Continue

Python3 实例 在 Python 中，列表切片是一种非常强大的功能，它允许你从一个列表中提取一部分元素，形成一个新的子列表。切片操作使用方括号 [] 和冒号：来指定起始索引、结束索引和步长。 假设我们有一个列表 my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]，我们想要提取其中的一部分元素，可以使用切片操作。 my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] # 提取索引 2 到 5 的元素（不包括索引 5） sub_list = my_list[2:5] print(sub_list) 代码解析： my_list[2:5]：这是一个切片操作，2 是起始索引，5 是结束索引。切片操作会提取从索引 2 开始到索引 5 之前的元素（即不包括索引 5 的元素）。 sub_list：这是切片操作后得到的新列表。 输出结果： [2, 3, 4] Python3 实例 >>> li = ["a", "b", "mpilgrim", "z", "example"] >>> li ['a', 'b', 'mpilgrim', 'z', 'example'] >>> li[1] 'b' >>> li ['a', 'b', 'mpilgrim', 'z', 'example'] >>> li[-1] 'example' >>> li[-3] 'mpilgrim' >>> li ['a', 'b', 'mpilgrim', 'z', 'example'] >>> li[1:3] ['b', 'mpilgrim'] >>> li[1:-1] ['b', 'mpilgrim', 'z'] >>> li[0:3] ['a', 'b', 'mpilgrim'] Python3 实例 定义一个列表，并将它翻转。 例如： 翻转前 : list = [10, 11, 12, 13, 14, 15] 翻转后 : [15, 14, 13, 12, 11, 10] def Reverse(lst): return [ele for ele in reversed(lst)] lst = [10, 11, 12, 13, 14, 15] print(Reverse(lst)) 以上实例输出结果为： [15, 14, 13, 12, 11, 10] def Reverse(lst): lst.reverse() return lst lst = [10, 11, 12, 13, 14, 15] print(Reverse(lst)) 以上实例输出结果为： [15, 14, 13, 12, 11, 10] def Reverse(lst): new_lst = lst[::-1] return new_lst lst = [10, 11, 12, 13, 14, 15] print(Reverse(lst)) 以上实例输出结果为： [15, 14, 13, 12, 11, 10] Python3 实例 Python3 实例 在 Python 中，我们可以通过简单的代码来计算一个列表中所有元素的平均值。首先，我们需要将列表中的所有元素相加，然后除以列表的长度。 def calculate_average(numbers): return sum(numbers) / len(numbers) # 示例列表 numbers = [10, 20, 30, 40, 50] average = calculate_average(numbers) print("列表的平均值是:", average) 代码解析： calculate_average 函数接受一个列表 numbers 作为参数。 sum(numbers) 计算列表中所有元素的总和。 len(numbers) 计算列表的长度，即列表中元素的数量。 将总和除以列表长度得到平均值，并返回该值。 示例列表 numbers 包含五个元素。 调用 calculate_average 函数计算平均值，并打印结果。 输出结果： 列表的平均值是: 30.0 Python3 实例 Ollama 是一个开源的本地大语言模型运行框架，专为在本地机器上便捷部署和运行大型语言模型（LLM）而设计。 Ollama 支持多种操作系统，包括 macOS、Windows、Linux 以及通过 Docker 容器运行。 Ollama 提供对模型量化的支持，可以显著降低显存要求，使得在普通家用计算机上运行大型模型成为可能。 Ollama 适用于开发者、研究人员以及对数据隐私有较高要求的用户，它可以帮助用户在本地环境中快速部署和运行大型语言模型，同时提供灵活的定制化选项。 使用 Ollama，我们可以在在本地运行 Llama 3.3、DeepSeek-R1、Phi-4、Mistral、Gemma 2 和其他模型。 学习本教程前你需要了解 本教程适合有 Python 基础的开发者学习，如果不了解 Python 可以查阅 Python 3.x 基础教程。 理解 Docker 镜像和容器的区别，知道如何从 Docker Hub 拉取镜像并运行容器，docker 相关内容参见： Docker 教程。 熟悉命令行工具（如终端或命令提示符）的基本操作，例如文件和目录的创建、删除、移动，以及如何运行脚本和程序。 创建新的模型 我们可以使用 ollama create 命令从 Modelfile 创建模型： ollama create model -of ./Modelfile 相关链接 Ollama 官方地址： Github 开源地址： Ollama 官方文档： 序列是Python中最基本的数据结构。 序列中的每个元素都分配一个数字 - 它的位置，或索引，第一个索引是0，第二个索引是1，依此类推。 Python有6个序列的内置类型，但最常见的是列表和元组。 序列都可以进行的操作包括索引，切片，加，乘，检查成员。 此外，Python已经内置确定序列的长度以及确定最大和最小的元素的方法。 列表是最常用的Python数据类型，它可以作为一个方括号内的逗号分隔值出现。 列表的数据项不需要具有相同的类型 创建一个列表，只要把逗号分隔的不同的数据项使用方括号括起来即可。 如下所示： list1 = ['physics', 'chemistry', 1997, 2000] list2 = [1, 2, 3, 4, 5] list3 = ["a", "b", "c", "d"] 与字符串的索引一样，列表索引从0开始。 列表可以进行截取、组合等。 访问列表中的值 使用下标索引来访问列表中的值，同样你也可以使用方括号的形式截取字符，如下所示： list1 = ['physics', 'chemistry', 1997, 2000] list2 = [1, 2, 3, 4, 5, 6, 7] print "list1[0]: ", list1[0] print "list2[1:5]: ", list2[1:5] 以上实例输出结果： list1[0]: physics list2[1:5]: [2, 3, 4, 5] 更新列表 你可以对列表的数据项进行修改或更新，你也可以使用append()方法来添加列表项，如下所示： list = [] list.append('Google') list.append('Runoob') print list 注意：我们会在接下来的章节讨论append()方法的使用 以上实例输出结果： ['Google', 'Runoob'] 删除列表元素 可以使用 del 语句来删除列表的元素，如下实例： list1 = ['physics', 'chemistry', 1997, 2000] print list1 del list1[2] print "After deleting value at index 2 : " print list1 以上实例输出结果： ['physics', 'chemistry', 1997, 2000] After deleting value at index 2 : ['physics', 'chemistry', 2000] 注意：我们会在接下来的章节讨论remove()方法的使用 Python列表脚本操作符 列表对 + 和 * 的操作符与字符串相似。 + 号用于组合列表，* 号用于重复列表。 如下所示： Python 表达式结果 描述 len([1, 2, 3])3长度 [1, 2, 3] + [4, 5, 6][1, 2, 3, 4, 5, 6]组合 ['Hi!'] * 4['Hi!', 'Hi!', 'Hi!', 'Hi!']重复 3 in [1, 2, 3]True元素是否存在于列表中 for x in [1, 2, 3]: print x,1 2 3迭代 Python列表截取 Python 的列表截取实例如下： >>>L = ['Google', 'Runoob', 'Taobao'] >>> L[2] 'Taobao' >>> L[-2] 'Runoob' >>> L[1:] ['Runoob', 'Taobao'] >>> 描述： Python 表达式结果 描述 L[2]'Taobao'读取列表中第三个元素 L[-2]'Runoob'读取列表中倒数第二个元素 L[1:]['Runoob', 'Taobao']从第二个元素开始截取列表 Python列表函数&方法 Python包含以下函数: Python包含以下方法: C++ 标准库提供了丰富的功能，其中 是一个非常重要的容器类，用于存储元素集合，支持双向迭代器。 是 C++ 标准模板库（STL）中的一个序列容器，它允许在容器的任意位置快速插入和删除元素。与数组或向量（）不同， 不需要在创建时指定大小，并且可以在任何位置添加或删除元素，而不需要重新分配内存。 语法 以下是 容器的一些基本操作： 包含头文件：#include 声明列表：std::list mylist; ，其中 T 是存储在列表中的元素类型。 插入元素：mylist.push_back(value); 删除元素：mylist.pop_back(); 或 mylist.erase(iterator); 访问元素：mylist.front(); 和 mylist.back(); 遍历列表：使用迭代器 for (auto it = mylist.begin(); it != mylist.end(); ++it) 特点 双向迭代： 提供了双向迭代器，可以向前和向后遍历元素。 动态大小：与数组不同， 的大小可以动态变化，不需要预先分配固定大小的内存。 快速插入和删除：可以在列表的任何位置快速插入或删除元素，而不需要像向量那样移动大量元素。 声明与初始化 的声明和初始化与其他容器类似： #include #include int main() { std::list list1; // 空的list std::list list2(5); // 包含5个默认初始化元素的list std::list list3(5, 10); // 包含5个元素，每个元素为10 std::list list4 = {1, 2, 3, 4}; // 使用初始化列表 return 0; } 实例 下面是一个使用 的简单示例，包括创建列表、添加元素、遍历列表和输出结果。 #include #include int main() { // 创建一个整数类型的列表 std::list numbers; // 向列表中添加元素 numbers.push_back(10); numbers.push_back(20); numbers.push_back(30); // 访问并打印列表的第一个元素 std::cout