

Python入门与基础

第三讲 线性数据结构 Part I

String List Tuple & LinkedList

林平之老师



扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: [ninechapter](#)

知乎专栏: <http://zhuanlan.zhihu.com/jiuzhang>

微博: <http://www.weibo.com/ninechapter>

官网: www.jiuzhang.com

版权声明

九章课程不提供视频，也严禁录制视频的侵权行为
否则将追求法律责任和经济赔偿
请一定不要缺课

- 字符串(String)及相关问题
- 列表(List)和元组(Tuple)
- 引用与链表(LinkedList)

字符串 String

```
1 str1 = 'Hello World!'  
2 str1 = "Hello JiuZhang!"  
3
```

- 可以使用单引号或者双引号
- 在一份代码中保持一致

字符串的常见操作

- 连接两个字符串
- 计算字符串的长度
- 遍历字符串中的每一个字符
- 替换, 截取, 查找
 - 能否修改字符串中的字母?

连接两个字符串



```
1 str1 = "Hello "  
2 str2 = "JiuZhang!"  
3  
4 print str1 + str2  
5
```

- + 字符串连接
- * 重复输出字符串
- % 格式字符串

提问: 如何给子串连接一个整数, 或者一个实数?

str 函数: 其他任何类型的变量转变成为字符串


```
1 str1 = "Hello JiuZhang!"  
2 print len(str1)  
3  
4
```

举例:输入密码的时候, 长度不够, 比如至少6位的密码, 只输入了5位, 不合格

len 函数:获取字符串的长度

访问字符串中的字符

```
1 str1 = "Hello JiuZhang!"  
2 print str1[0]  
3 print str1[6]  
4 print str1[5]  
5
```

通过name[index]的方式访问

- for循环遍历每一个字符
- 两种for循环方式, 取决于是否需要index
 - for c in str:
 - for idx in range(len(str)):

String To Integer

<http://www.lintcode.com/en/problem/string-to-integer/>

<http://www.jiuzhang.com/solution/string-to-integer/>

Rotate String

<http://www.lintcode.com/en/problem/rotate-string/>

<http://www.jiuzhang.com/solution/rotate-string/>

```
1 str1 = "Hello JiuZhang!"
2 str1[1] = 'c'
3
4 print str1
5
6
```

提问: 上述代码会出现什么错误?

List 列表

Python的基本数据结构之一：

```
1 a = ['1', '2', '3']
```

提问：list中的元素必须是同一中类型么？



提问: 为什么我们需要List?

列表的常见操作：

```
1 score = [91, 92, 68, 97, 96, 100, 59]
```

- 索引、切片
- 加、乘
- 元素检测
- 获取list长度
- 求list的最大, 最小值

List 列表 元素的访问与修改

- 访问第*i*个学生的分数
- 修改第*i*个学生的分数
- 查询是否含有100分(满分)的同学
- 查询班级中的最低分、最高分

- 添加元素
 - append操作
 - extend操作
- 删除元素
 - del 操作
 - pop 操作

Remove Element

<http://www.lintcode.com/en/problem/remove-element/>

<http://www.jiuzhang.com/solution/remove-element/>

Tuple 元祖

提问：为什么我们需要tuple？

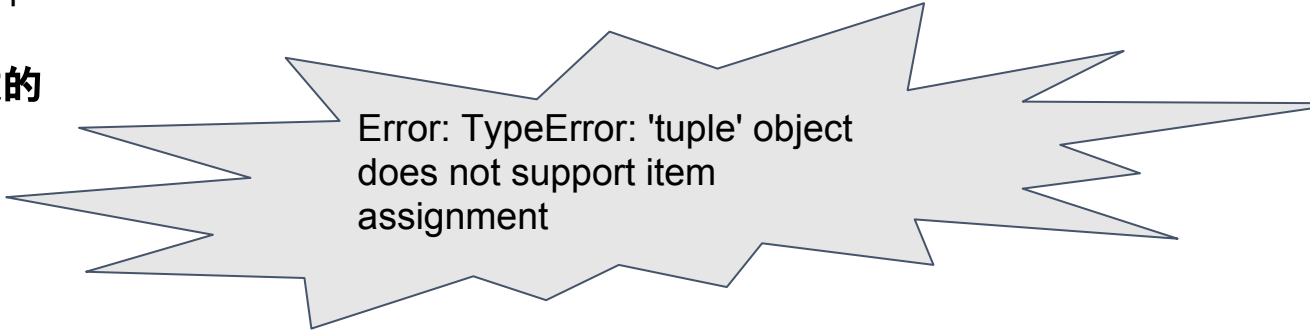
```
4  tup1 = ('Hello', 'Jiuzhang', 5)
5  tup2 = (1, 2, 3 )
6  tup3 = "a", "b", "c"
7
8  # Create empty tuple
9  tup4 = ()
10
11 tup5 = (10)
12
```

提问: 上述tuple的构建中, 哪些是错误的?

Tuple 元组的访问与修改

- 访问第i个元素
- 提取一段sub tuple
- 获取tuple中的元素个数
- 获取tuple中的最大值, 最小值
- 判断一个元素是否在tuple中
- **tuple中的元素是无法修改的**
 - **tup2[2] = 4 出错**

```
1 tup1 = ('Hello', 'Jiuzhang', 5)
2 tup2 = (1, 2, 3)
3 tup3 = "a", "b", "c"
4
5 # Create empty tuple
6 tup4 = ()
7
8 tup2[2] = 4
9
10
```



Error: TypeError: 'tuple' object
does not support item
assignment

- 连接操作生成了新的tuple
 - 因为tuple不支持修改, 所以只能生成新的
- 使用del语法删除已经存在的tuple
 - `del tup2`

表达式	结果	说明
<code>len((1,2,3))</code>	3	获取tuple的元素个数
<code>4 in (4,5,6)</code>	True	判断元素是否在tuple中出现
<code>for x in (4,5,6): print x</code>	4 5 6	遍历tuple
<code>(1,2,3)+(4,5,6)</code>	(1,2,3,4,5,6)	连接生成新的tuple

- tuple([])
 - tuple([1,2,3]) => (1,2,3)
- list()
 - list((1,2,3)) => [1,2,3]

LinkedList 链表

- 什么是reference
- 链表及其常用操作



对象去哪儿了？

```
node = ListNode(10)
```

node不是对象本身，是对象的引用

对象存储在堆空间(heap space)上(画图)

- 堆空间 heap space
- 栈空间 stack space (演示stack overflow)
- 注意区别于数据结构的heap & stack

什么是引用(Reference)？

引用好比遥控器，对象好比电视机

引用的赋值 - very important!

```
node1 = ListNode(10)
```

```
node2 = ListNode(20)
```

- 这里有两个对象
- node1和node2是这两个对象的引用

提问 : `node1 = node2` // What happened here?

两个对象:

node1 这个
对象,value=10

node2 这个
对象
value=20

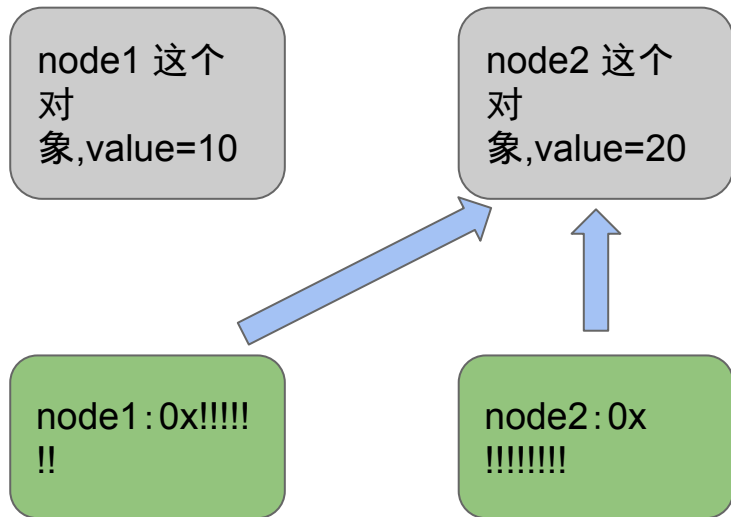
两个引用:

node1:0x??
???

node2:0x
!!!!!!!

当 $\text{node1} = \text{node2}$ 之后:

两个对象:



两个引用:

提问:如果把node1指向的对象的內容修改了,那么node2看到的对象是否也会发生变化呢?

引用的好处与用处：

- 引用也存的是数据，存的是指向这个对象的地址
- 使得数据更加的整齐
 - 画图
- 需要专递引用去修改数据

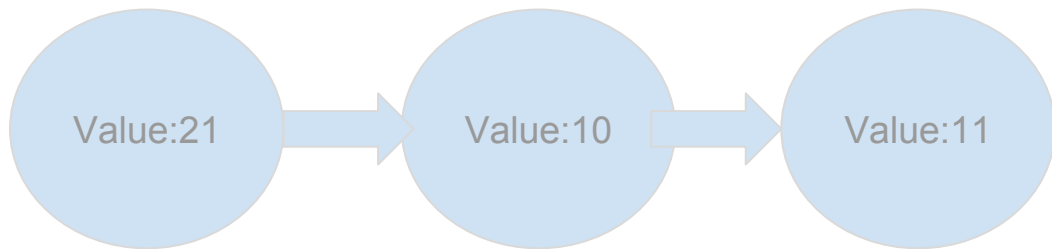
什么是数据结构(data structure)

- 数据
 - 存储数据的功能
- 结构
 - 如何组织排列存储的数据
- 操作
 - 如何查询, 添加, 删除维护存在的数据



什么是链表(linked list)

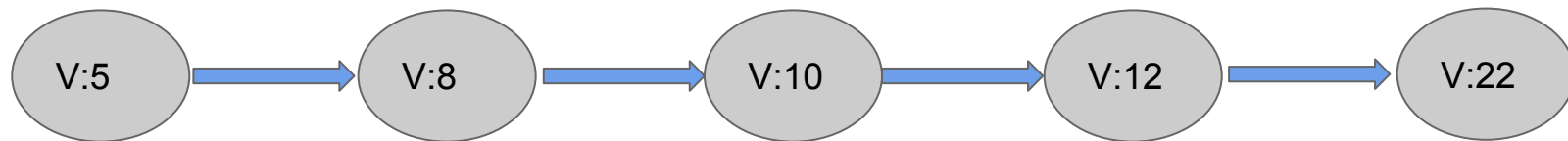
- 由节点构成的列表
- 线性的数据结构



```
class ListNode:
```

```
    def __init__(self, value):  
        self.val = value  
        self.next = None
```

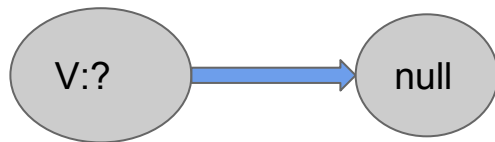
Linkedlist: 5->8->10->12->22->null



Coding: 基于ListNode实现一个Linked List

不得不提的dummy Node: 伟大的哨兵节点:

- dummyNode-> null



作用(前驱节点的重要性):

- 使得每一个元素都有前驱节点

链表的操作 (Coding演示)

遍历(traverse)

插入(insert)

查找(find)

删除(delete)

LinkedList Class的接口：

- 读取操作 `get(location)` // 获取location位置上的node的value
- 查找操作 `contains(val)` // 判断链表中是否含有val值的node
- 插入操作 `add(location, val)` // 在location的位置上插入一个值为val的node
- 删除操作 `remove(location)` // 删除location位置上的元素

Reverse Linked List

<http://www.lintcode.com/en/problem/reverse-linked-list/>

<http://www.jiuzhang.com/solutions/reverse-linked-list/>

Remove Nth Node From End of List

<http://www.lintcode.com/en/problem/remove-nth-node-from-end-of-list/>

<http://www.jiuzhang.com/solution/remove-nth-node-from-end-of-list/>



扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: [ninechapter](#)

微博: <http://www.weibo.com/ninechapter>

官网: www.jiuzhang.com



谢谢大家