# Python的变量操作

1. # Author:Alex Li

4. name = input("name:")
5. #raw\_input 2.x   input 3.x
6. #input 2.x =
7. age = int(input("age:") ) #integer
8. **print**(type(age)   , type(  str(age) ))
9. job = input("job:")
10. salary  = input("salary:")
11. #变量替代符号的应用
12. info = '''
13. -------- info of  %s  -----
14. Name:%s
15. Age:%d
16. Job:%s
17. Salary:%s
18. ''' % (name,name,age,job,salary)
20. info2 = '''
21. -------- info of {\_name}  -----
22. Name:{\_name}
23. Age:{\_age}
24. Job:{\_job}
25. Salary:{\_salary}
26. '''.format(\_name=name,
27. \_age=age,
28. \_job=job,
29. \_salary=salary)
31. info3 =  '''''
32. -------- info of {0} -----
33. Name:{0}
34. Age:{1}
35. Job:{2}
36. Salary:{3}
37. '''.format(name,age,job,salary)
38. **print**(info3)

1. 1.  **import** getpass
2. 2.
3. 3.  \_username = 'alex'
4. 4.  \_password = 'abc123'
5. 5.  username = input("username:")
6. 6.  #password = getpass.getpass("password:")  #getpass能够使输入的密码隐藏起来
7. 7.  password = input("password:")
8. 8.  **if** \_username == username **and** \_password == password:
9. 9.      **print**("Welcome user {name} login...".format(name=username))
10. 10. **else**:
11. 11.     **print**("Invalid username or password!")

**For ……… else** 循环，else后面的语句只有在for循环正常执行完毕的情况下，才会接着执行

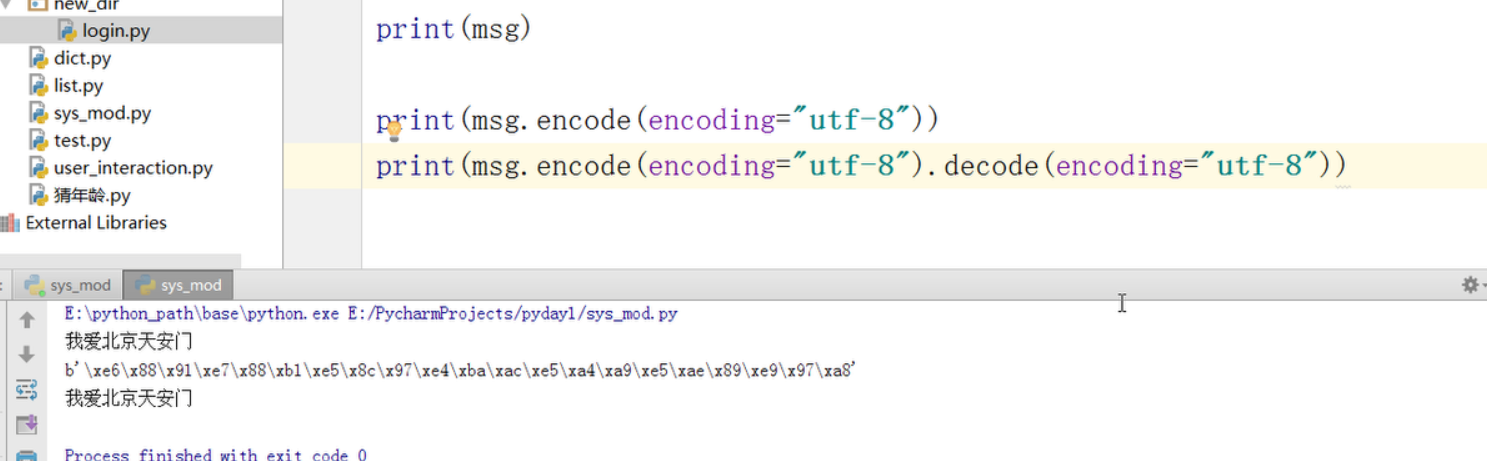
1. **for** i **in** range(0,10,2):
2. **print**(i)

#range（0,10,2）中2是指隔几个数取一个数

## 1、 三元运算：

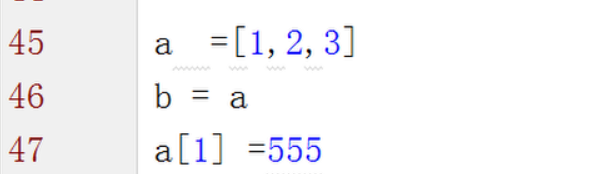
Result = 值1 if 条件 else 值2 当条件满足时，result为值1，条件不满足时result为值2

## 2、 字符编码



# 列表

**列表是指向一个内存地理位置的**

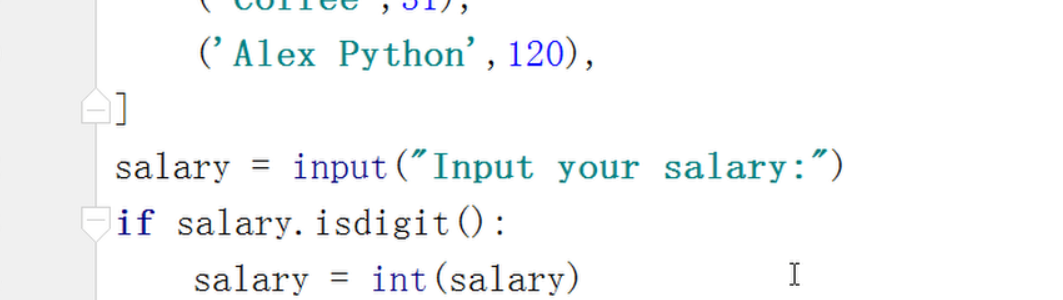


那么 print（b）时 output 是 [1, 555, 3]

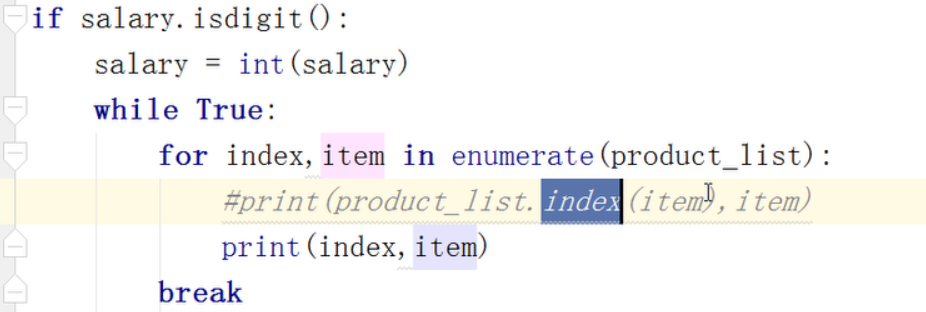
## 1、列表中运用切片，最后一个数2表示步长为2



## 2、isdigit()用来判断输入的数是否是数字，并且是整数

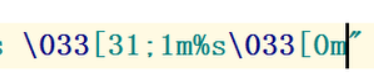


## 3、用enumerate来取出列表的小标和值，其中index是下表



## 4、让打印出来的内容高亮显示

其中%s中替代的内容在打印出来的时候会高亮显示



# 字符串

## 1、字符串操作

1. # Author:Alex Li
3. name = "my \tname is {name} and i am {year} old"
4. print(name.capitalize())
5. print(name.count("a")) #计算“a”在字符串中的数量
6. print(name.center(50,"-")) #输出为————my name is alex ————一共打印50个字符不够的话用—补上
7. print(name.endswith("ex")) #判断字符串以什么结尾，以“ex”结尾
8. print(name.expandtabs(tabsize=30)) #制表符的数量，30表示空格的数量
9. print(name[name.find("name"):]) #找到相应的字符，并返回起始位置索引值
10. print(name.format(name='alex',year=23)) #格式化输出，对应字符用{}包起来
11. print(name.format\_map( {'name':'alex','year':12} )) #用字典输入
12. print('ab23'.isalnum()) #判断是否全是英文字符和阿拉伯字符
13. print('abA'.isalpha()) #判断是否全是英文字符，包含大小写
14. print('1A'.isdecimal()) #判断是否是10进制的数值
15. print('1A'.isdigit()) #判断字符是否是整数
16. print('a 1A'.isidentifier()) #判读是不是一个合法的标识符，是否是合法的变量名
17. print('33A'.islower()) #判断是不是小写
18. print('33A'.isnumeric()) #判断是不是一个数字，只有数字不包括小数点
19. print('My Name Is '.istitle()) #判断是不是标题，要求首字母都是大写
20. print('My Name Is '.isprintable()) #tty file ,drive file 判断这些文件是否能打印
21. print('My Name Is '.isupper()) #判断是不是大写
22. print('+'.join( ['1','2','3']) ) #在每个字符里面加一个“+”号
23. print( name.ljust(50,'\*') ) #50个字符，左边剩下的用\*\*\*\*表示
24. print( name.rjust(50,'-') ) #50个字符，右边剩下的用—表示
25. print( 'Alex'.lower() )
26. print( 'Alex'.upper() )
27. print( '\nAlex'.lstrip() ) # 从左边去空格和回车
28. print( 'Alex\n'.rstrip() ) # 从右边去空格和回车
29. print( ' Alex\n'.strip() ) #去掉所有的回车和空格
30. p = str.maketrans("abcdefli",'123$@456') # 字符"abcdefli"和 字符'123$@456'之间相互转化有对应的关系
31. print("alex li".translate(p) )
32. print('alex li'.replace('l','L',1)) #替换，后面的表示替换的数量
33. print('alex lil'.rfind('l')) #找到最右边的值并且返回，output = 7
34. print('1+2+3+4'.split('\n')) #根据括号里面的表达式来分割字符串，括号里的表达式变成空格
35. print('1+2\n+3+4'.splitlines()) #依据换行符来切割字符
36. print('Alex Li'.swapcase()) #字符串的大小写都反转，“aLEX lI”
37. print('lex li'.title()) # 是单词首字母大写 “Lex Li”
38. print('lex li'.zfill(50)) #用零来填充
39. print( '---')

## 2、for循环

**for循环中，用continue的话会先跳出本次循环，继续下一次循环，用break的话，就直接终止循环了。**

3、readlines()会将文件的所有的行都读取到内存里面

# 集合

## 1、 集合的运用

1. \_\_author\_\_ = "Alex Li"
2. list\_1 = [1,4,5,7,3,6,7,9]
3. list\_1 = set(list\_1) #变成集合之后，就没有重复的数据了，list\_1 = [1,4,5,3,6,7,9] 用set来将列表变为集合
4. list\_1 = list(list\_1) # 又可以将set变成list
5. list\_2 =set([2,6,0,66,22,8,4])
6. print(list\_1,list\_2)
7. '''
8. #交集
9. print( list\_1.intersection(list\_2) ) #intersection是交叉的意思
10. #并集
11. print(list\_1.union(list\_2))
12. #差集 in list\_1 but not in list\_2
13. print(list\_1.difference(list\_2))
14. print(list\_2.difference(list\_1))
15. #子集
16. list\_3 = set([1,3,7])
17. print(list\_3.issubset(list\_1))
18. print(list\_1.issuperset(list\_3))
19. #对称差集
20. print(list\_1.symmetric\_difference(list\_2))
21. print("-------------")
22. list\_4 = set([5,6,7,8])
23. print(list\_3.isdisjoint(list\_4)) # Return True if two sets have a null intersection.
24. '''
25. #交集
26. print(list\_1 & list\_2)
27. #union
28. print(list\_2 | list\_1)
29. #difference
30. print(list\_1 - list\_2) # in list 1 but not in list 2
31. #对称差集
32. print(list\_1 ^ list\_2)
33. list\_1.add(999) #添加一项
34. list\_1.update([888,777,555]) #添加多项
35. print(list\_1)
36. print(list\_1.pop())
37. print(list\_1.pop())
38. print(list\_1.pop())
39. print(list\_1.pop())
40. print( list\_1.discard(888) )
41. x in a #测试x是否在a中，列表的测试也是如此

# 字典

## 1、字典的应用

（1）字典是无序的，字典无下标，因不需要定位，通过key来取数

**av\_catalog = {**

**"欧美":{**

**"www.youporn.com": ["很多免费的,世界最大的","质量一般"],**

**"www.pornhub.com": ["很多免费的,也很大","质量比yourporn高点"],**

**"letmedothistoyou.com": ["多是自拍,高质量图片很多","资源不多,更新慢"],**

**"x-art.com":["质量很高,真的很高","全部收费,屌比请绕过"]**

**},**

**"日韩":{**

**"tokyo-hot":["质量怎样不清楚,个人已经不喜欢日韩范了","听说是收费的"]**

**},**

**"大陆":{**

**"1024":["全部免费,真好,好人一生平安","服务器在国外,慢"]**

**}**

**}**

**av\_catalog["大陆"]["1024"][1] = "可以在国内做镜像"**

**av\_catalog.setdefault("大陆",{"www.baidu.com":[1,2]})** #取大陆的这个key的值，如果能取到则返回，取不到就创建对应的key，值为后面的{"www.baidu.com":[1,2]}

**print(av\_catalog)**

**info = {**

**'stu1101': "TengLan Wu",**

**'stu1102': "LongZe Luola",**

**'stu1103': "XiaoZe Maliya",**

**}**

**b ={**

**'stu1101': "Alex",**

**1:3,**

**2:5**

**}**

**info.update(b)** # 交叉更新，有相应的值就直接替代，没有就重新创建，输出如下：



**print(info )**

**c = dict.fromkeys([6,7,8],’test’])** #新建了一个字典，输出如下：



**c = dict.fromkeys([6,7,8],[1,{"name":"alex"},444])** **#三个key共享了一个内存地址，改一个key值，其他的都会变**



**print(c )**

**c[7][1]['name'] = "Jack Chen"**

**print(c)'''**

**#print(info.items() )**

**info['stu1104']**

**print(info.get('stu1103'))**  #获取字典是否有相应的key，如果存在则返回对应的值，不存在则返回none

**print('stu1103' in info) #info.has\_key("1103") in py2.x**

**'''**

**#print(info["stu1101"])**

**info["stu1101"] ="武藤兰"**

**info["stu1104"] ="CangJingkong"**

**#del**

**del info["stu1101"]**  #删除字典的内容

**info.pop("stu1101")**  #删除指定内容

**info.popitem()**  #随便删除一个

**print(info)**

'''

**info = {**

**'stu1101': "TengLan Wu",**

**'stu1102': "LongZe Luola",**

**'stu1103': "XiaoZe Maliya",**

**}**

**for i in info:**

**print(i,info[i])**

**for k,v in info.items():** #k和v分别代表了键和值

**print(k,v)**

# 文件的操作

**'''  
data = open("yesterday",encoding="utf-8").read() #命令用UTF-8的格式来打开文件  
f = open("yesterday2",'a',encoding="utf-8") #文件句柄  
#a = append 追加  
  
f.write("\nwhen i was young i listen to the radio\n")  
data = f.read()  
print('--read',data)  
  
f.close()  
'''  
f.readlines() #生成的是一个列表**

**f.readline() #读取当前的行**f = open("yesterday",'w',encoding="utf-8") **#用w是写的模式，用写的话会自动创建新的文件，如果同名的文件存在，那么新的文件将会将旧的文件覆盖，注意别轻易用w。***#f = open("yesterday2",'r+',encoding="utf-8") #文件句柄* ***读写*** *#f = open("yesterday2",'w+',encoding="utf-8") #文件句柄* ***写读*** *#f = open("yesterday2",'a+',encoding="utf-8") #文件句柄 追加读写***f = open("yesterday2",'wb')** *#文件句柄 二进制文件***f.write("hello binary\n".encode())  
f.close()**  
  
**'''  
  
print(f.encoding)  
f.tell() #定位句柄的位置，定位的是读到多少个字了**

**f.reand(n) #表示读之后的n个字符**

**f.seek(n) #让句柄回到第n个字符**

**f.encoding #返回文件的编码格式  
print(f.flush()) #将内存里面的内容，写进硬盘里面  
print(dir(f.buffer))**

**#high bige，下面的方法实现一行行的读，每次内存里面只占用一行  
count = 0  
for line in f:  
 if count == 9:  
 print('----我是分割线----------')  
 count += 1  
 continue  
 print(line)  
 count +=1  
  
#low loop  
  
for index,line in enumerate(f.readlines()): #enumerate取出列表的对应下表  
 if index == 9:  
 print('----我是分割线----------')  
 continue  
 print(line.strip())  
#for i in range(5):  
# print(f.readline())  
'''**

**for** i **in** range(30):  
 sys.stdout.write(**"#"**)  
 sys.stdout.flush() *#及时显示出来* time.sleep(0.1)

f = open(**"yestreday2"**,**"r"**,encoding=**"utf-8"**)  
f\_new = open(**"yesterday.bac"**,**"w"**,encoding=**"utf-8"**)  
*#修改原文件的一段话，生成一个新的文件来保存原来的修改***for** line **in** f:  
 **if "肆意的快乐等我享受" in** line:  
 line = line.replace(**"肆意的快乐等我享受"**,**"肆意的快乐等Bling享受"**)  
 f.write(line)  
f.close()  
f\_new.close()  
*#可以用with管理文件，with可以打开多个文件***with** open(**"log1"**) **as** obj1,open(**"log2"**) **as** obj2:  
 **pass**

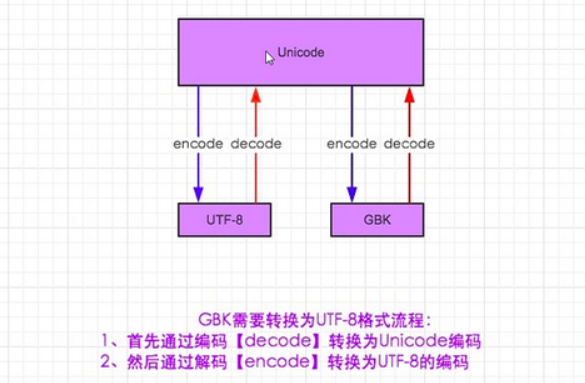
# 字符编码转换详解

Unicode 都是占两个字节，也就是16位，能够存放2^16=65536个字符；

ASCII 码占一个字节，也就是8位，能够存2^8 = 255，英文用的是ASCII码；

Utf-8是一种可以变的编码，英文字符用一个字节，中文字符占3个字节，占用更多的空间。

**Unicode是所有编码的父集，其他编码都可以转换为Unicode？？转换方式如下：**



**Import sys**

**Sys.getdefaultencoding()** *#获取系统默认的编码*

## 1、以下这是在python2中的转码方式

**import** sys  
print(sys.getdefaultencoding())  
\_\_author\_\_ = **"Alex Li"***#uncode编码打印的时候，如果是元组，那么显示的是编码*s= **"你好"**s\_to\_unicode = s.decode(**"utf-8"**) *#将utf-8转换为unicode*s\_to\_gbk = s\_to\_unicode.encode(**"gbk"**) *#将unicode转换为utf-8*gbk\_to\_utf8 = s\_gbk.decode(**"gbk"**).encode(**"utf-8"**) *#将gbk先用decode解码为unicode，然后再编码为utf—8  
  
#unicode 和utf-8之间是可以直接打印的*s1 = **u"你好"** *#这样写，那么s1就表明是用unicode编码了*

## 2、在python3里面的转码

**第一种情况：**

*#-\*-coding:gbk-\*-***import** sys  
print(sys.getdefaultencoding())  
\_\_author\_\_ = **"Alex Li"**s = **"你哈"** *#s是unicode编码，python程序默认编码是unicode，所以s.decode 没法用，因为s已经是unicode编码了*print(s.encode(**"gbk"**).decode(**"gbk"**)) *#这样就能显示中文“你哈”*print(s.encode(**"utf-8"**))  
print(s.encode(**"utf-8"**).decode(**"utf-8"**).encode(**"gb2312"**))

**第二种情况：**

**import** sys  
print(sys.getdefaultencoding())  
\_\_author\_\_ = **"Alex Li"**s = **"你好"**s\_gbk = s.encode(**"gbk"**) *#encode会将原来的字符转换为字节bytes类型*print(s\_gbk)  
print(s.encode()) *#将字符转换为bytes类型*gbk\_to\_utf8 = s\_gbk.decode(**"gbk"**).encode(**"utf-8"**) *#decode中的括号是告诉程序字符类型*print(**"utf8"**,gbk\_to\_utf8)

# 函数

函数和过程的区别，过程只是没有返回值的函数，但是在python中，隐形的给过程返回None值，两者的区别如下：

*#函数***def** func1():  
 *"""testing1"""* print(**'in the func1'**)  
 **return** 0  
*#过程***def** func2():  
 *'''testing2'''* print(**'in the func2'**)

x=func1()  
y=func2()

print(**'from func1 return is %s'** %x)

print(**'from func2 return is %s'** %y)

**def** test2():  
 print(**'in the test2'**)  
 **return** 0  
**def** test3():  
 print(**'in the test3'**)  
 *#return 1,'hello',['alex','wupeiqi'],{'name':'alex'}* **return** test2 *#返回的test2的内存地址*

**def** test(x,y,z):  
 print(x)  
 print(y)  
 print(z)  
注意：关键参数不能写在位置参数前面  
*# test(y=2,x=1) #与形参顺序无关  
# test(1,2) #与形参一一对应  
#test(x=2,3) #这种方式函数没法运行*

*#test(2,x=3) #这种方式函数是可以运行，包括位置调用和关键字调用*test(3,z=2,y=6)

**def** conn(host,port=3306):  
 **pass***#默认参数特点：调用函数的时候，默认参数非必须传递，也可以传递值给默认参数*

## 1、关键词参数和字典参数的运用

\_\_author\_\_ = **"Alex Li"***#\*args:接受N个位置参数，转换成元组形式，针对实参不固定的情况，只要以“\*”号开头就行，后面随便命名***def** test(\*args):  
 **print**(args)  
  
test(1,2,3,4,5,5)  
test(\*[1,2,4,5,5]) *# args=tuple([1,2,3,4,5])  
  
#位置函数和任意函数一起使用***def** test1(x,\*args):  
 **print**(x)  
 **print**(args)  
  
test1(1,2,3,4,5,6,7)  
  
  
*#\*\*kwargs：接受N个关键字参数，转换成字典的方式***def** test2(\*\*kwargs):  
 **print**(kwargs)  
 **print**(kwargs[**'name'**])  
 **print**(kwargs[**'age'**])  
 **print**(kwargs[**'sex'**])  
  
test2(name=**'alex'**,age=8,sex=**'F'**)  
test2(\*\*{**'name'**:**'alex'**,**'age'**:8}) *#此种方式也是可以的，输出字典，函数内部也可以调用字典的各个值***def** test3(name,\*\*kwargs):  
 **print**(name)  
 **print**(kwargs)  
  
*# test3('alex',{"age":18,"sex":'m'}) #这种方式是没法用的，必须是age=18，sex=m 这种形式*test3(**'alex'**,\*\*{**"age"**:18,**"sex"**:**'m'**}) *#这种方式可以用*位置参数、默认参数、字典参数可以一起使用，字典参数可以不指定  
**def** test4(name,age=18,\*\*kwargs):  
 **print**(name)  
 **print**(age)  
 **print**(kwargs)  
  
test4(**'alex'**,sex=**'m'**,hobby=**'tesla'**,age=3) *#给默认参数赋值可以用位置参数形式赋值，也可以关键字参数赋值，如age=3***def** test4(name,age=18,\*args,\*\*kwargs):  
 **print**(name)  
 **print**(age)  
 **print**(args)  
 **print**(kwargs)  
 logger(**"TEST4"**)  
  
*#\*args接受的是位置参数***def** logger(source):  
 **print**(**"from %s"** % source)  
*#*test4(**'alex'**,age=34,sex=**'m'**,hobby=**'tesla'**) *#test4的调用需要放在logger函数后面，因test4函数中调用了logger函数*

## 2、局部变量和全局变量

在子程序中定义的变量称为局部变量，在程序的一开始定义的变量称为全局变量。

全局变量作用域是整个程序，局部变量作用域是定义该变量的子程序。

当全局变量与局部变量同名时：

在定义局部变量的子程序内，局部变量起作用；在其它地方全局变量起作用。

以下方式也是可以将局部变量输出到全局变量**（最好不要这样做），不应该在函数里面改全局变量，因为外面没有定义该全局变量**：

**def** change\_name():  
 **global** name  
 name =**"bling"**change\_name()  
**print**(name)

>>>bling

school = **"Oldboy edu."**names = [**"Alex"**,**"Jack"**,**"Rain"**] *# 字符串和整体型的数据等全局变量，在不加global时没法在函数里面修改  
#但是比如列表、字典、集合和类等的全局变量类型在不加global的时候函数也能修改*names\_tuple = (1,2,3,4)  
**def** change\_name():  
 names[0] = **"金角大王"  
 print**(**"inside func"**,names)  
change\_name()  
**print**(names)  
  
  
**def** change\_name(name):  
 **global** school *#将局部变量school变成全局变量，这样函数能够改变全局变量的值* **print**(**"before change"**,name,school)  
 school = **"Mage Linux"** name =**"Alex li"** *#这个函数就是这个变量的作用域，全局变量里面的name并没有发生改变  
 # age =23* **print**(**"after change"**,name,school)

## 3、递归

在函数内部，可以调用其他函数。如果一个函数在内部调用自身本身，这个函数就是递归函数。如下：

**def** calc(n):  
 **print**(n)  
 **if** int(n/2) >0: #必须有一个明确的结束条件  
 **return** calc( int(n/2) )  
 **print**(**"->"**,n)  
calc(10)

1. 必须有一个明确的结束条件

2. 每次进入更深一层递归时，问题规模相比上次递归都应有所减少

3. 递归效率不高，递归层次过多会导致栈溢出（在计算机中，函数调用是通过栈（stack）这种数据结构实现的，每当进入一个函数调用，栈就会加一层栈帧，每当函数返回，栈就会减一层栈帧。由于栈的大小不是无限的，所以，递归调用的次数过多，会导致栈溢出）

## 4、高阶函数

包含了多个阶的函数

**def** add(a,b,f):   
 **return** f(a)+f(b)  
  
res = add(3,-6,abs) #其中f为传入函数，求绝对值的函数abs  
**print**(res)

>>> 9

# 装饰器

定义：装饰器，本质上是函数（装饰其他函数），为其他函数添加附加功能

原则：

1、不能修改被装饰的函数源代码

2、不能修改被装饰的函数的调用方式

装饰器对待被装饰的函数完全是透明的，原函数的功能能够全部正常运行。

实现装饰器知识储备：

1.函数即“变量”

2.高阶函数

A、把一个函数名当做实参传给另外一个函数（在不修改被装饰函数的情况下为其添加功能）

B、返回值中包含函数名（不修改函数的调用方式 ）

3.嵌套函数

博客链接：<http://egon09.blog.51cto.com/9161406/1836763>

**装饰器的示范**：

**import** time  
**def** timer(func): *#timer(test1) func=test1* **def** deco(\*args,\*\*kwargs): *#能够传递任意数量的参数，满足调用test1无参数和test2两个参数*  
 start\_time=time.time()  
 func(\*args,\*\*kwargs) *#run test1()* stop\_time = time.time()  
 print(**"the func run time is %s"** %(stop\_time-start\_time))  
 **return** deco  
@timer *#引用装饰器相当于运行test1=timer(test1)***def** test1():  
 time.sleep(1)  
 print(**'in the test1'**)  
  
@timer *# test2 = timer(test2) = deco test2(name) =deco(name)***def** test2(name,age):  
 time.sleep(2)  
 print(**"test2:"**,name,age)  
  
test1()  
test2(**"alex"**,22)

# 迭代器与生成器

列表生成器 [ i\*2 for i in range(10) ]，生成列表[0, 2, 4, 6…..]



**下面这个就是生成器，generator**



1、生成器只有在调用的时候才会生成相应的数据；

2、只记录当前位置；

3、只有一个\_\_next\_\_()方法；

## 生成器并行

**import** time  
**def** consumer(name):  
 **print**(**"%s 准备吃包子啦!"** %name)  
 **while** True:  
 baozi = **yield  
  
 print**(**"包子[%s]来了,被[%s]吃了!"** %(baozi,name))  
  
*# c = consumer("ChenRonghua")  
# c.\_\_next\_\_()  
# c.\_\_next\_\_()  
  
# b1= "韭菜馅"  
# b2 ="玉米猪肉馅"  
# c.\_\_next\_\_()  
# c.send(b1) #send中的值被yield接受了，从而给baozi赋值“韭菜馅”，同时调用了yield，后面的语句继续运行  
# c.send(b2) #send中的值被yield接受了，从而给baozi赋值“玉米猪肉馅”，同时调用了yield，后面的语句继续运行  
# c.\_\_next\_\_()***def** producer(name):  
 c = consumer(**'A'**)  
 c2 = consumer(**'B'**)  
 c.\_\_next\_\_() *#只有加上这句，才能使生成器运行到baozi=yield这段语句，然后暂停返回*  
 c2.\_\_next\_\_()  
 **print**(**"老子开始准备做包子啦!"**)  
 **for** i **in** range(10):  
 time.sleep(1)  
 **print**(**"做了1个包子,分两半!"**)  
 c.send(i) *#send可以传值给yield，然后进一步将值传给baozi* c2.send(i)  
producer(**"alex"**)

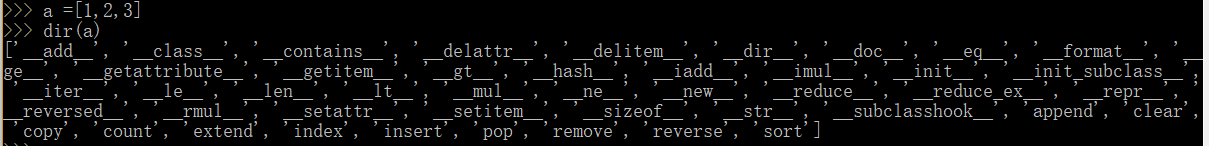
输出如下：



## 迭代器

我们已经知道，可以直接作用于for循环的数据类型有以下几种：一类是集合数据类型，如list、tuple、dict、set、str等；一类是generator，包括生成器和带yield的generator function。这些可以直接作用于for循环的对象统称为可迭代对象：Iterable。可以使用isinstance()判断一个对象是否是Iterable对象用法如：isinstance([],Iterable)；**\*可以被next()函数调用并不断返回下一个值的对象称为迭代器：Iterator，生成器一定是一个迭代器**。

用dir命名可以查看一个对象可以调用的所有的方法，可以看到列表a没有next()方法，因此a不是迭代器。



把list、dict、str等Iterable变成Iterator可以使用iter()函数：

**小结**

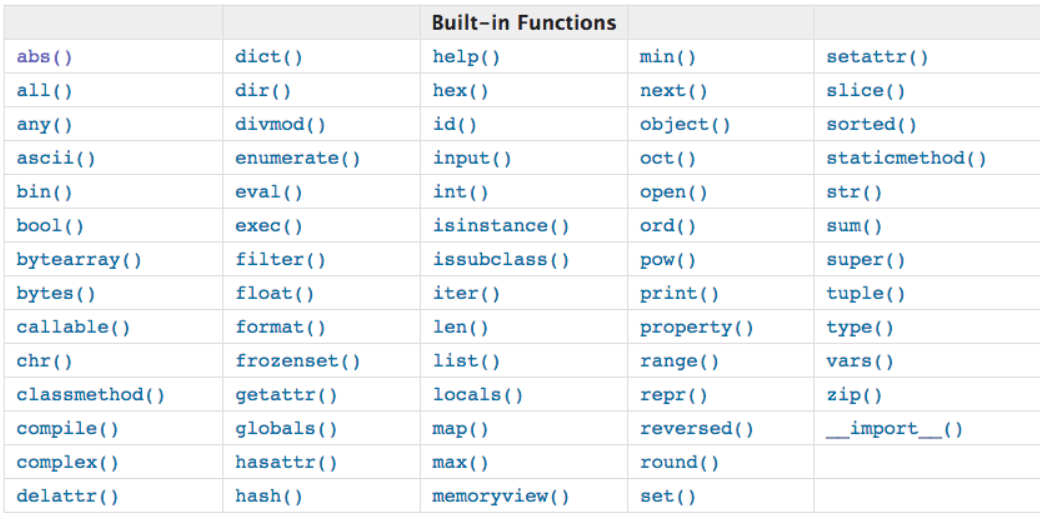
凡是可作用于for循环的对象都是Iterable类型；

凡是可作用于next()函数的对象都是Iterator类型，它们表示一个惰性计算的序列；

集合数据类型如list、dict、str等是Iterable但不是Iterator，不过可以通过iter()函数获得一个Iterator对象。

# Python的内置方法

Python有如下的内置方法：



**def** all(iterable):

**for** element **in** iterable:

**if** **not** element:

**return** **False**

**return** **True**

**def** any(iterable):

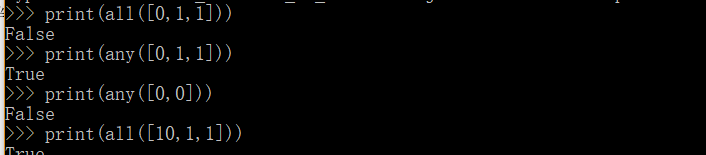
**for** element **in** iterable:

**if** element:

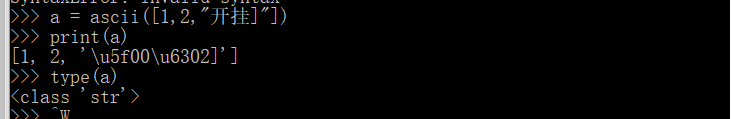
**return** **True**

**return** **False**

判断可迭代参数里面所包含的值是否都为真，如：



ascii的应用



bin(x)

将一个整数转换为2进制：

**>>>** bin(3)

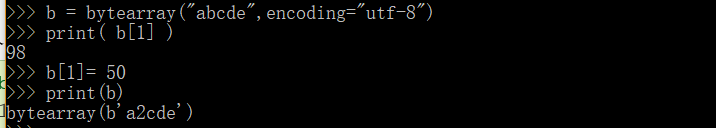
'0b11'

**>>>** bin(-10)

'-0b1010'

boo1（）布尔函数，判断真假

bytearray([*source*[, *encoding*[, *errors*]]])



**callable(object)**

判断目标是否能够调用

Chr（）根据括号里面的数字，返回对应的unicode字符，ord()刚好和chr()反过来。

