数据存储：数据结构与算法

数据结构基础

1. 数组与链表

1.1 数组

数组是一种基本的数据结构，用于存储相同类型的数据元素。这些元素在内存中是连续存储的，可以通过索引直接访问，提供快速的读取速度。数组的索引通常从0开始，每个元素可以通过其位置快速定位。

示例代码：数组的使用

```
# 创建一个数组
arr = [10, 20, 30, 40, 50]

# 访问数组中的元素
print(arr[0])  # 输出：10

# 修改数组中的元素
arr[0] = 100
print(arr)  # 输出：[100, 20, 30, 40, 50]

# 遍历数组
for i in arr:
    print(i)

# 添加元素到数组
arr.append(60)
print(arr)  # 输出：[100, 20, 30, 40, 50, 60]

# 删除数组中的元素
arr.remove(20)
print(arr)  # 输出：[100, 30, 40, 50, 60]
```

1.2 链表

链表是一种线性数据结构，其中的元素不是在内存中连续存储的，而是通过指针链接在一起。每个元素（称为节点）包含数据和指向下一个节点的指针。链表在插入和删除元素时比数组更高效，因为不需要移动大量数据。

示例代码：单链表的实现

```
class Node:
    """链表节点类"""
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    """单链表类"""
    def __init__(self):
        self.head = None

    def append(self, data):
        """在链表末尾添加元素"""
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            return
        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node

    def print_list(self):
        """打印链表"""
        cur_node = self.head
        while cur_node:
            print(cur_node.data)
            cur_node = cur_node.next

# 创建链表并添加元素
ll = LinkedList()
ll.append("A")
ll.append("B")
ll.append("C")

# 打印链表
ll.print_list()  # 输出：A B C
```