
def greet():
    """打印问候语的函数"""
    print("Hello, World!")


## 调用函数
greet()  # 输出：Hello, World!
```

4.5.1 函数的定义与调用

简单函数示例

当函数需要处理不同的数据时，可以通过参数来传递信息。

```
## 定义带参数的函数
def greet_person(name):
    """向指定的人打招呼"""
    print(f"Hello, {name}!")


## 调用带参数的函数
greet_person("Alice")  # 输出：Hello, Alice!
```

4.5.1 函数的定义与调用

带返回值的函数：很多时候，我们需要函数处理数据后 return结果，而不仅仅是打印信息。使用return语句可以让函数返回计算结果，这样就可以在其他地方使用这个结果。

```
## 定义带返回值的函数
def add_numbers(a, b):
    """计算两个数的和"""
    return a + b
## 调用函数并使用返回值
result = add_numbers(5, 3)
print(f"5 + 3 = {result}")  # 输出：5 + 3 = 8
```

4.5.1 函数的定义与调用

带返回值的函数：很多时候，我们需要函数处理数据后 return结果，而不仅仅是打印信息。使用return语句可以让函数返回计算结果，这样就可以在其他地方使用这个结果。

```
## 函数可以返回多个值
def get_name_age():
    """返回姓名和年龄"""
    return "Alice", 25

name, age = get_name_age()
print(f"姓名：{name}，年龄：{age}") # 输出：姓名：Alice，年龄：25
```

4.5.2 函数参数的类型

函数参数的类型

1、位置参数

位置参数是最基本的参数传递方式，调用函数时参数的传递顺序必须与函数定义时的参数顺序一致。参数的值按照位置顺序依次传递给函数。

2、关键字参数

关键字参数允许通过参数名来指定参数值，这样可以不按照参数定义的顺序传递参数。使用关键字参数可以让函数调用更清晰，特别是当函数有很多参数时。

3、默认参数

默认参数允许为函数的某些参数设置默认值。如果调用函数时没有为这些参数提供值，函数将使用默认值。这样可以让函数更加灵活，减少必须传递的参数数量。

4.5.2 函数参数的类型

示例：矩形计算函数

下面的示例展示了位置参数的使用方式：

```
def calculate_area_perimeter(length, width):
    """计算矩形的面积和周长"""
    area = length * width
    perimeter = 2 * (length + width)
    return area, perimeter


## 按位置传递参数
area, perimeter = calculate_area_perimeter(5, 3)
print(f"矩形面积：{area}，周长：{perimeter}") # 输出：矩形面积：15，周长：16
```

4.5.2 函数参数的类型

下面的示例展示了关键字参数的使用方式：

```
def introduce_person(name, age, city):
    """介绍一个人的信息"""
    print(f"我叫{name}，今年{age}岁，来自{city}")

## 使用关键字参数，顺序可以任意
introduce_person(age=25, city="北京", name="张三")
introduce_person(name="李四", city="上海", age=30)

## 位置参数和关键字参数混合使用
introduce_person("王五", age=28, city="深圳")
```

4.5.2 函数参数的类型

下面的示例展示了默认参数的使用方式：

```
def greet_with_title(name, title="先生"):
    """带称谓的问候函数"""
    print(f"您好，{name}{title}！")


## 使用默认参数
greet_with_title("张三")        # 输出：您好，张三先生！
greet_with_title("李四", "女士")  # 输出：您好，李四女士！
```

4.5.3 递归函数

递归函数是指在函数内部调用自身的函数。递归函数必须包含两个要素：基础情况（停止递归的条件）和递归调用（函数调用自身）。递归调用会在满足基础情况时停止，否则会导致无限递归。

递归的思维过程：

1. 将大问题分解为相同类型的小问题：识别问题的递归结构，把规模为n的问题转化为规模更小的同类问题
2. 找到最简单情况的直接答案（基础情况）：确定递归终止的条件，这是可以直接给出答案的最简单情况
3. 确保每次递归都向基础情况靠近：每次递归调用时，问题规模必须缩小，最终能够达到基础情况

4.5.3 递归函数

示例：阶乘计算

下面是一个计算阶乘的递归函数示例。

阶乘的定义是：n! = n × (n-1) × … × 2 × 1，其中0! = 1。

递归分析：

- 大问题：计算 n!
- 小问题：n! = n × (n-1)!，即计算 (n-1)!
- 基础情况：0! = 1 和 1! = 1
- 递归靠近：每次调用时 n 减1，最终会达到基础情况

4.5.3 递归函数

示例：阶乘计算

```
def factorial(n):
    """计算n的阶乘"""
    if n == 0 or n == 1:
        return 1  # 基础情况
    else:
        return n * factorial(n - 1)  # 递归调用

## 测试阶乘函数
print(f"5! = {factorial(5)}")  # 输出：5! = 120
```

4.5.5 Ask AI：探索函数的高级特性用

当你想要深入了解Python函数的更多特性时，可以向AI助手提出以下问题：

- "什么是装饰器？如何使用装饰器来增强函数功能？"
- "如何编写可以接受任意数量参数的函数？"
- "什么是闭包？闭包在实际编程中有什么用途？"
- "如何使用 lambda函数简化代码？"
- "函数的递归调用有哪些注意事项和优化方法？"

实践练习

练习 4.5.1：时间处理函数

编写一组时间处理函数：将秒数转换为时分秒格式、给定一个小时数（0-23）判断这个时间点是否在工作时间内、计算两个时间点（小时:分:秒）的间隔，每个函数都要包含完整的文档字符串。

练习 4.5.2：文本处理函数

创建文本处理函数：统计字符串中单词数量、查找最长单词、反转句子中单词的顺序。

练习 4.5.3：数据分析函数

编写函数分析数字列表：计算中位数、查找众数、计算标准差，并返回分析报告