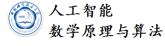


人工智能数学原理与算法

实验 3: 函数和模块

2025年3月16日

目录



定义和调用函数

局部变量和全局变量

默认值形参和关键字实参

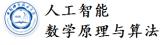
可变数量的实参

函数式编程

递归

创建和使用模块

实验 4: 函数和模块

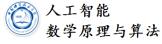


函数是一组语句,可以根据输入参数计算输出结果。把需要 多次运行的代码写成函数,可以实现代码的重复利用。以函 数作为程序的组成单位使程序更易理解和维护。

函数的定义包括函数头和函数体两部分。

- ▶ 函数头以关键字 def 开始,之后是空格和函数的名称。函数名称后面是一对圆括号,括号内可为空(表示无形参)或包含一些形参。形参表示函数的输入值。如果形参的数量超过一个,它们之间用逗号分隔。
- ▶ 函数体由一条或多条语句构成,完成函数的功能,相对函数头需要有四个空格的缩进。

调用函数

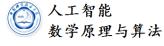


调用函数的语法是在函数的名称后面加上一对圆括号,括号内可为空(表示无实参)或包含一些实参。如果实参的数量超过一个,它们之间用逗号分隔。这些实参必须和函数的形参在数量上相同,并且在顺序上一一对应。

函数可以返回多个结果,这些结果之间用逗号分隔,构成一个元组。

程序 4.1

局部变量和全局变量

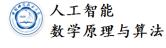


函数的形参和在函数体内定义的变量称为局部变量。局部变量只能在函数体内访问,在函数运行结束时即被销毁。

在函数体外定义的变量称为全局变量。全局变量在任何函数中都可以被访问,除非某个函数中定义了同名的局部变量。如果需要在函数体中修改某个全局变量,需要用 global 声明它。

程序 4.3

默认值形参和关键字实参

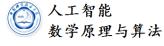


函数头可以给一个或多个形参赋予默认值,这些形参称为默认值形参。这些默认值形参的后面不能出现普通的形参。

在调用函数的语句中,可以在一个或多个实参的前面写上其对应的形参的名称。这些实参称为关键字实参。 此时实参的顺序不必和函数头中的形参的顺序保持一致。

如果一个函数的返回值有多个而且数量未知,可以把所有需要返回的结果存储在一个容器 (例如列表) 中,最后返回整个容器。程序 4.5

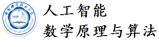
可变数量的实参



函数可以接受未知数量的位置实参和关键字实参。

程序 4.6 的第 1 行至第 4 行定义了一个函数 fun。形参 args 是一个元组,可接受未知数量的位置实参 (即普通实参)。形参kwargs 是一个字典,可接受未知数量的关键字实参。

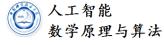
函数式编程



Python 语言支持函数式编程 (functional programming) 范式的基本方式是函数具有和其他类型 (如 int、float 等) 同样的性质:

- ▶被赋值给变量;
- ▶ 作为实参传给被调用函数的形参;程序 4.7
- ▶ 作为函数的返回值。程序 4.9

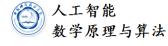
如果一个函数在定义以后只使用一次,并且函数体可以写成一个表达式,则可以使用 Lambda 函数语法将其定义成一个匿名函数: g = lambda 形参列表: 函数体表达式 程序 4.8



递归就是一个函数调用自己。当要求解的问题满足以下三个条件时, 递归是有效的解决方法。

- 1. 原问题可以分解为一个或多个结构类似但规模更小的子问题。
- 2. 当子问题的规模足够小时可以直接求解, 称为递归的终止条件; 否则可以继续对子问题递归求解。
- 3. 原问题的解可由子问题的解合并而成。

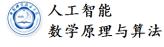
用递归方法解决问题的过程是基于对问题的分析提出递归公式。



阶乘的定义本身就是一个递归公式:

$$n! = \begin{cases} 1 & \text{if } n = 1 \\ n * (n-1)! & \text{if } n > 1 \end{cases}$$

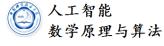
程序 4.11 的 gcd 函数求解两个正整数的最大公约数。递归函数都可以转换成与其等价的迭代形式,例如这个 gcd 函数对应的迭代形式是程序 4.1 中的 gcd 函数。



设 a 和 b 表示两个正整数。若 a > b,则易证 a 和 b 的公约数集合等于 a - b 和 b 的公约数集合,因此 a 和 b 的最大公约数等于 a - b 和 b 的最大公约数。若 a = b,则 a 和 b 的最大公约数等于 a。由此可总结出递归公式如下:

$$\gcd(a,b) = \begin{cases} a & \text{if } a = b \\ \gcd(a-b,b) & \text{if } a > b \\ \gcd(a,b-a) & \text{if } a < b \end{cases}$$

计算字符串反转

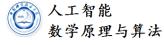


字符串反转就是将原字符串中的字符的先后次序反转,例如 "ABCDE" 反转以后得到 "EDCBA"。问题的分析过程如下。

- ► "ABCDE"="ABCD"+"E"
- ▶ "EDCBA"="E"+"DCBA"
- "DCBA"=reverse("ABCD")
- "E"=reverse("E")

递归的终止条件是:由单个字符构成的字符串的反转就是原字符串。由此可总结出递归公式如下:

$$reverse(s) = \begin{cases} s & \text{if } len(s) = 1\\ s[-1] + reverse(s[:-1]) & \text{if } len(s) > 1 \end{cases}$$



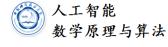
快速排序是一种著名的排序算法,以下用一个实例描述其求解过程。要排序的原始数据集是列表 [3, 6, 2, 9, 7, 3, 1, 8]。

- ▶ 以第一个元素 3 为基准对其进行调整,把比 3 小的元素移动到 3 的左边,把比 3 大的元素移动到 3 的右边。调整的结果为 [2, 1, 3, 3, 6, 9, 7, 8],可看成是三个列表的连接: [2, 1], [3, 3],和 [6, 9, 7, 8]。
- ▶ 排序完成的结果是 [1, 2, 3, 3, 6, 7, 8, 9], 也可看成是三个 列表的连接: [1, 2], [3, 3], 和 [6, 7, 8, 9]。
- ▶ 对 [2, 1] 进行排序可得 [1, 2], 这是原问题的一个子问题。 对 [6, 9, 7, 8] 进行排序可得 [6, 7, 8, 9], 这也是原问题的一 个子问题。

由此可总结出递归公式如下:

$$\operatorname{qsort}(s) = \begin{cases} s & \text{if } \operatorname{len}(s) \leq 1\\ \operatorname{qsort}(\{i \in s | i < s[0]\}) + \\ \{i \in s | i = s[0]\} + \operatorname{qsort}(\{i \in s | i > s[0]\}) & \text{if } \operatorname{len}(s) > 1 \end{cases}$$

创建和使用模块



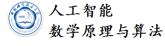
模块是一个包含了若干函数和语句的文件,文件名是模块的名称加上".py"后缀。一个模块实现了某类功能,是规模较大程序的组成单位,易于重复利用代码。

每个模块都有一个全局变量___name___。 模块的使用方式有两种。

- 1. 模块作为一个独立的程序运行,此时变量___name___的值为'___main___'。
- 2. 被其他程序导入以后调用其中的函数,此时变量 ___name___的值为模块的名称。

程序 4.16 列出了模块作为一个独立的程序在 IPython 中运行的示例。

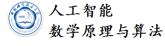
也可以在操作系统的命令行窗口中运行模块。此时需要将程序 4.16 的 In[2] 行的 run 改为 python。



模块除了可以作为一个独立的程序运行,也可以被其他程序导入以后调用其中的函数。如果使用模块的程序和模块文件在同一个目录下时,使用 import 语句导入模块即可使用。例如程序 4.17。

如果使用模块的程序和模块文件不在同一个目录下时,使用 import 语句导入模块会报错。此时需要将模块所在目录插入 到列表 sys.path 中 (程序 4.18), 然后可以导入模块。 本节以输出日历为例介绍创建和使用模块的方法。Python 标准库的 calendar 模块的 TextCalendar 类已提供了输出日历的功能。

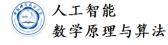
实验 4: 函数和模块



本实验的目的是掌握以下内容:定义和调用函数,创建和使用模块。

在 Blackboard 系统提交一个文本文件 (txt 后缀),文件中记录 每道题的源程序和运行结果。

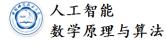
1. 二分查找



编写一个程序使用二分法查找给定的包含若干整数的列表 s 中是否存在给定的整数 k。使用二分查找的前提是列表已按照从小到大的顺序排序。为此,程序需要先判断 s 是否已经排好序。若未排好序,则需调用 qsort 函数进行排序并输出排序结果。

程序 4.21 已列出了部分代码,需要实现函数 is_sorted 和递归函数 binary_search。binary_search 在列表 s 的索引值属于闭区间 [low,high] 的元素中查找 k,若找到则返回 k 的索引值,否则返回-1。

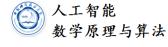
2. 有理数的四则运算



有理数的一般形式是 a/b, 其中 a 是整数, b 是正整数, 并且当 a 非 0 时 |a| 和 b 的最大公约数是 1。编写一个模块 rational.py 实现有理数的四则运算。

程序 4.22 已列出了部分代码,需要实现标注了"to be implemented"的函数。程序中用一个列表 [n, d] 表示有理数,其中 n 表示分子,d 表示分母。reduce 函数调用 gcd 函数进行约分。函数 add、sub、mul 和 div 分别进行加减乘除运算,运算的结果都需要约分,并且分母不出现负号。函数test_all_functions 使用已知答案的数据对这些运算进行测试。

2. 有理数的四则运算



函数 output 按照示例的格式输出有理数,例如 [-13,12]表示 的有理数的输出结果是字符串"-13/12"。用户在命令行输入 三个命名参数。"-op"表示运算符,可以是"add"(加法)、 "sub" (减法)、"mul" (乘法)或 "div" (除法)。"-x" 和 "-y" 表示进行计算的两个有理数。有理数以字符串的形式输入, 必须用圆括号括起,分子和分母之间用"/"分隔。例如有理 数 "-20/-3" 对应的输入形式是 (-20/-3), 用户输入的有理数 可以在分母出现负号。函数 get_rational 从表示有理数的字符 串中得到列表 [n, d], 例如从字符串 "(-20/-3)" 得到 [-20,-3]。