序列,可迭代对象,迭代器,生成器之间的区别与联系

序列:

常见序列类型包括字符串(普通字符串和unicode字符串),列表和元组.所谓序列,即成员有序排列,可通过下标访问. Python包含 6 种内建的序列，包括列表、元组、字符串、Unicode字符串、buffer对象和xrange对象。

序列通用操作有:

1. 索引
2. 切片
   1. obj[start\_index:end\_index], 切片中冒号左边索引对应的元素晚于右边索引对应元素出现时,返回结果是[]
   2. [start\_index:end\_index:step] (step>0)
   3. [start\_index:end\_index:step] (step<0), 此时start\_index对应的元素要晚于end\_index对应的元素出现,否则返回[]
3. 相加,乘法
   1. 两种相同的序列才可以进行加法操作,即list1+list2,str1+str2
   2. str1\*3 ,list1\*3
4. 判断成员资格

If x in str1, if x in list1:

1. 最大值max(),最小值min(),求长度len()

参考1: <https://www.cnblogs.com/diaosir/p/6575891.html>

参考2: <https://blog.csdn.net/hedan2013/article/details/55000018>

可迭代对象:

1. 精确定义: 只要一个对象定义了可以返回一个迭代器的\_\_iter\_\_()方法，或者定义了可以支持下标索引的\_\_getitem\_\_()方法 ，那么它就是一个可迭代对象。
2. 模糊定义: 可以用for..in 循环遍历的对象就是可迭代对象,比如, list,dict,set,str，generator（生成器）,iterator（迭代器）全部都是可迭代对象（英语叫Iterable）

迭代器:

精确定义:在python2中必须实现了next()方法的类对象,在python3中实现了\_\_iter\_\_和\_\_next\_\_()方法的对象

那么什么是迭代器呢？它是一个带状态的对象，他能在你调用next()方法的时候返回容器中的下一个值，任何实现了\_\_iter\_\_和\_\_next\_\_()（python2中实现next()）方法的对象都是迭代器，\_\_iter\_\_返回迭代器自身，\_\_next\_\_返回容器中的下一个值，如果容器中没有更多元素了，则抛出StopIteration异常，至于它们到底是如何实现的这并不重要。

凡是可以返回一个迭代器的对象都可称之为可迭代对象

迭代器有一种具体的迭代器类型，比如list\_iterator，set\_iterator。可迭代对象实现了\_\_iter\_\_方法，该方法返回一个迭代器对象。

生成器:

生成器算得上是Python语言中最吸引人的特性之一，生成器其实是一种特殊的迭代器，不过这种迭代器更加优雅。它不需要再像上面的类一样写\_\_iter\_\_()和\_\_next\_\_()方法了，只需要一个yiled关键字。 生成器一定是迭代器（反之不成立），但迭代器不一定是生成器

生成器的实现方式:

1. 生成器表达式,返回一个生成器对象
2. 带yield的函数为生成器函数

参考1: <http://python.jobbole.com/87805/>

参考2: <https://www.cnblogs.com/wswang/p/6047994.html>

参考3: <https://blog.csdn.net/u014745194/article/details/70176117>

参考4: <https://www.zhihu.com/question/20829330>

参考5: <http://python.jobbole.com/85240/>

Python 2 和 Python 3 有哪些主要区别？

参考: <https://www.zhihu.com/question/19698598>

1.Python是如何进行内存管理的？

答:从三个方面来说,一对象的引用计数机制,二垃圾回收机制,三内存池机制

一、对象的引用计数机制

Python内部使用引用计数，来保持追踪内存中的对象，所有对象都有引用计数。

引用计数增加的情况：

1，一个对象分配一个新名称

2，将其放入一个容器中（如列表、元组或字典）

引用计数减少的情况：

1，使用del语句对对象别名显示的销毁

2，引用超出作用域或被重新赋值

sys.getrefcount( )函数可以获得对象的当前引用计数

多数情况下，引用计数比你猜测得要大得多。对于不可变数据（如数字和字符串），解释器会在程序的不同部分共享内存，以便节约内存。

二、垃圾回收

1，当一个对象的引用计数归零时，它将被垃圾收集机制处理掉。

2，当两个对象a和b相互引用时，del语句可以减少a和b的引用计数，并销毁用于引用底层对象的名称。然而由于每个对象都包含一个对其他对象的应用，因此引用计数不会归零，对象也不会销毁。（从而导致内存泄露）。为解决这一问题，解释器会定期执行一个循环检测器，搜索不可访问对象的循环并删除它们。

三、内存池机制

Python提供了对内存的垃圾收集机制，但是它将不用的内存放到内存池而不是返回给操作系统。

1，Pymalloc机制。为了加速Python的执行效率，Python引入了一个内存池机制，用于管理对小块内存的申请和释放。

2，Python中所有小于256个字节的对象都使用pymalloc实现的分配器，而大的对象则使用系统的malloc。

3，对于Python对象，如整数，浮点数和List，都有其独立的私有内存池，对象间不共享他们的内存池。也就是说如果你分配又释放了大量的整数，用于缓存这些整数的内存就不能再分配给浮点数。

4.请写出一段Python代码实现删除一个list里面的重复元素

1,使用set函数，set(list)

2，使用字典函数，

>>>a=[1,2,4,2,4,5,6,5,7,8,9,0]

>>> b={}

>>>b=b.fromkeys(a)

>>>c=list(b.keys())

>>> c

3.编程用sort进行排序，然后从最后一个元素开始判断

if \_\_name\_\_ == '\_\_main\_\_':

a = [1,3,5,2,2,3,5,4]

a.sort()

last = a[-1]

for i in range(len(a)-2,-1,-1):

if last == a[i]:

del a[i]

else:

last = a[i]

print(a)

'''

输出结果

[1, 2, 3, 4, 5]

[Finished in 0.2s]

'''

6.Python里面如何拷贝一个对象？（赋值，浅拷贝，深拷贝的区别）

赋值（=），就是创建了对象的一个新的引用，修改其中任意一个变量都会影响到另一个。

浅拷贝：创建一个新的对象，但它包含的是对原始对象中包含项的引用（如果用引用的方式修改其中一个对象，另外一个也会修改改变）{1,完全切片方法；2，工厂函数，如list()；3，copy模块的copy()函数}

深拷贝：创建一个新的对象，并且递归的复制它所包含的对象（修改其中一个，另外一个不会改变）{copy模块的deep.deepcopy()函数}

9.介绍一下Python下range()函数的用法？

10.如何用Python来进行查询和替换一个文本字符串？

答：可以使用re模块中的sub()函数或者subn()函数来进行查询和替换，

格式：sub(replacement, string[,count=0])（replacement是被替换成的文本，string是需要被替换的文本，count是一个可选参数，指最大被替换的数量）

>>> import re

>>>p=re.compile(‘blue|white|red’)

>>>print(p.sub(‘colour’,'blue socks and red shoes’))

colour socks and colourshoes

>>>print(p.sub(‘colour’,'blue socks and red shoes’,count=1))

colour socks and redshoes

subn()方法执行的效果跟sub()一样，不过它会返回一个二维数组，包括替换后的新的字符串和总共替换的数量

11.Python里面match()和search()的区别？

答：re模块中match(pattern,string[,flags]),检查string的开头是否与pattern匹配。

re模块中research(pattern,string[,flags]),在string搜索pattern的第一个匹配值。

>>>print(re.match(‘super’, ‘superstition’).span())

(0, 5)

>>>print(re.match(‘super’, ‘insuperable’))

None

>>>print(re.search(‘super’, ‘superstition’).span())

(0, 5)

>>>print(re.search(‘super’, ‘insuperable’).span())

(2, 7)

12.用Python匹配HTML tag的时候，<.\*>和<.\*?>有什么区别？

答：术语叫贪婪匹配( <.\*> )和非贪婪匹配(<.\*?> )

例如:

test

<.\*> :

test

<.\*?> :

13.Python里面如何生成随机数？

答：random模块

随机整数：random.randint(a,b)：返回随机整数x,a<=x<=b

random.randrange(start,stop,[,step])：返回一个范围在(start,stop,step)之间的随机整数，不包括结束值。

随机实数：random.random( ):返回0到1之间的浮点数

random.uniform(a,b):返回指定范围内的浮点数。

14.有没有一个工具可以帮助查找python的bug和进行静态的代码分析？

答：PyChecker是一个python代码的静态分析工具，它可以帮助查找python代码的bug, 会对代码的复杂度和格式提出警告

Pylint是另外一个工具可以进行codingstandard检查

# 简单基础

2.什么是lambda函数？它有什么好处?

答：lambda 表达式，通常是在需要一个函数，但是又不想费神去命名一个函数的场合下使用，也就是指匿名函数

lambda函数：首要用途是指短小的回调函数

用法: lambda [arguments] : expression

>>> a=lambdax,y:x+y

>>> a(3,11)

3.Python里面如何实现tuple和list的转换？

答：直接使用tuple和list函数就行了，type()可以判断对象的类型

7.介绍一下except的用法和作用？

答：try…except…except…[else…][finally…]

执行try下的语句，如果引发异常，则执行过程会跳到except语句。对每个except分支顺序尝试执行，如果引发的异常与except中的异常组匹配，执行相应的语句。如果所有的except都不匹配，则异常会传递到下一个调用本代码的最高层try代码中。

try下的语句正常执行，则执行else块代码。如果发生异常，就不会执行

如果存在finally语句，最后总是会执行。

8.Python中pass语句的作用是什么？

答：pass语句不会执行任何操作，一般作为占位符或者创建占位程序

15.如何在一个function里面设置一个全局的变量？

答：解决方法是在function的开始插入一个global声明：