1.如何反向迭代一个序列

```
#如果是一个list,最快的方法使用reverse
tempList = [1,2,3,4]
tempList.reverse()
for x in tempList:
    print x
```

```
#如果不是list,需要手动重排
templist = (1,2,3,4)
for i in range(len(templist)-1,-1,-1):
    print templist[i]
```

2.如何查询和替换一个文本中的字符串

```
#最简单的方法使用replace()
tempstr = "hello you hello python are you ok"
print tempstr.replace("you","python")
```

```
#还可以使用正则,有个sub()
tempstr = "hello you hello python are you ok"
import re
rex = r'(hello|Use)'
print re.sub(rex,"Bye",tempstr)
```

3.使用python实现单例模式

```
#方法一:可以使用__new__方法
#在__new__方法中把类实例绑定到类变量_instance上，如果cls._instance为None表示该类还没有实例化过，实例化该类并返回。如果cls_instance不为None表示该类已实例化，直接返回cls_instance
class SingleTon(object):
    def __new__(cls,*args,**kwargs):
```

```
        if not hasattr(cls,'_instance'):
            cls._instance =
object.__new__(cls,*args,**kwargs)
        return cls._instance
class TestClass(SingleTon):
    a = 1

test1 = TestClass()
test2 = TestClass()
print test1.a,test2.a

test1.a=2
print test1.a,test2.a

print id(test1),id(test2)
```

```
#方法二:使用装饰器,建立过实例的就放到instances里面,下次建立的时候
先检查里面有没有
def SingleTon(cls,*args,**kwargs):
    instances = {}
    print instances
    def _singleton():
        if cls not in instances:
            instances[cls] = cls(*args,**kwargs)
        print instances
        return instances[cls]
    return _singleton

@SingleTon
class LastClass(object):
    a = 1
test1 = LastClass()
print test1.a
test2 = LastClass()
print test2.a
```

```
#方法三:使用__metaclass__(元类)关于元类看看这个吧;http://blog.jobbole.com/21351/
class SignalTon(type):
    def __init__(cls,name,bases,dict):
        super(SignalTon, cls).__init__(name,bases,dict)
        cls._instance = None

    def __call__(cls, *args, **kwargs):
        if cls._instance is None:
            cls._instance = super(SignalTon,cls).__call__(*args,**kwargs)
        return cls._instance

class TestClass(object):
    __metaclass__ = SignalTon

test1 = TestClass()
test2 = TestClass()

test1.a = 2
print test1.a,test2.a
print id(test1),id(test2)
```

```
#方法四:共享属性　所谓单例就是所有的引用（实例，对象）拥有相同的属性和方法，同一个类的实例天生都会有相同的方法，那我们只需要保证同一个类所产生的实例都具有相同的属性。所有实例共享属性最简单直接的方法就是共享__dict__属性指向。

class SingleTon(object):
    _state = {}
    def __new__(cls, *args, **kwargs):
        obj = object.__new__(cls,*args,**kwargs)
        obj.__dict__ = cls._state
        return obj

class TestClass(SingleTon):
```

```
    a = 1

test1 = TestClass()
test2 = TestClass()
print test1.a,test2.a
test1.a = 2
print test1.a,test2.a
print id(test1),id(test2)
#方法五:使用同一个模版
#写在mysingleton.py中
class My_Singleton(object):
    def foo(self):
        pass

my_singleton = My_Singleton()

#写在要使用这个实例的py文件里面,在不同的引用的地方都引用相同的实例,以此实现单例模式
from mysingleton import my_singleton
my_singleton.foo()
```

4.重新实现str.strip()

```
def rightStrip(tempStr,splitStr):
    endindex = tempStr.rfind(splitStr)
    while endindex != -1 and endindex == len(tempStr) - 1:
         tempStr = tempStr[:endindex]
         endindex = tempStr.rfind(splitStr)
    return tempStr

def leftStrip(tempStr,splitStr):
    startindex = tempStr.find(splitStr)
    while startindex == 0:
        tempStr = tempStr[startindex+1:]
        startindex = tempStr.find(splitStr)
```

```
    return tempStr

str = "  H  "
print str
print leftStrip(str,' ')
print rightStrip(str,' ')
#输出
  H
H
  H
```

5.super的原理

```
#阅读下面的代码，它的输出结果是什么?
class A(object):
  def __init__(self):
   print "enter A"
   super(A, self).__init__()  # new
   print "leave A"

 class B(object):
  def __init__(self):
   print "enter B"
   super(B, self).__init__()  # new
   print "leave B"

 class C(A):
  def __init__(self):
   print "enter C"
   super(C, self).__init__()
   print "leave C"

 class D(A):
  def __init__(self):
   print "enter D"
   super(D, self).__init__()
```

```
    print "leave D"
 class E(B, C):
  def __init__(self):
    print "enter E"
    super(E, self).__init__()  # change
    print "leave E"

 class F(E, D):
  def __init__(self):
    print "enter F"
    super(F, self).__init__()  # change
    print "leave F"

#输出

 enter F
 enter E
 enter B
 enter C
 enter D
 enter A
 leave A
 leave D
 leave C
 leave B
 leave E
 leave F
```

非常棒的讲解:

http://www.cnblogs.com/lovemo1314/archive/2011/05/03/2035005.html

6.包

常用的装饰器就是闭包的一种

```
def make_adder(addend):
    def adder(addend):
        return addend+addend
return adder

P1 = make_adder(5)
P2= make_adder(4)

print p1(10)
#输出15
print p2(10)
#输出14
```

闭包（Closure）是词法闭包（Lexical Closure）的简称，是引用了自由变量的函数。这个被引用的自由变量将和这个函数一同存在，即使已经离开了创造它的环境也不例外 http://www.cnblogs.com/ma6174/archive/2013/04/15/3022548.html

https://foofish.net/python-closure.html

7.给列表中的字典排序

list 对象 alist [{“name”:”a”,”age”:20},{“name”:”b”,”age”:30},
{“name”:”c”,”age”:25}]按照 age 从大到小排序

```
alist = [{"name":"a","age":20},{"name":"b","age":30},
{"name":"c","age":25}]
alist.sort(key=lambda:x:-x.get("age"))
print alist
```

8.合并两个列表排除重复元素

用简洁的方法合并alist = [‘a’,’b’,’c’,’d’,’e’,’f’] blist = [‘x’,’y’,’z’,’e’,’f’]并且元素不能重复

```
alist = ['a','b','c','d','e','f']
blist = ['x','y','z','e','f']
def merge_list(*args):
    s = set()
    for i in args:
        s = s.union(i)
    print(s)
    return s

merge_list(alist,blist)
```

9.打乱一个排好序的列表

```
from random import shuffle
alist = range(10)
print(alist)
shuffle(alist)
print(alist)
```

10.简单的实现一个栈结构 stack

```
class Stack(object):
    def __init__(self):
        self.value = []

    def push(self,x):
        self.value.append(x)

    def pop(self):
        self.value.pop()

stack = Stack()

stack.push(1)
stack.push(2)
stack.push(3)
```

```
print(stack.value)
stack.pop()
print(stack.value)
```

11.输入一个日期,返回时一年中的哪一天

```
from datetime import datetime
def which_day(year,month,day):
    return (datetime(year,month,day)-datetime(year,1,1)).days+1

print(which_day(2017,1,15))
```

12.把字符串”k1:1|k2:2|k3:3”处理成 python 字典的形式:{k1:1,k2:2,k3:3}

```
def string_to_dict(string):
    d = {}
    for kv in string.split("|"):
        k,v = kv.split(":")
        if v.isdigit():
            v=int(v)
        d[k]=v
    return d

print(string_to_dict("k1:1|k2:2|k3:3"))
```

13.判断输入的值是否在矩阵之中(杨氏矩阵)

在一个二维数组之中,每一行都按照从走到右递增的顺序排序,每一列到按照从上到下的顺序排序.请完成一个函数,输入这样的一个二维手术和一个整数,判断数组中是否含有该整数

```
#处理数组矩阵
```

```
arr = [[1,4,7,10,15],[2,5,8,12,19],[3,6,9,16,22],
[10,13,14,17,24],[18,21,23,26,30]]
def get_num(num,data=None):
    while data:
        if num > data[0][-1]:
            del data[0]
        elif num<data[0][-1]:
            data = list(zip(*data))
            del data[-1]
            data = list(zip(*data))
        else:
            return True
            data.clear()
    return False
print (get_num(18,arr))
```

不处理数组矩阵 使用 step-wise 线性搜索

```
def getvalue(data,value):
    m = len(data)-1
    n = len(data[0])-1
    r = 0
    c = n
    while c>=0 and r<=m:
        if value == data[r][c]:
            return True
        elif value>data[r][c]:
            r = r+1
        else:
            c = c-1
    return False
```

14.获取最大公约数(欧几里得算法)

```
a= 25
b=15

def max_common(a,b):
    while b:
        a,b=b,a%b
    return a
```

详解:

https://blog.csdn.net/franktan2010/article/details/38229641

15.求两个数的最小公倍数(公式法)

两个数的乘积等于这两个数的 最大公约数与最小公倍数的积

```
a=25
b=15
def min_common(a,b):
    c= a*b
    while b:
        a,b=b,a%b
    return c//a
```

详情:

https://zhidao.baidu.com/question/90232880

16.获取中位数

如果总数个数是奇数，按从小到大的顺序，取中间的那个数；如果总数个数是偶数个的话，按从小到大的顺序，取中间那两个数的平均数。

```
#计算中位数
def mediannum(num):
    listnum = [num[i] for i in range(len(num))]
    listnum.sort()
    lnum = len(num)
    if lnum % 2 == 1:
        i = int((lnum + 1) / 2)-1
        return listnum[i]
    else:
        i = int(lnum / 2)-1
        return (listnum[i] + listnum[i + 1]) / 2
```

详情:

https://blog.csdn.net/qq_33363973/article/details/78773144

```
def medin(data):
    data.sort()
    half = len(data)//2
    return (data[half]+data[~half])/2
l = [1,3,4,53,2,46,8,42,82]
print (median(l))
```