# **1. 字符串介绍**

想一想：

当打来浏览器登录某些网站的时候，需要输入密码，浏览器把密码传送到服务器后，服务器会对密码进行验证，其验证过程是把之前保存的密码与本次传递过去的密码进行对比，如果相等，那么就认为密码正确，否则就认为不对；服务器既然想要存储这些密码可以用数据库（比如MySQL），当然为了简单起见，咱们可以先找个变量把密码存储起来即可；那么怎样存储带有字母的密码呢？

答：字符串

## 1.1. Python中字符串的格式

如下定义的变量a，存储的是数字类型的值

a = 100

如下定义的变量b，存储的是字符串类型的值

b = "hello atguigu.com"

或者

b = 'hello atguigu.com'

c = '100'

c = '18.8.8'

**小总结：**

**双引号或者单引号中的数据，就是字符串.**

## 1.2. 字符串在内存中的存储

### 1.2.1. 计算机中的存储单位

Python 在 sys 模块中提供函数 getsizeof 来计算 Python 对象的大小。

sys.getsizeof(object[, default])

以**字节（byte）**为单位返回对象大小。

数据存储是以10[进制](http://baike.baidu.com/view/15954.htm" \t "https://my.oschina.net/repine/blog/_blank)表示，数据传输是以2进制表示的，所以1KB不等于1000B。

1KB=1024B；1MB=1024KB=1024×1024B。其中1024=210。

1B（byte，字节）= 8 bit（见下文）；

1KB（Kibibyte，[千字节](http://baike.baidu.com/view/4420688.htm" \t "https://my.oschina.net/repine/blog/_blank)）=1024B= 2^10 B；

1[MB](http://baike.baidu.com/view/35572.htm" \t "https://my.oschina.net/repine/blog/_blank)（Mebibyte，[兆字节](http://baike.baidu.com/view/3592561.htm" \t "https://my.oschina.net/repine/blog/_blank)，百万字节，简称“兆”）=1024KB= 2^20 B；

1[GB](http://baike.baidu.com/view/50311.htm" \t "https://my.oschina.net/repine/blog/_blank)（Gigabyte，[吉字节](http://baike.baidu.com/view/8445997.htm" \t "https://my.oschina.net/repine/blog/_blank)，十亿字节，又称“千兆”）=1024MB= 2^30 B；

1[TB](http://baike.baidu.com/view/347017.htm" \t "https://my.oschina.net/repine/blog/_blank)（Terabyte，万亿字节，[太字节](http://baike.baidu.com/view/3592539.htm" \t "https://my.oschina.net/repine/blog/_blank)）=1024GB= 2^40 B；

**1[PB](http://baike.baidu.com/view/344884.htm" \t "https://my.oschina.net/repine/blog/_blank)（Petabyte，千万亿字节，[拍字节](http://baike.baidu.com/view/3592577.htm" \t "https://my.oschina.net/repine/blog/_blank)）=1024TB= 2^50 B**；

1[EB](http://baike.baidu.com/view/48988.htm" \t "https://my.oschina.net/repine/blog/_blank)（Exabyte，百亿亿字节，[艾字节](http://baike.baidu.com/view/2536975.htm" \t "https://my.oschina.net/repine/blog/_blank)）=1024PB= 2^60 B；

1[ZB](http://baike.baidu.com/view/228795.htm" \t "https://my.oschina.net/repine/blog/_blank)（Zettabyte，十万亿亿字节，[泽字节](http://baike.baidu.com/view/8438227.htm" \t "https://my.oschina.net/repine/blog/_blank)）= 1024EB= 2^70 B；

1[YB](http://baike.baidu.com/view/478158.htm" \t "https://my.oschina.net/repine/blog/_blank)（Yottabyte，一亿亿亿字节，[尧字节](http://baike.baidu.com/view/8438250.htm" \t "https://my.oschina.net/repine/blog/_blank)）= 1024ZB= 2^80 B；

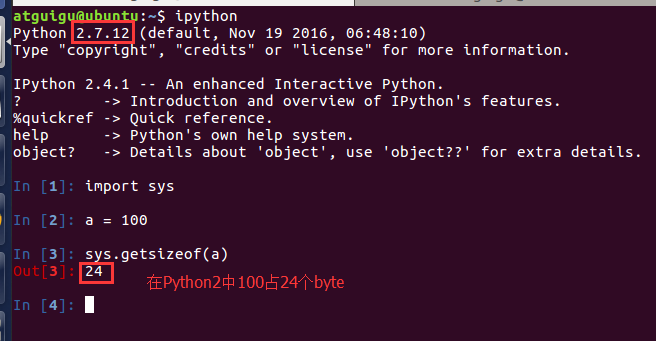
1[BB](http://baike.baidu.com/view/86977.htm" \t "https://my.oschina.net/repine/blog/_blank)（Brontobyte，一千亿亿亿字节）= 1024YB= 2^90 B；

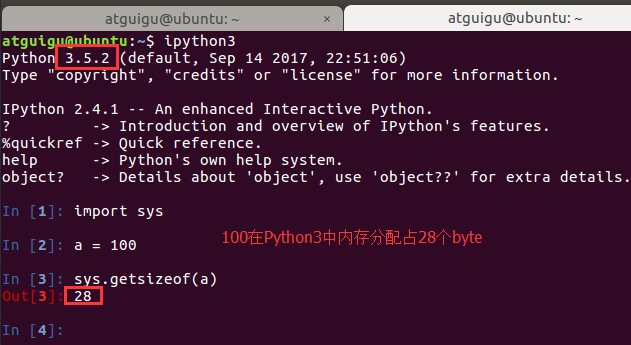
1NB（NonaByte，一百万亿亿亿字节） = 1024 BB = 2^100 B；

1DB（DoggaByte，十亿亿亿亿字节） = 1024 NB = 2^110 B；



**注意** ，注意同样是int 类型的同一个值，在Python2和Python3的长度是不一样的。





### 1.2.2. 为什么运营商提供的带宽与下载速度不一样

网络带宽的单位与下载的单位，其实是不同的。

**运营商所说的“带宽”是数据传输的单位，即每秒可以传送多少个0或1，通常叫“速率”。单位是比特率（bps）。**

**我们所说的“速度”是传输文件容量的单位，即每秒可以传送多少八个一组0或1组成的数据组，通常叫“速度”。单位是字节(Byte)。**

比特率（bps），

​在数据传输中，数据通常是串行传输的，即一个0或1接一个0或1地传输。比特率就是每秒串行通过的位数。

字节(Byte)

​是记忆体储存文件和传输文件容量的基本单位，1个字节等于8位二进制。即1个字节与八个0或1所占的空间是相同的。

一个字节可代表一个字元(A~Z)、数字(0~9)、或符号(,.?!%&+-\*/)，至於每个中文字则须要两bytes。   当记忆体容量过大时，位元组这个单位就不够用，因此就有千位元组的单位出现，以下乃个记忆体计算单位之间的相关性：

1 Byte = 8 Bits

1 KB = 1024 Bytes

1 MB = 1024 KB

1 GB = 1024 MB

1 TB = 1024 GB

**Kbps是什么意思？**

K是数量单位（即1024），b(bit)是二进制位数（即0或1）。p(per)是每次的意思。s(second)是时间单位，秒的意思。

因此Kbps中的含义是：每秒能够传递多少次1024个二进制位数

KB/S是什么意思？

K是数量单位（即1024），B(Byte)字节（即八个0或1所占的空间），是最基础的存储空间单位，S(second)是时间单位，秒的意思。　B/S的含义就是：每秒传递多少字节数据。KB/S的含义就是每秒传输多少千字节数据，也就是多少KB数据

**实际应用中网速大约要减去30%的网络损耗**，这些损耗主要是受服务供应商因素所影响、用户本身所使用操作系统、浏览器、电脑配备等亦可影响宽带速度、跟你连接的网站的服务器能力，提供的带宽有密切关系。同时浏览同个网站的人数越多，这个网站所能提供给每个浏览者的带宽就越低)

例如：

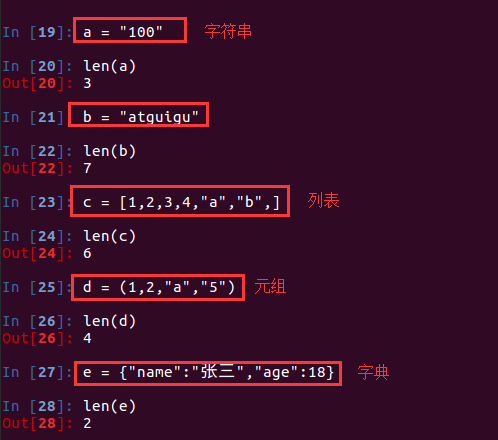
4M宽带，下载速度有400KB/S是正常的。

100M带宽的网络下载时，理论上的速度应该是100除以8等于12.5MB。

通俗讲就是每秒能传送100M个0或1的带宽，理论上下载文件的速度是100除以8等于每秒传送12.5M个字节的数据.

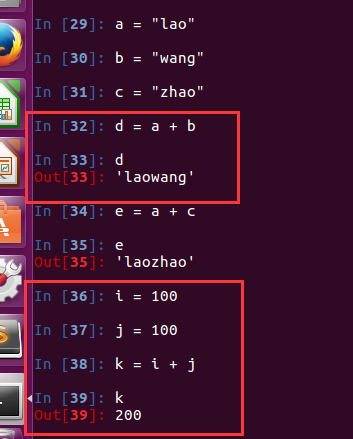
### 1.2.3 len函数返回对象的长度或者个数

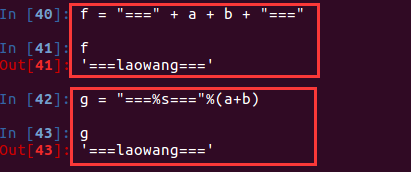
Python len() 方法返回对象（字符、列表、元组、字典等）长度或项目个数。



## 1.3. 组成字符串的另外一种方式

字符串会拼接，数字会相加





# 2. 字符串输出

## 2.1. 把你的职业信息输出

#输入信息

name = input("请输入你的姓名:")

position = input("请输入你的职业:")

address = input("请输入你的地址:")

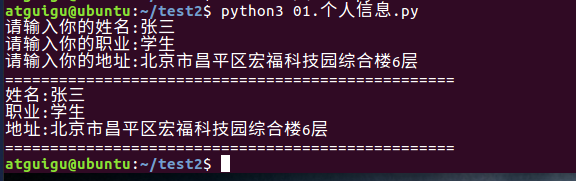
#输入信息

print("="\*50)

print("姓名:%s\n职业:%s\n地址:%s" % (name,position,address))

print("="\*50)

## 2.2. 结果



# **3. 字符串输入**

之前在学习input的时候，通过它能够完成从键盘获取数据，然后保存到指定的变量中；

注意：在python3中**input获取的数据，都以字符串的方式进行保存，即使输入的是数字，那么也是以字符串方式保存**

## 3.1. 判断密码是否正确

user\_name = input("请输入用户名:")

password = input("请输入密码:")

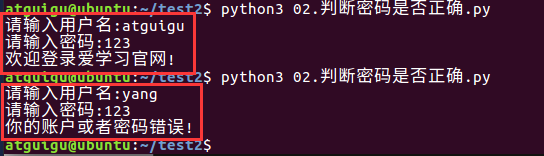
if user\_name == "atguigu" and password == "123" :

print("欢迎登录爱学习官网!")

else :

print("你的账户或者密码错误!")

结果：（根据输入的不同结果也不同）



# 4. 下标和切片

## 4.1.下标索引

所谓“下标”，就是编号，就好比超市中的存储柜的编号，通过这个编号就能找到相应的存储空间。

生活中的 "下标"超市储物柜



高铁二等座



高铁一等座



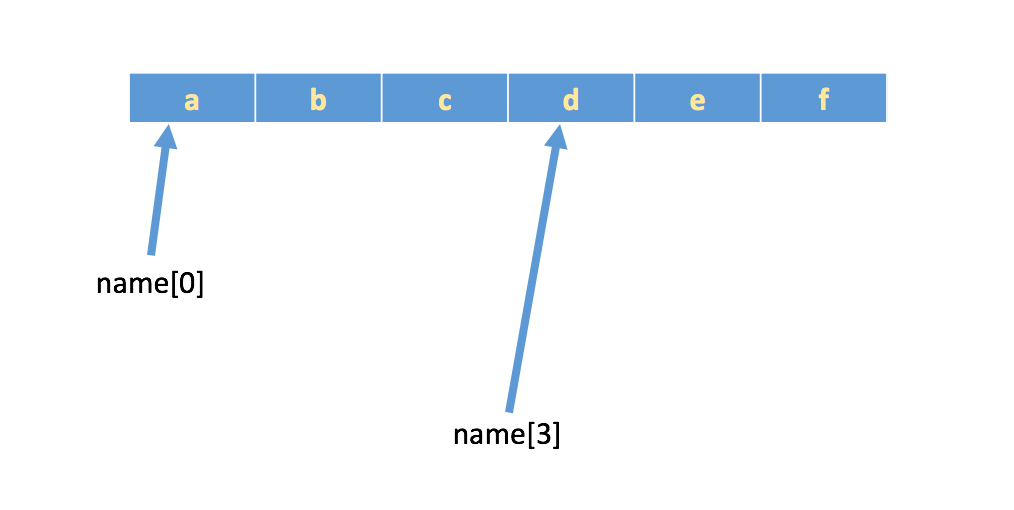
绿皮车



字符串中"下标"的使用

**列表与元组支持下标索引好理解，字符串实际上就是字符的数组，所以也支持下标索引。**

### 4.1.1. 通过下标取出部分字符

如果有字符串:name = 'abcdef'，在内存中的实际存储如下:

如果想取出部分字符，那么可以通过下标的方法，（注意python中下标从 0 开始）

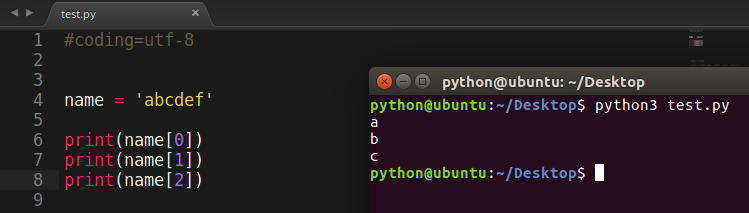
name = 'abcdef'

print(name[0])

print(name[1])

print(name[2])

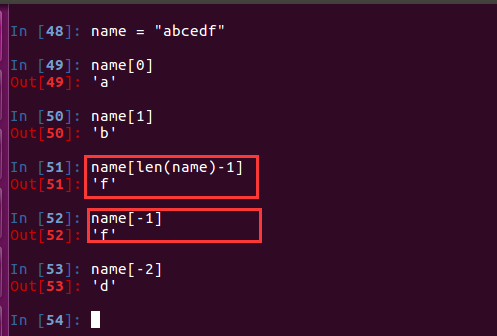
运行结果:



如果超出下标会报错



### 4.1.2. 从右边取字符串



## 4.2. 切片

**切片是指对操作的对象截取其中一部分的操作**。字符串、列表、元组都支持切片操作。

**切片的语法：[起始:结束:步长]**

注意：选取的区间属于**左闭右开型**，即从"起始"位开始，到"结束"位的前一位结束（不包含结束位本身)，注意，如果不写步长默认是1.

步长是控制方向的，1从左往右取，-1是从右到左取

我们以字符串为例讲解。

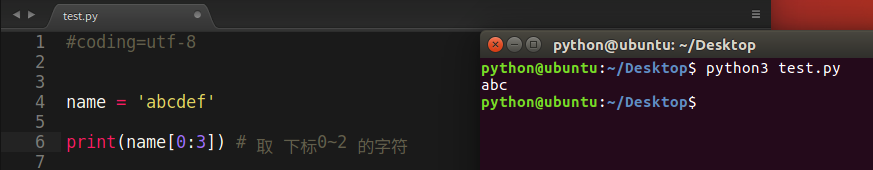
如果取出一部分，则可以在中括号[]中，使用:

### 4.2.1. 取下标0~2 的字符

name = 'abcdef'

print(name[0:3]) # 取 下标0~2 的字符

运行结果:

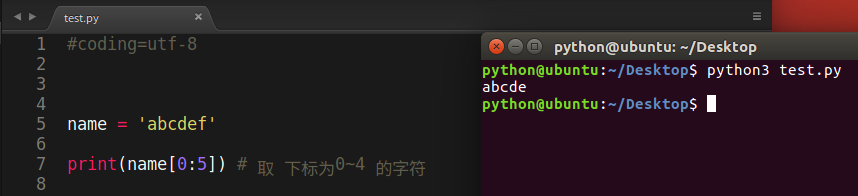


### 4.2.2. 取下标为0~4 的字符

name = 'abcdef'

print(name[0:5]) # 取 下标为0~4 的字符

运行结果:

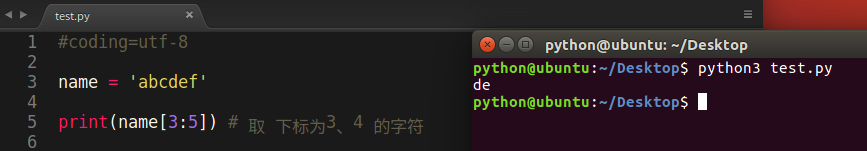


### 4.2.3. 取下标为3~4 的字符

name = 'abcdef'

print(name[3:5]) # 取 下标为3、4 的字符

运行结果:

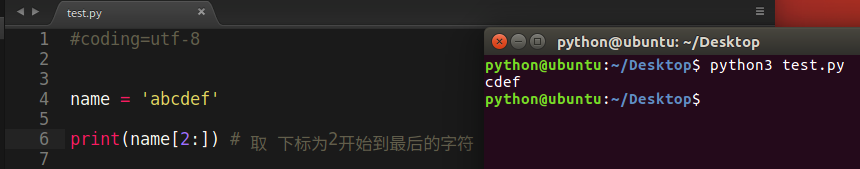


### 4.2.4. 取下标为2开始到最后的字符

name = 'abcdef'

print(name[2:]) # 取 下标为2开始到最后的字符

运行结果:

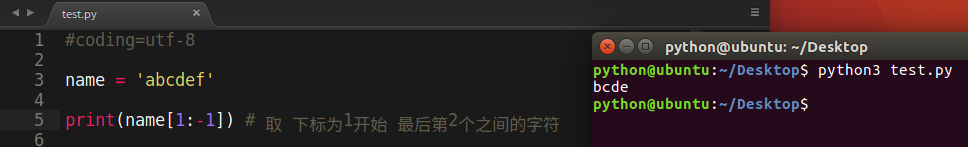


### 4.2.5. 取下标为1开始到最后第2个之间的字符

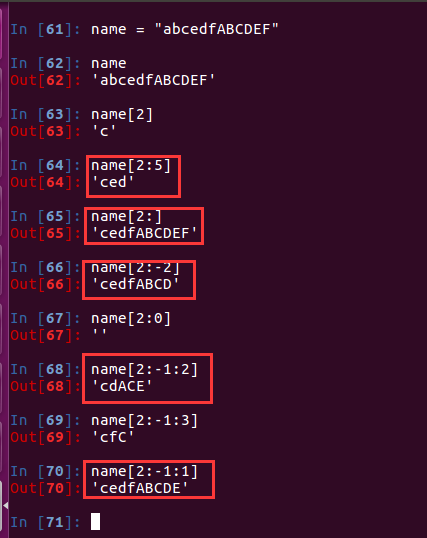
name = 'abcdef'

print(name[1:-1]) # 取 下标为1开始 到 最后第2个 之间的字符

运行结果:



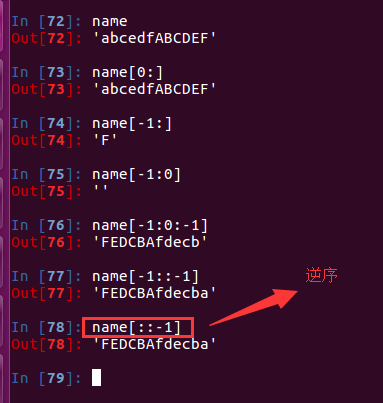
### 4.2.6. 综合练习



## 4.3 逆序（倒序）

### **4.3.1. 想一想**

（面试题）给定一个字符串aStr, 请反转字符串



## 4.4. 小结

[:] 提取从开头（默认位置0）到结尾（默认位置-1）的整个字符串  
[start:] 从start 提取到结尾  
[:end] 从开头提取到end - 1  
[start:end] 从start 提取到end - 1  
[start:end:step] 从start 提取到end - 1，每step 个字符提取一个

[::-1]逆序

# 5. 字符串常见函数

如有字符串mystr = 'hello world atguigu and atguiguPython' 以下是常见的操作

## 5.1.必须掌握字符串函数

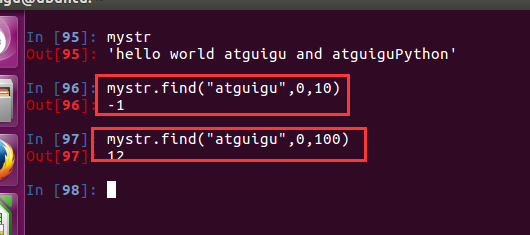
注意，下面这些函数执行后并没有改变原字符串.

### 5.1.1. find

检测 str 是否包含在 mystr中，如果是返回开始的索引值，否则返回-1

mystr.find(str, start=0, end=len(mystr))

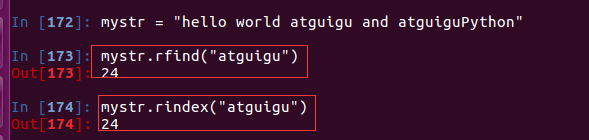




### 5.1.2. rfind

类似于 find()函数，不过是从右边开始查找.

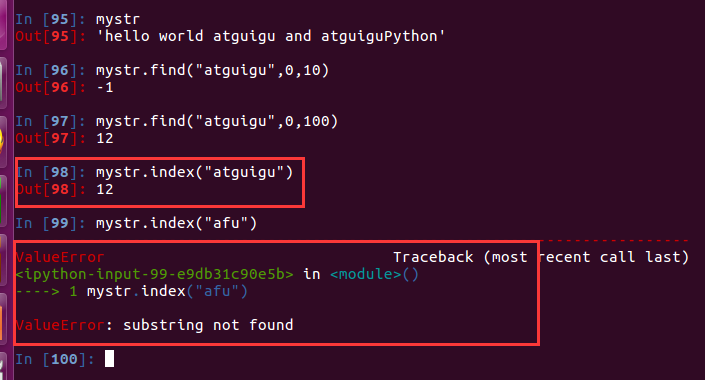
mystr.rfind(str, start=0,end=len(mystr) )

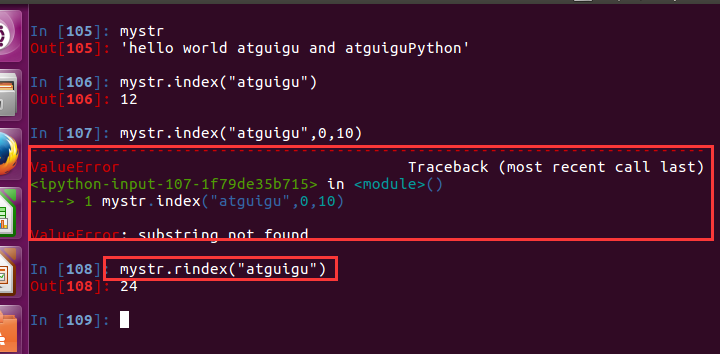


### 5.1.3. index

index()跟find()方法一样，只不过如果str不在 mystr中会报一个异常.

mystr.index(str, start=0, end=len(mystr))

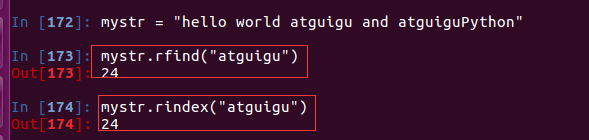




### 5.1.4. rindex

类似于 index()，不过是从右边开始.

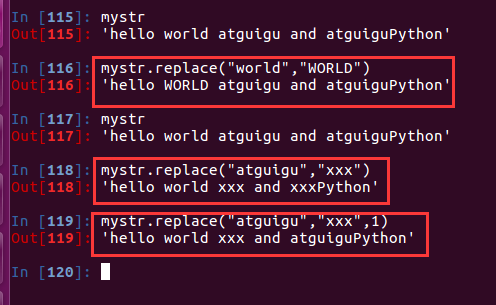
mystr.rindex( str, start=0,end=len(mystr))



### 5.1.5. replace

把 mystr 中的 str1 替换成 str2,如果 count 指定，则替换不超过 count 次，最多等于count

mystr.replace(str1, str2, mystr.count(str1))



注意：没有从右边开始替换的方法

### 5.1.6. split

以 str 为分隔符切片 mystr，如果 maxsplit有指定值，则仅分隔 maxsplit 个子字符串

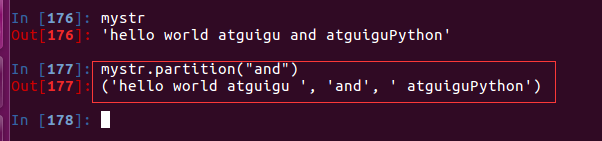
mystr.split(str=" ", maxsplit)



### 5.1.7. partition

把mystr以str分割成三部分,str前，str自身和str后

mystr.partition(str)

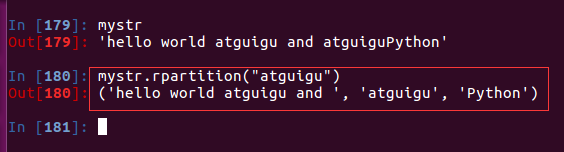


### 5.1.8. rpartition

partition[pɑ:ˈtɪʃn]分割

类似于 partition()函数,不过是从右边开始.

mystr.rpartition(str)

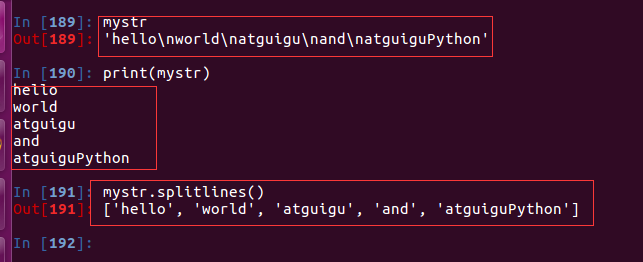


### 5.1.9. splitlines

按照行分隔，返回一个包含各行作为元素的列表,按照换行符分割

mystr = 'hello\nworld\natguigu\nand\natguiguPython'

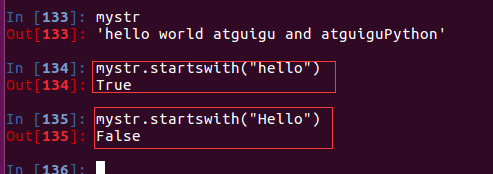
mystr.splitlines()



### 5.1.10. startswith

检查字符串是否是以 obj 开头, 是则返回 True，否则返回 False

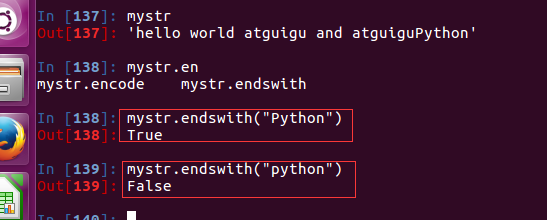
mystr.startswith(obj)



### 5.1.11. endswith

检查字符串是否以obj结束，如果是返回True,否则返回 False.

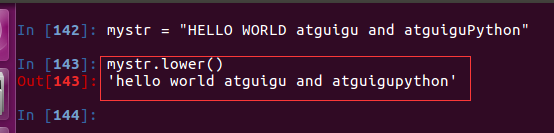
mystr.endswith(obj)



### 5.1.12. lower

转换 mystr 中所有大写字符为小写

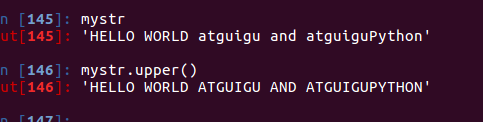
mystr.lower()



### 5.1.13. upper

转换 mystr 中的小写字母为大写

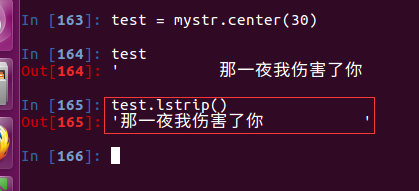
mystr.upper()



### 5.1.14. lstrip

删除 mystr 左边的空白字符

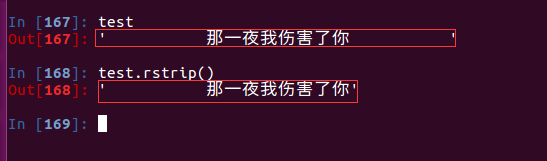
mystr.lstrip()



### 5.1.15. rstrip

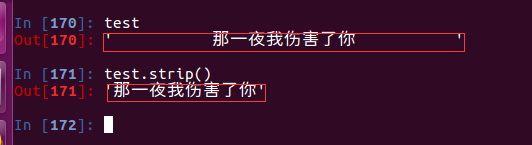
删除 mystr 字符串末尾的空白字符

mystr.rstrip()



5.1.11. strip

删除mystr字符串两端的空白字符

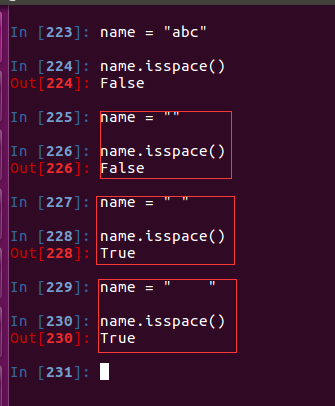


### 5.1.16. isspace

如果 mystr 中**只包含空格**，则返回 True，否则返回 False.

如果有多个空格同样返回True

mystr.isspace()

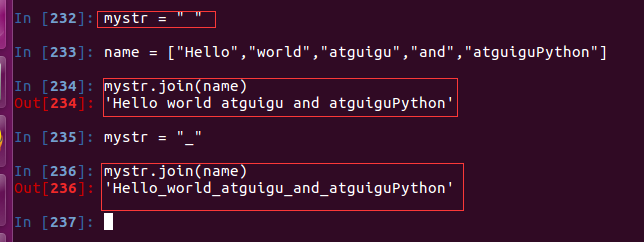


### 5.1.17. join

name = ['hello', 'world', 'atguigu', 'and', 'atguiguPython']

mystr 中每个字符后面插入str,构造出一个新的字符串

mystr.join(str)

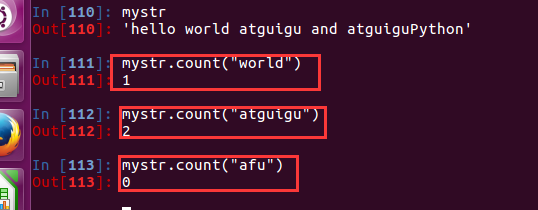


## 5.2.扩展字符串函数

### 5.2.1. count

返回 str在start和end之间 在 mystr里面出现的次数

mystr.count(str, start=0, end=len(mystr))

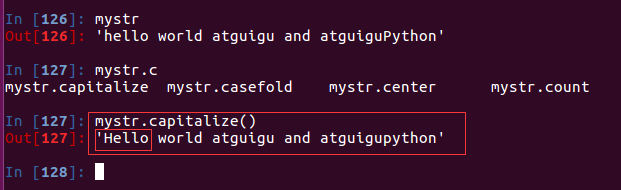


### 5.2.2. capitalize

[ˈkæpɪtəlaɪz]

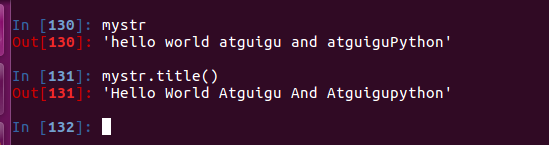
把字符串的第一个字符大写

mystr.capitalize()



### 5.2.3. title

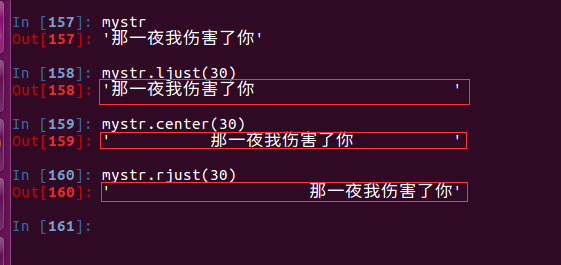
把字符串的**每个单词首字母大写**



### 5.2.4. ljust

返回一个原字符串左对齐,并使用空格填充至长度 width 的新字符串

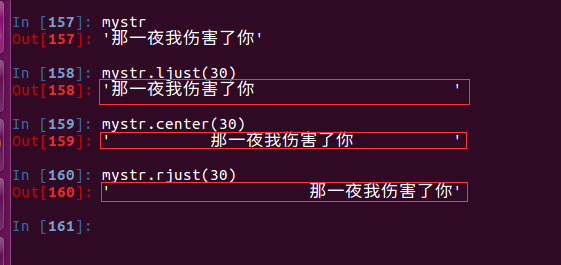
mystr.ljust(width)



### 5.2.5. rjust

返回一个原字符串右对齐,并使用空格填充至长度 width 的新字符串

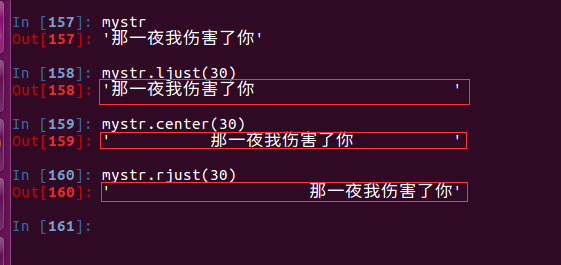
mystr.rjust(width)



### 5.2.6. center

返回一个原字符串居中,并使用空格填充至长度 width 的新字符串

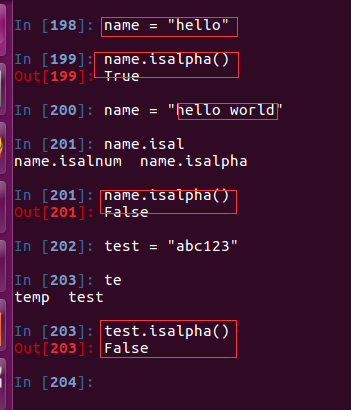
mystr.center(width)

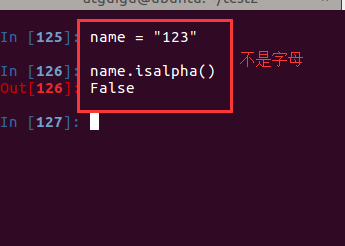


### 5.2.7. isalpha

如果 mystr 所有字符都连续的是**字母** 则返回 True,否则返回 False

mystr.isalpha()





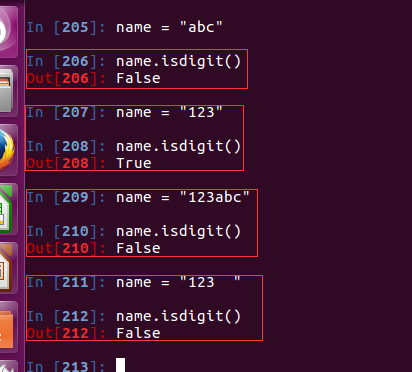
### 5.2.8. isdigit

[ˈdɪdʒɪt]

如果 mystr **只包含数字**则返回 True 否则返回 False.

应用场景：输入数字的时候，判断是纯数字的时候，就可以转成ini

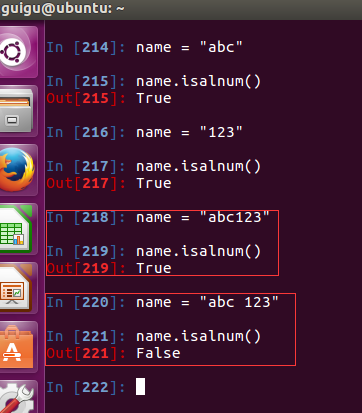
mystr.isdigit()



### 5.2.9. isalnum

如果 mystr 所有字符都是**字母或数字**则返回 True,否则返回 False

mystr.isalnum()



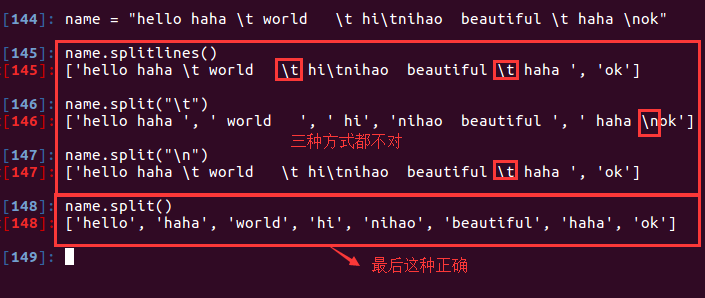
## 5.3. 想一想

给定一个字符串aStr，返回使用空格或者'\t'分割后的子串,要求里面不能有\n或者\t。

给定字符串：

name = "hello haha \t world \t hi\tnihao beautiful \t haha \nok"

思路：



# 6. 列表(list)介绍

想一想：

前面学习的字符串可以用来存储一串信息，那么想一想，怎样存储咱们班所有同学的名字呢？

定义100个变量，每个变量存放一个学生的姓名可行吗？有更好的办法吗？

答：

列表

## 6.1. 列表的格式

变量A的类型为列表

namesList = ['xiaoWang','xiaoZhang','xiaoHua']

### 6.1.1 Python比java和C语言的数组强大的地方在于列表中的元素可以是不同类型的

testList = [1, 'a']

## 6.2. 打印列表

### 6.2.1 打印多个姓名

namesList = ['xiaoWang','xiaoZhang','xiaoHua']

print(namesList[0])

print(namesList[1])

print(namesList[2])

结果：

xiaoWang

xiaoZhang

xiaoHua

# 7. 列表的循环遍历

## 7.1. 使用for循环

为了更有效率的输出列表的每个数据，可以使用循环来完成

### 7.1.1. 打印多个姓名

names\_list = ['xiaoWang','xiaoZhang','xiaoHua']

for name in names\_list:

print(name)

结果:

xiaoWang

xiaoZhang

xiaoHua

## 7.2. 使用while循环

为了更有效率的输出列表的每个数据，可以使用循环来完成

### 7.2.1. 打印多个姓名

names\_list= ['xiaoWang','xiaoZhang','xiaoHua']

length = len(names\_list)

i = 0

while i<length:

print(names\_list[i])

i+=1

结果:

xiaoWang

xiaoZhang

xiaoHua

# 8. 列表的增删改查

列表中存放的数据是可以进行修改的，比如"增"、"删"、"改""

## 8.1. 添加元素("增"append, extend, insert)

### 8.1.1. append可以向列表添加元素

通过append可以向列表添加元素

#定义变量A，默认有3个元素

A = ['xiaoWang','xiaoZhang','xiaoHua']

print("-----添加之前，列表A的数据-----")

for tempName in A:

print(tempName)

#提示、并添加元素

temp = input('请输入要添加的学生姓名:')

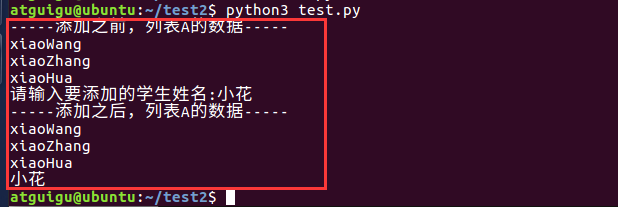
A.append(temp)

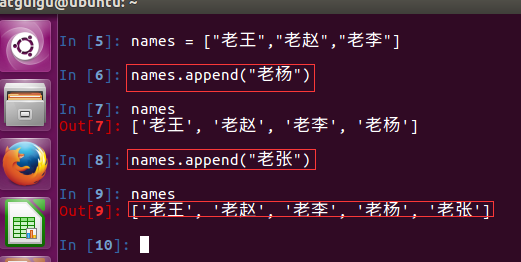
print("-----添加之后，列表A的数据-----")

for tempName in A:

print(tempName)

结果:





### 8.1.2. extend将另一个集合中的元素逐一添加到列表中

通过extend可以将另一个集合中的元素逐一添加到列表中

>>> a = [1, 2]

>>> b = [3, 4]

>>> a.append(b)

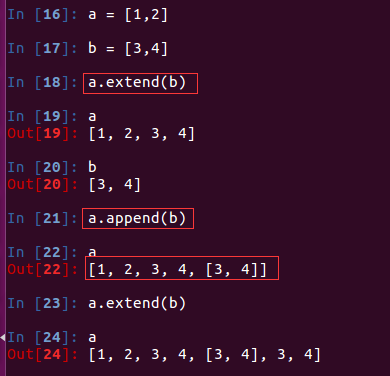
>>> a

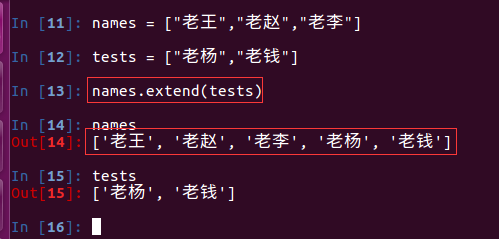
[1, 2, [3, 4]]

>>> a.extend(b)

>>> a

[1, 2, [3, 4], 3, 4]





### 8.1.3. insert在指定位置index前插入元素

注意只能插入元素

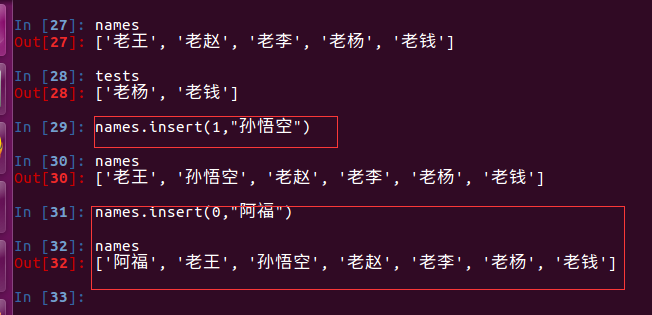
insert(index, object) 在指定位置index前**插入元素**object

>>> a = [0, 1, 2]

>>> a.insert(1, 3)

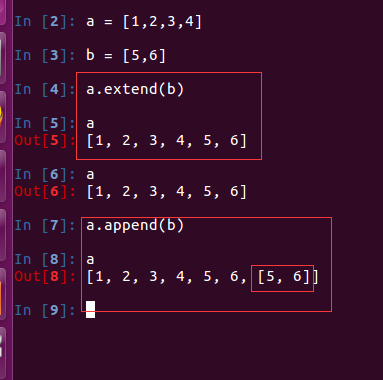
>>> a

[0, 3, 1, 2]

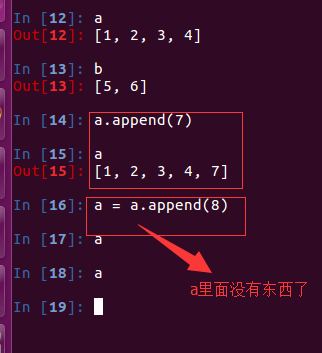


### 8.1.4. append和extend的区别和append的注意事项

**append和extend的区别**



**append的注意事项**



## 8.2. 删除元素("删"del, pop, remove)

类比现实生活中，如果某位同学调班了，那么就应该把这个调走后的学生的姓名删除掉；在开发中经常会用到删除这种功能。

列表元素的常用删除方法有：

### 8.2.1. del根据下标进行删除

movieName = ['加勒比海盗','骇客帝国','第一滴血','指环王','霍比特人','速度与激情']

print('------删除之前------')

for tempName in movieName:

print(tempName)

**del movieName[2]**

print('------删除之后------')

for tempName in movieName:

print(tempName)

结果:

------删除之前------

加勒比海盗

骇客帝国

第一滴血

指环王

霍比特人

速度与激情

------删除之后------

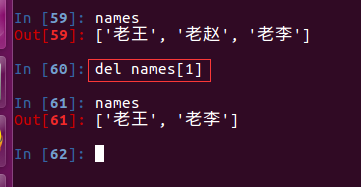
加勒比海盗

骇客帝国

指环王

霍比特人

速度与激情



### 8.2.2. pop删除最后一个元素

movieName = ['加勒比海盗','骇客帝国','第一滴血','指环王','霍比特人','速度与激情']

print('------删除之前------')

for tempName in movieName:

print(tempName)

**movieName.pop()**

print('------删除之后------')

for tempName in movieName:

print(tempName)

结果:

------删除之前------

加勒比海盗

骇客帝国

第一滴血

指环王

霍比特人

速度与激情

------删除之后------

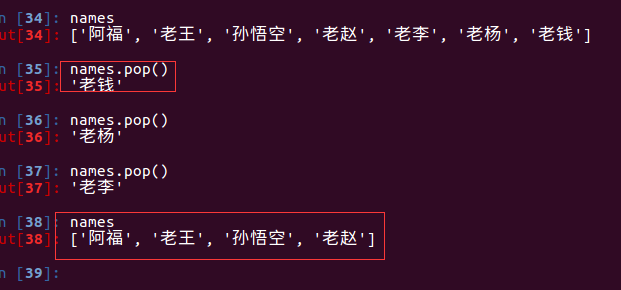
加勒比海盗

骇客帝国

第一滴血

指环王

霍比特人



### 8.2.3 remove根据元素的值进行删除

movieName = ['加勒比海盗','骇客帝国','第一滴血','指环王','霍比特人','速度与激情']

print('------删除之前------')

for tempName in movieName:

print(tempName)

**movieName.remove('指环王')**

print('------删除之后------')

for tempName in movieName:

print(tempName)

结果:

------删除之前------

加勒比海盗

骇客帝国

第一滴血

指环王

霍比特人

速度与激情

------删除之后------

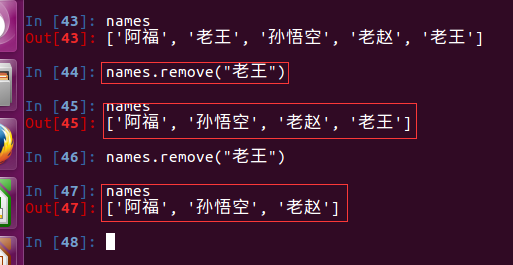
加勒比海盗

骇客帝国

第一滴血

霍比特人

速度与激情



注意，如果列表中有两个数据相同，移除的时候，值移除前一个数据

## 8.3. 通过下标修改元素("改")

修改元素的时候，要通过下标来确定要修改的是哪个元素，然后才能进行修改

### 8.3.1 修改名单

#定义变量A，默认有3个元素

A = ['xiaoWang','xiaoZhang','xiaoHua']

print("-----修改之前，列表A的数据-----")

for tempName in A:

print(tempName)

#修改元素

**A[1] = 'xiaoLu'**

print("-----修改之后，列表A的数据-----")

for tempName in A:

print(tempName)

结果:

-----修改之前，列表A的数据-----

xiaoWang

xiaoZhang

xiaoHua

-----修改之后，列表A的数据-----

xiaoWang

xiaoLu

xiaoHua

## 8.4. 查找元素("查"in, not in, index, count)

所谓的查找，就是看看指定的元素是否存在

### 8.4.1. in和not in

python中查找的常用方法为：

**in（存在）,如果存在那么结果为True，否则为False**

**not in（不存在），如果不存在那么结果为True，否则False**

**在名单中查找某个名字同学**

#待查找的列表

nameList = ['xiaoWang','xiaoZhang','xiaoHua']

#获取用户要查找的名字

findName = input('请输入要查找的姓名:')

#查找是否存在

**if findName in nameList**:

print('在字典中找到了相同的名字')

else:

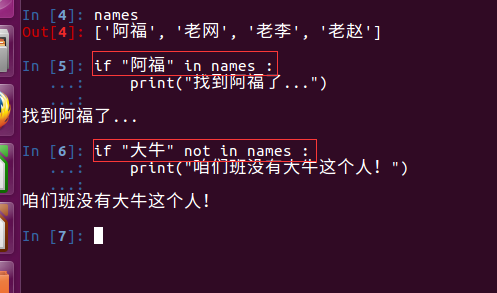
print('没有找到')

结果1：(找到)

结果2：(没有找到)

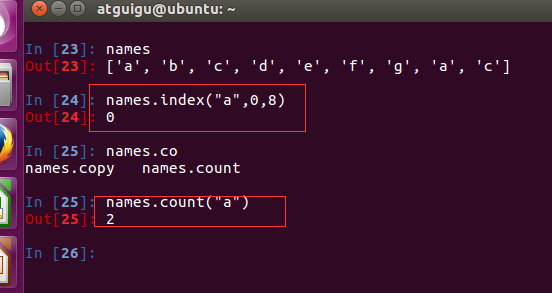
说明：

in的方法只要会用了，那么not in也是同样的用法，只不过not in判断的是不存在



### 8.4.2. index和count

index和count与字符串中的用法相同



## 8.5. 排序(sort, reverse)

sort方法是将list按特定顺序重新排列，默认为由小到大，参数reverse=True可改为倒序，由大到小。

reverse方法是将list逆置。

### 8.5.1. 列表排序案例

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

>>> a

[1, 4, 2, 3]

>>> a.reverse()

>>> a

[3, 2, 4, 1]

>>> a.sort()

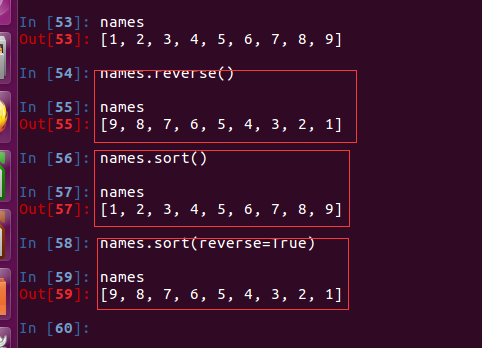
>>> a

[1, 2, 3, 4]

>>> a.sort(reverse=True)

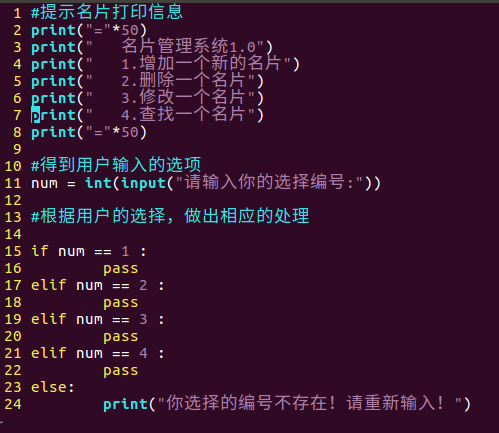
>>> a

[4, 3, 2, 1]

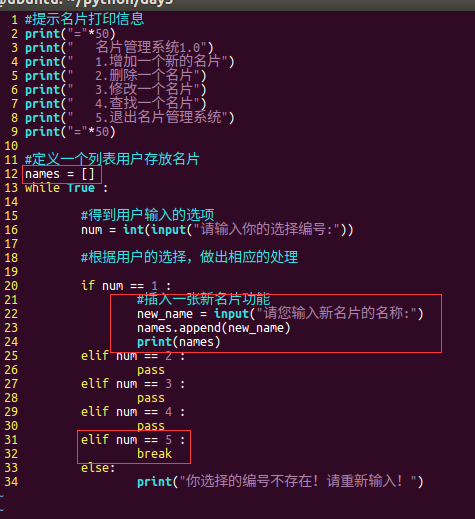


## 8.6. 综合应用-名片管理系统

### 8.6.1. 大体框架完成



### 8.6.2. 重复增加插入和退出功能



把剩下的功能完成

# 9. 列表的嵌套

## 9.1. 列表嵌套

类似while循环的嵌套，列表也是支持嵌套的

一个列表中的元素又是一个列表，那么这就是列表的嵌套

类似java中二维数组

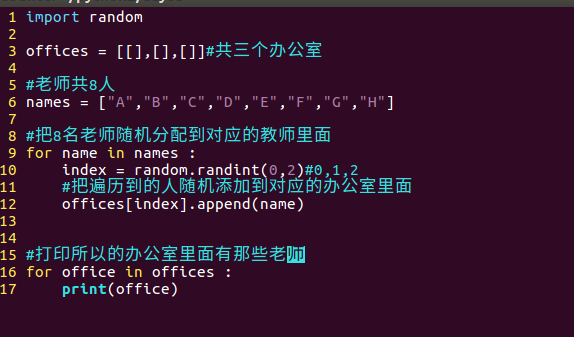
school\_names = [['北京大学','清华大学'],

['南开大学','天津大学','天津师范大学'],

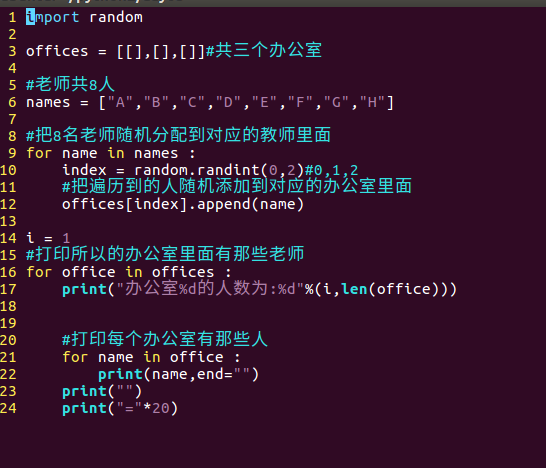
['山东大学','中国海洋大学']]

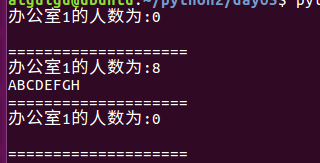
## 9.2. 列表嵌套的应用-随机安排老师工位

一个学校，有3个办公室，现在有8位老师等待工位的分配，请编写程序，完成随机的分配



完整版本





#encoding=utf-8

import random

# 定义一个列表用来保存3个办公室

offices = [[],[],[]]

# 定义一个列表用来存储8位老师的名字

names = ['A','B','C','D','E','F','G','H']

for name in names:

index = random.randint(0,2)

offices[index].append(name)

i = 1

for tempNames in offices:

print('办公室%d的人数为:%d'%(i,len(tempNames)))

i+=1

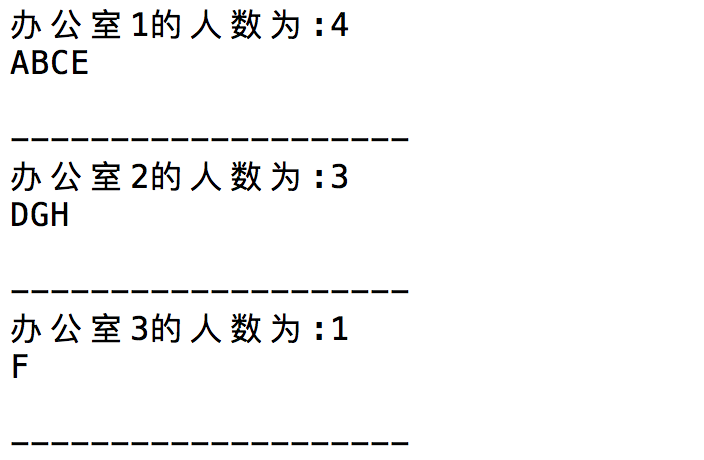
for name in tempNames:

print("%s"%name,end='')

print("\n")

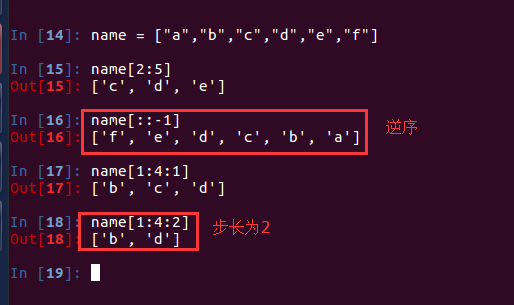
print("-"\*20)

运行结果如下:



扩展，如何保证至少一个办公室有一个人

# 10. 列表的切片



# 11. 总结与作业

## 1. 编写程序，完成以下要求：

统计字符串中，各个字符的个数

"hello world" 字符串统计的结果为： h:1 e:1 l:3 l:3 o:2 w:1 o:2 r:1 l:3 d:1

扩展：

如何把"hello world" 字符串统计的结果为： h:1 e:1 l:3 o:2 d:1 r:1 w:1

## 2. 编写程序，完成以下要求：

完成一个路径的组装

先提示用户多次输入路径，最后显示拼接一个完成的路径，比如/home/atguigu/ftp/share

## 3. 编写程序，完成“名片管理器”项目

需要完成的基本功能：

添加名片

删除名片

修改名片

查询名片

退出系统

程序运行后，除非选择退出系统，否则重复执行功能

1. 一个学校，有3个办公室，现在有8位老师等待工位的分配，请编写程序，完成随机的分配,要求办公室至少有一个人。

import random

offices =[[],[],[]]#三个办公室

#8个老师

names = ['A','B','C','D','E','F','G','H']

for office in offices :

name = names[random.randint(0,len(names)-1)]

office.append(name)

names.remove(name)

print(len(names))

for name in names :

index =random.randint(0,2)

offices[index].append(name)

for office in offices :

print(office)