**5.数据结构**

同学翻译：

已经学了int/float/str. 前两个是标量（scalar）str是有结构的，通过索引来找substr

介绍了元组（tuple）列表和字典。

元组和字符串拥有同样的通用化表达。

字典和列表更六，他们是可变的

元组是有序的，而且它的元素不一定是字符，可以是任何东西，且不需要相同。

元组的语法是用小括号括起来，用逗号来间隔。

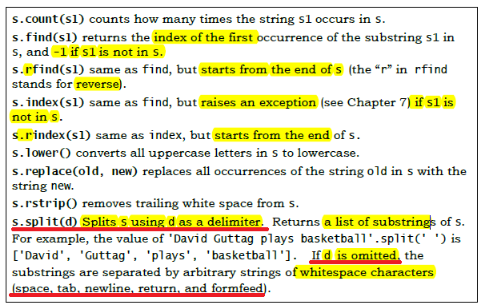
老师讲解：

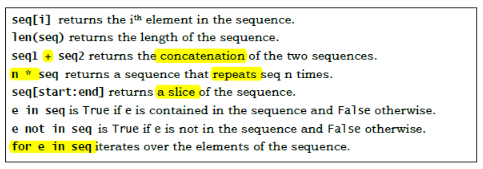
元组

1. 元组有序，可以用索引查找、元素可以不同、小括号和逗号**（一个元素的元组（10，）不然只认为是整数）**
2. 元组可以索引，分片（从哪里开始到哪里结束，这里是访问到结束号码减一，即从开始之处，找这么多个元素），连在一起（相加）
3. 一定会考分片str[start,end]从str中找start至end-1
4. 返回可以返回一个元组

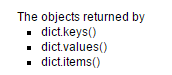
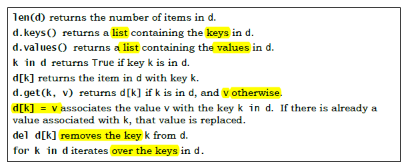
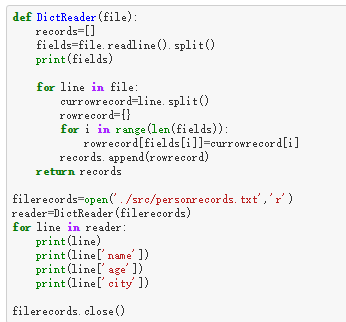
列表一定会考

1. 列表是可变的有序的可以去**索引**他,**分片**，元素可以是任意类型
2. 语法定义是中括号和逗号[10]是可以的，空的话是[]
3. 分片一般放在列表上面一定会放在程序里考，对列表分片后也可以索引的[1,2,3,4][1:3][1]=3
4. Univs == Univs1 判断的是两个列表的值是否相等，并不代表是同一个列表
5. 事实上，对每个内存对象有一个id是唯一的。两个列表值相等，不代表两个列表的id相等。
6. 可变类型：可以改变首地址后面内容，但是首地址仍然不变。所以可能访问的是同一个头，但是内容不一样了
7. 列表名.append()列表加一个元素，他和+是不一样的。Extend、Append是对原来的列表进行操作而+是两个加起来生成一个新的列表。
8. 避免混淆：for e1 in L1[:]:分片也是产生新的内存，不是在本地改 newL1=List(L1)生成新的列表 copy
9. 序列的操作 分片，重复，长度，遍历，寻找在不在（in/not in）
10. yao.split(s)用s将yao字符串分成子串一定会考(分割) 纯文本文件

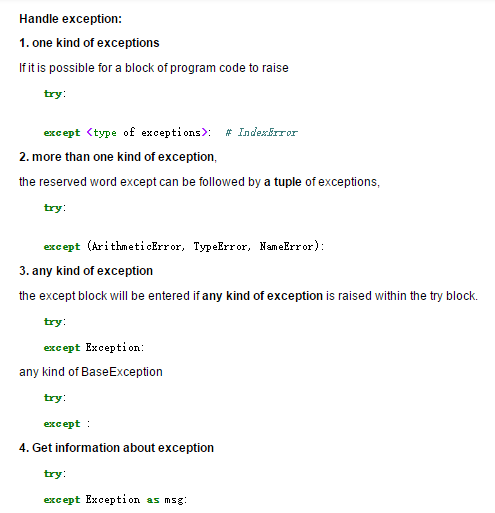
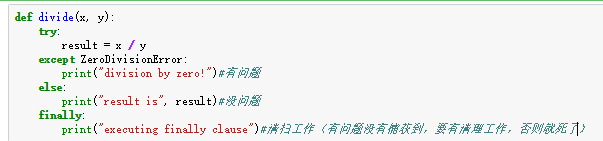




**字典**一定会考

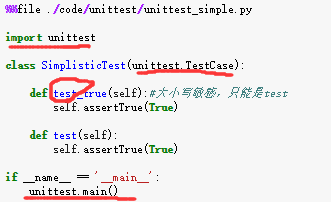
1. 用大括号括起，内容是许多个键值对key:value，用逗号隔开
2. 用key来索引，key一定不可以是可变类型（一定不能是列表）
3. 遍历字典得到的是key而不是键值对，要想返回键值对for (key,value) in yao.items():就可
4. 字典的可变性：同样的对象可以用yao[key]=value对他进行操作增加
5. 字典是无序的，不能用index来访问

**7错误处理机制**

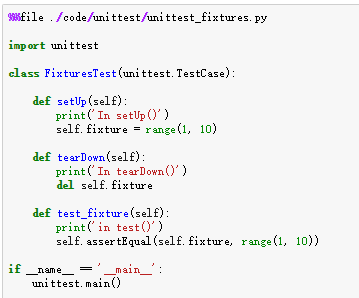
1. 错误的内部处理机制（看懂错误，知道意思）
2. as msg表示可以给出异常的信息，如果不加，仅仅捕获不解释try:except:
3. 抛出异常不考
4. 清理工作finally：（异常处理机制）必考

有问题，没问题，有问题没有预想到

**devtool单元测试**

1. 需要先引入，再使用

这个在交互环境里不能做

1. 浮点数的几乎相等self.assertNotAlmostEqual(1.11, 3.3-2.0, places=1)
2. 完整测试单元框架（注意setUp和tearDown的大小写）