云南大学数学与统计学院

上机实践报告

|  |  |  |
| --- | --- | --- |
| **课程名称**：数据结构与算法实验 | **年级**：2015级 | **上机实践成绩**： |
| **指导教师**：陆正福 | **姓名**：刘鹏 |  |
| **上机实践名称**：数组序列实验 | **学号**：20151910042 | **上机实践日期**：2017-03-28 |
| **上机实践编号**：No.5 | **组号**： | **上机实践时间**：上午三、四节 |

# 一、实验目的

1. 熟悉与Python序列类型有关的数据结构与算法；

2. 熟悉主讲教材Chapter 5的代码片段。

# 二、实验内容

1. 数组序列有关的数据结构设计与算法设计

2. 调试主讲教材Chapter 5的5.5.3节的 Simple Cryptography 程序

# 三、实验平台

Windows 10 Enterprise 中文版；

Python 3.6.0；

Wing IDE Professional 6.0.2-1集成开发环境。

# 四、实验记录与实验结果分析

1.题

When creating a low-level array in a computer system, the precise size of that array must be explicitly declared in order for the system to properly allocate a consecutive piece of memory for its storage. For example, Figure 5.11 displays an array of 12 bytes that might be stored in memory locations 2146 through 2157.

＊在计算机系统中创建低级数组时，必须明确声明该数组的精确大小，以便系统正确分配连续的存储空间。例如，图5.11显示了可能存储在存储单元2146到2157中的12个字节的数组。（图略）

Because the system might dedicate neighboring memory locations to store other data, the capacity of an array cannot trivially be increased by expanding into subsequent cells. In the context of representing a Python tuple of str instance, this constraint is no problem. Instance of those classes are immutable, so the correct size for an underlying array can be fixed when the object is instantiated.

＊因为系统可能分配相邻的存储空间给其他的数据，因此数组的容量不能简单地通过往后延伸而扩展容量。在表达Python中的元组这个str类的实例时，这种约束条件成立。这些类的实例都是不能改变的，因此只有当一个对象被实例化的时候，底层数组的大小才能被确定下来。

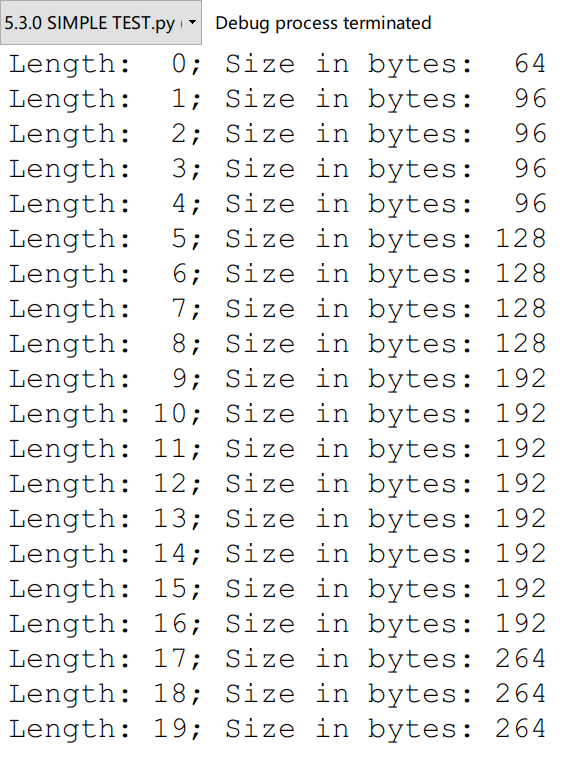
Python 804239504’s list class presents a more interesting abstraction. Although a list

程序代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # 5.3.0 SIMPLE TEST  **import** sys  data **=** **[]**  **for** k **in** range**(**20**):**  a **=** len**(**data**)**  b **=** sys**.**getsizeof**(**data**)**  **print(**'Length:{0:3d}; Size in bytes:{1:4d}'**.**format**(**a**,**b**))**  data**.**append**(None)** |

程序代码 1

实验结果：



运行结果 1

代码分析：

2.

程序代码：

# 五、实验体会

Translation

Chapter 5 Array-Based Sequence

＊第五章 基于数组的序列

5.1 Python’s Sequence Types

＊5.1节 Python的序列类型

In this chapter, we explore Python’s various “sequence” classes, namely the built-in **list**, **tuple**, and **str** classes. There is significant commonality between these classes, most notably: each supports indexing to access an individual element of a sequence, using a syntax such as seq[k], and each uses a low-level concept known as an ***array*** to represent the sequence. However, there are significant differences in the abstractions that these classes represent, and in the way that instances of these classes are represented internally by Python. Because these classed are used so widely in Python programs, and because they will become building blocks upon which we will develop more complex data structures, it is imperative that we establish a clear understanding of both the public behavior and inner workings of these classes.

＊在这一章中，我们将要探究一下Python语言中多变的序列类，也就是列表、元组还有字符串这三种内建类。这三种类之间有着很明显的共性，最明显的就是都支持通过下标来访问序列中的每一个元素，这可以用seq[k]这个语句来实现。并且这三个类都是通过一种叫做数组的底层概念来进行表达的。然而，在这些类的抽象表达中仍然有着明显的差异，而且正是通过这些差异Python才在语言内部能支持这些类的实例。我们之所以如此想要将这些类的公共行为与内部工作机制研究透彻，既是因为这些类在Python程序中被广泛使用，又是因为这些类能构建许多复杂的数据结构。

Public Behaviors

＊公共行为

A proper understanding of the outward semantics for a class is a necessity for a good programmer. While the basic usage of lists, strings, and tuples may seem straightforward, there are several important subtleties regarding the behaviors associated with these classes (such as what it means to make a copy of a sequence, or to take a slice of a sequence). Having a misunderstanding of a behavior can easily lead to inadvertent bugs in a program. Therefore, we establish an accurate mental model for each of these classes. These images will help when exploring more advanced usage, such as representing a multidimensional data set as a list of lists.

＊对一个好的程序设计师来说，透彻理解一个类的外在语义是必需的。

END

# 六、参考文献

[1] Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser, *Data Structures and Algorithms in Python*