## 编程错误

**IndentationError: unindent does not match any outer indentation level**错误表明，你使用的缩进方式不一致，有的是 tab 键缩进，有的是空格缩进，改为一致即可。

ValueError: unsupported format character ':' (0x3a) at index 3

print("%d%:%d:%d" %(h, m, s))

%d%: %d:%

IndentationError: unindent does not match any outer indentation level（对齐）

## Python概述



特点：易于学习、易于阅读、易于维护、一个广泛的数据库、互动模式、可移植、可扩展、数据库、GUI编程、可嵌入

缺点：运行速度快、代码不能加密

### 1.1数据的存储

思考：为什么使用计算机？

答：存储数据，计算数据。

思考：数据存在哪？

答：数据存储在内存里

思考：数据怎么在内存里存储？

怎么存储数据？

10——1010（2）0000 0000 0000 0000 0000 0000 0000 1010

计算机存储数据，先开辟内存空间，再存储数据。计算机开辟内存的最小单位是字节。在存储数据时，用最高位标识符号，1表示负数，0表示正数

内存：内存是计算机中最重要的部件之一，它是与GPU进行沟通的桥梁。

（抽象理解）一个开光，有两种状态，开启和关闭。一种状态对应1，另一种状态对应0。把8个开关放在一个房间里面，这个房间称为”一个字节”，一个开关代表“一位”。每个房间都有门牌号，看做“地址”。把无数个房间堆叠起来组成摩天大夏，可以把摩天大夏看成“内存”。

内存中以二进制形式存储数据。

二进制： 0 1 逢二进一 1+ 1 = 10

八进制：0 1 2 3 4 5 6 7 逢八进一 1 + 7 = 10

十进制：0 1 2 3 4 5 6 7 8 9 逢十进一 1 + 9 = 10

十六进制：0 1 2 3 4 5 6 7 8 9 ~~10~~ a b c d e f 逢十六进一 1 + f = 10

十进制—》二进制

倒除法、余数逆序

10（10）-> 1010（2）

二进制—》十进制

当前的数字，乘以2的位数次方，最后相加

1010（2）->10（10）

0 \* 2^0 + 1 \* 2^1 + 0 \* 2^2 + 1 \* 2^3 = 0 + 2 + 0 +8 = 10

八进制—》二进制

【一转三位】八进制的一位相当于二进制的三位，转换时按照十进制转换

65（8）->~~0110 0101~~  110101）（2） （8 4 2 1）

十六进制—》二进制

【一转四位】八进制的一位相当于二进制的四位，转换时按照十进制转换

a4 (16) -> 10100100 (2) （16 8 4 2 1）

二进制—》八进制

【三位一取】从低位开始取，高位不够取0

1 111 010（2） -> 172（8）

二进制—》十六进制

【四位一取】从低位开始取，高位不够取0

1bit

8bit == 一字节

1024字节 == 1K

1024K == 1M

1024M == 1G

1024G == 1T

32位操作系统，寻址空间比较小，最大匹配4G内存；

64位操作系统，寻址空间很大。

原码，反码、补码

原码：规定看字节数，写明了符号位，就得到了数据的原码（二进制）

-3——10000011

反码：正数的反码是其原码，负数的反码是其原码的符号位不动，其他位取反

-3——10000011——11111100

补码：正数的补码是其原码，负数的补码是将其原码除符号位之外的各位取反之后在末位再加1

-3——10000011——11111100——11111101

绝对路径：从根目录开始链接的目录 E:\软件学习\Python\python100天\day02

相对路径：不从根目录开始链接的目录 Python\python100天\day02

命令行模式

dir:查看当前目录下的所有文件

cd: +路径 跳转到指定目录

cd+ .. .. 代表的是上一级目录

### 1.2print&input与变量和运算符

# 注释一行

“””

注释多行

”””

多行添加注释 ctrl + ‘/’

多行消除注释 ctrl + ‘/’

#### print输出

print('today is a good day', 'a sunny day', 'a bright day')  
print('10 + 8 = ', 18)  
n = 10  
sum = 15  
print('1 到 %d 之和为: %d' % (n, sum))

#### input输入

#从外部获取变量的值  
#等待输入，输入的内容保存到age里  
age = input('请输入一个数：')  
print('age = ', age)

****

#### 关键字

import keyword  
print(keyword.kwlist)

#程序执行的过程：自上而下顺序执行（自上而下，逐步求精）

#### 变量和常量

age = 18  
print("age = ", age)  
#del age  
#查看变量类型  
print(type(age))  
#查看变量地址  
print(id(age))

****

#### 数字类型

*"""  
分类：整数、浮点数、复数  
整数：python可以处理任意大小的整数，包括负数，在程序中的表示和数学的写法一样  
浮点数：浮点数由整数部分和小数部分组成，浮点数运算可能会有四舍五入的误差  
"""*num1 = 10  
num2 = num1  
# 连续定义多个变量  
number3 = number4 = number5 = 1  
print(number3, number4, number5)  
# 交互式赋值定义变量  
number6, number7 = 2, 3  
print(number6, number7)  
  
# 浮点数：浮点数由整数部分和小数部分组成，浮点数运算可能会有四舍五入的误差  
f1, f2 = 1.1 , 2.2  
print(f1 + f2) #3.3000000000000003  
  
#复数：实数和虚数部分组成  
  
#数据类型转换  
print(int(1.9))  
print(float(1))  
print(int('123'))  
print(int('123abc')) #ValueError: invalid literal for int() with base 10: '123abc'

#### 数字函数

#返回数字的绝对值 abs()

#比较两个数大小 print((6<9) – (9 >6)) -1

#返回给定参数的最大值 print(max(1, 2, 3 ,5)) 5

#返回给定参数的最小值 print(min(1, 2, 3 ,5)) 5

#求x的y次方 2^5 print(pow(2, 5)) 32

#round(x[, n]) 返回浮点数x的四舍五入值，如果给出n值，则代表舍入到小数点后的n位

print(round(3.456)) 3

print(round(3.556)) 4

print(round(3.556, 2)) 3.46

#导入库

库：封装一些功能

import math

print(math.ceil(18.1)) 19 向上取整

print(math.floor(18.9)) 18 向下取整

print(math.modf(22.3)) (0.3000000000007, 22.0) 返回整数部分和小数部分（浮点数）

print(math.sqrt(16)) 4 开方

#### 随机数

import random

法一：从序列的元素中随机挑选一个元素

print(random.choice[1, 3, 5, 7 ,9]) 5

print(random.choice[19，‘aa’]) aa

print(random.choice(range(5))) #range(5) == [0, 1, 2 ,3 , 4]

print(random.choice(“sunck”))) #”sunck” == [“s”, “u”, “n”, “c”, “k”]

#生成一个1-100之间的随机数

r1 = random.choice(range(100)) +1 54

法二：random.randrange([]start, stop, [,step])

#start:指定范围的开始值，包含到范围内，默认是0

#stop:指定范围的结束值，不包含到范围内

#step:指定的递增基数，默认是1

print(random.randrange(1, 100, 2))

法三：random.random()

随机生成[ 0, 1)之间的数（ 浮点数）

print(random.random ()) 0.368468415154

法四：random.shuffle()

#将序列的所有元素随机排序

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

print(random.shuffle(list)) 2, 5 ,1, 4, 3

#随机生成一个实数，在[3, 9]范围内

print(random.uniform(3, 9))

#### 运算符与表达式

表达式：由变量、常量和运算符的式子

阅读表达式：1.功能； 2.值

算术运算符（+ - \* / %（取模） \*\*（求幂pow(a, b)） //（取整））

算术运算表达式（1 + 2 2 \* 3 a / 3）

功能：进行相关符号的数学运算，不会改变变量的值

值：相关的数学运算的结果

**赋值运算符个赋值运算表达式**

赋值运算符 =

赋值运算表达式

格式： 变量 = 表达式

功能：计算了等号右侧“表达式”的值，并赋值给等号左侧的变量

值：赋值结束后变量的值

**复合运算符**

**+= -= \*= /= %= \*\*= ……**

a = a + b a = a – b a = a \* b

**if语句**

格式：

if 表达式：

语句

逻辑：当程序执行到if语句时，首先计算’表达式’的值，如果‘表达式’的值为真，那么执行if下的‘语句’，如果‘表达式’的值为假，则跳过整个if语句，继续向下进行。

假： 0 0.0 ‘’（空字符串） None False

其余为真

**if-else语句**

格式：

if 表达式：

语句1

else 表达式：

语句2

逻辑：当程序执行到if-else语句时，首先计算’表达式’的值，如果‘表达式’的值为真，那么执行‘语句1’,跳出整个if-else语句。如果‘表达式’的值为假，则执行‘语句2’，继续向下进行。

### 1.3字符串与循环中的while

#### 运算符

*"""  
按位与运算符（&）：按位运算符是把数字看做二进制数来进行计算,相应的位数都是1，则该位的结果位1，否则为0  
"""*print(5 & 5) #5  
print(5 & 6) #4  
"""  
按位或运算符（|）：两个二进制有一个为1时，结果为1   
"""  
print(5 | 5) #5  
print(5 | 6) #7  
"""   
按位异或运算符（^）：两个二进制相异时，结果为1   
"""  
print(5 ^ 5) #0  
print(5 ^ 6) #3  
"""  
按位取反运算符（~）：两个二进制相异时，结果为1   
"""  
print(~ 5) #-6  
print(~ 6) #-7  
  
"""  
<< 左移动运算符  
各二进制全部左移动若干位，由<<右侧的数字决定，高位丢弃，低位补0  
"""  
print(2 << 3) #16  
"""  
>> 左移动运算符  
各二进制全部左移动若干位，由>>右侧的数字决定，高位丢弃，低位补0  
"""  
print(13 >> 2) #3  
print(-13 >> 2) #-4  
"""  
关系运算符和关系运算表达式  
== != > < >= <=  
关系运算表达式：  
格式： 表达式1 运算表达式 表达式2  
功能： 计算“表达式1” 和“表达式2” 的值  
值：如果关系成立，整个关系运算表达式的值为真，否则为假  
"""  
  
"""  
逻辑运算符： “与” “或” “非”  
逻辑“与”——“and”  
逻辑与运算表达式： 表达式1 and 表达式2  
值：如果表达式1的值为真，表达式2的值也为真，整体逻辑表达式的值为真，，其余情况为假。  
  
逻辑“或”——“or”：  
逻辑或运算表达： 表达式1 or 表达式2  
值：有一个值为真，整体逻辑为真  
  
逻辑“非”——“not”：  
逻辑非运算表达： not 表达式  
值：如果表达式值为真，整体为假；如果表达式值为假，整体为真  
"""  
  
"""  
成员运算符：  
in ： 如果在指定的序列中找到值，返回True，否则返回False  
not in ：如果在指定的序列中没有找到值，返回True，否则返回False  
"""  
  
"""  
身份运算符  
is: 判断两个标识符是不是引用同一个对象  
is not ： 判断两个标识符是不是引用不同的对象  
"""  
  
"""  
运算符优先级  
\*\*   
~ + - （正负号）  
\* / % //  
+ -   
按位移动 >> <<   
按位与 &  
^ |  
<= < > >=  
== !=  
= %= += -= //=  
is is not  
in not in  
not or and   
"""  
  
""""  
短路原则  
表达式1 and 表达式2 and 表达式3 and …… 表达式n  
表达式1 or 表达式2 or 表达式3 or …… 表达式n  
"""

#### 字符串

*"""  
什么是字符串：以单引号或者双引号括起来的任意文本  
‘abc’ "abc"  
"""*# 创建字符串  
str1 = 'abcde'  
str2 = 'def'  
#字符串的运算：  
#字符串的连接：  
str3 = str1 + str2  
print("str3 = ", str3) # str3 = abcdedef  
#字符串的乘法：  
str4 = str1 \* 3  
print("str4 = ", str4) # str4 = abcdeabcdeabcde  
#访问字符串中的某一个字符：通过索引下标，索引从0开始  
print(str1[1]) # b  
#截取字符串中的一部分： 总给定下标1开始，截取到给定下标2之前  
str5 = str1[1: 5]  
print("str5 = ", str5) # str5 = bcde  
  
# in  
if 'a' in str1:  
 print('yes') # yes  
  
#格式化输出  
print("123")  
num = 10  
num2 = 10.2  
print("num = ", num)  
print("num = %d, str6 = %s, num3 = %.2f" %(num, str2, num2))  
# num = 10, str6 = def, num3 = 10.20  
# %d %s %f 占位符，后面的值未定  
  
# 转义字符：将一些字符转换成有特殊含义的字符  
# \n(换行符)  
# \' \"  
# \t 制表符（四个空格）  
# r 表示内部的字符串默认不转义  
print("num = ", num)  
print("E:\软件学习\Python")  
  
"""  
eval()   
功能：将字符串str当成有效的表达式来求值并返回计算结果  
"""  
print(eval("12 + 3"))  
  
"""  
len():返回字符串的长度(字符个数)  
"""  
print(len("123 str as"))  
  
"""  
lower(): 将字符串中的大写字母变为小写字母  
"""  
str20 = " A Good Man"  
print(str20.lower())  
  
"""  
upper():将字符串中的小写字母变为大写字母  
"""  
print(str20.upper())  
  
"""  
swapcase():转换字符串中大写字母为小写，小写字母为大写  
"""  
print(str20.swapcase())  
  
"""  
capitalize():首字母大写，其他小写  
"""  
print(str20.capitalize())  
  
"""  
title():每个单词的首字母大写  
"""  
print(str20.title())  
  
"""  
center(width, fillchar):返回一个指定宽度，width居中的字符串为填充的字符  
"""  
print(str20.center(20, "\*")) #\*\*\*\* A Good Man\*\*\*\*\*

#### while语句

*"""  
while语句：  
while 表达式：  
 语句  
  
逻辑：当程序执行到while语句时，首先计算“表达式”的值，如果“表达式”的值为假，则结束整个while语句。  
 如果为真，则循环执行“语句”，直到“表达式”为假。  
"""*# 计算1……100的和 （shift+6 打印省略号）  
num = 1  
result = 0  
while num < 101:  
 result += num  
 num += 1  
print("result = %d" %(result))  
  
# 打印每个字符  
str1 = "ABC abc"  
index = 0  
while index < len(str1):  
 print("str1[%d] = %s" % (index, str1[index]))  
 index += 1

### 1.4布尔&list与条件循环语句与trutle

#### 布尔值

*"""  
布尔值：一个布尔值只有True和False  
空值：是python中一个特殊的值，用None表示。None不能理解为0，因为0是有意义的而None是一个特殊的值  
"""*

#### 列表

*"""  
列表：一种有序的集合  
创建列表： 列表名 = [列表选项1，列表选项2，……，创建列表n ]  
  
"""*#创建一个空列表  
list1 = []  
#创建带有元素的列表  
list2 = [11, 12, 13, 14, 15]  
index = 0  
sum = 0  
while index < 5:  
 sum += list2[index]  
 index += 1  
print("平均值 = %d" % (sum / 5))  
#注意：列表中的元素数据可以是不同类型  
#列表元素的访问,注意不要越界（下标超出可取的范围）  
#取值： 格式 ： 列表名[下标]  
list3 = [1, 2, 3]  
print(list3[2])  
"""  
列表操作：  
列表组合：list4 = list2 + list3  
列表的重复： list5 = list \* 4  
判断元素是否在列表中 print(3 in list[5])  
列表截取 list5[2:4] list[:3] list[2:]  
  
二维列表 list6 = [[1, 2],[3, 4],[5, 6]]  
print(list6[1]) # [1, 2]  
print(list6[0][1]) # 2  
"""  
  
"""  
列表方法：  
#append: 在列表中末尾添加新的元素  
list7 = [1, 2, 3, 4, 5]  
list7.append(6)  
print(list7) #[1, 2, 3, 4, 5, 6]  
list7.append([7, 8, 9])  
print(list7) #[1, 2, 3, 4, 5, 6, [7, 8, 9]]  
#extend 在末尾一次性追加另一个列表中的多个值  
list8 = [1, 2, 3, 4, 5]  
list8.extend([6, 7, 8])  
print(list8) #[1, 2, 3, 4, 5, 6, 7, 8]  
#insert 在下标处添加一个元素，不覆盖原数据，原数据向后顺延  
list9 = [1, 2, 3, 4, 5]  
list9.insert(1, 100)  
print(list9) #[1, 100, 2, 3, 4, 5]

list9.insert(1, [200, 300])  
print(list9) #[1, [200, 300], 100, 2, 3, 4, 5]  
#pop(x=list[-1]) 移除列表中指定下标中的元素（默认移除最后一个元素）  
list10 = [1, 2, 3, 4, 5]  
list10.pop()  
print(list10) #[1, 2, 3, 4]  
list10.pop(2)  
print(list10) #[1, 2, 4]  
#remove 移除列表中的某个元素第一个匹配的结果  
list11 = [1, 2, 3, 4, 5, 4]  
list11.remove(4)  
print(list11) #[1, 2, 3, 5, 4]  
#clear 清除列表中所有数据  
list12 = [1, 2, 3, 4, 5, 4]  
list12.clear()  
print(list12) #[]  
#index 从列表中找出某个值第一个匹配的索引值  
list13 = [1, 2, 3, 4, 5, 3]  
index13 = list13.index(3)   
#圈定范围 index(value, [start, [stop]])  
index14 = list13.index(3, 4, 7)  
print(index13, index14) #2 5  
#len 列表中元素个数  
list14 = [1, 2, 3, 4, 5, 3]  
print(len(list14)) #6  
#max 获取列表中的最大值  
list15 = [1, 2, 3, 4, 5, 3]  
print(max(list15)) # 5  
#max 获取列表中的最大值  
list16 = [1, 2, 3, 4, 5, 3]  
print(max(list16)) # 1  
#count 查看元素在列表中出现的次数  
list17 = [1, 2, 3, 4, 5, 3]  
print(list17.count(3)) #2  
  
"""

#reverse 倒序  
list18 = [1, 2, 3, 4, 5, 3]  
list18.reverse()  
print(list18) #[3, 5, 4, 3, 2, 1]  
#排序 sort  
list19 = [1, 2, 3, 4, 5, 3]  
list19.sort()  
print(list19) #[1, 2, 3, 3, 4, 5]  
#浅拷贝 引用拷贝  
list20 = [1, 2, 3, 4, 5, 3]  
list21 = list20  
list21[1] = 100  
print(list20) #[1, 100, 3, 4, 5, 3]  
print(id(list20)) #1973156821896  
print(list21) #[1, 100, 3, 4, 5, 3]  
print(id(list21)) #1973156821896  
#深拷贝 内存的拷贝  
list22 = [1, 2, 3, 4, 5, 3]  
list23 = list22.copy()  
list23[1] = 100  
print(list22) #[1, 2, 3, 4, 5, 3]  
print(id(list22)) #2126608710536  
print(list23) #[1, 100, 3, 4, 5, 3]  
print(id(list23)) #2126614417096  
#将元组转成列表  
list24 = list((1,2,3,4))  
print(list24) #[1, 2, 3, 4]

#### 条件控制语句

*"""  
if 语句  
if—else 语句  
if—elif—else 语句  
格式：  
if 表达式1：  
 语句1  
elif 表达式2：  
 语句2  
……  
if 表达式n：  
 语句n  
"""*

#### 例题1：计算 一串字符中有多少单词

str = input("请输入:")  
# sfds sf d sd d d  
str1 = str.strip()  
num = 0  
index = 0  
while index < len(str1):  
 while str1[index] != " ":  
 index += 1  
 if index == len(str1):  
 break  
 num += 1  
 if index == len(str1):  
 break  
 while str[index] == " ":  
 index += 1  
print(num) # 6

### 1.5元组&字符串&字典

#### 元组

#能用元组，就用元组，因为不可变  
# 创建空的元组  
tuple1 = ()  
print(tuple)  
#创建带有元素的元组，元组中的元素可以不同  
tuple2 = (1, 2, 3, "good", True)  
print(tuple2)  
#定义只有一个元素的元组  
tuple3 = (1,)  
print(tuple3)  
print(type(tuple3))  
#元组的访问 格式：元组名[] ,下标从0开始  
tuple4 = (1, 2, 3, 4, 5)  
print(tuple4[2])  
#获取最后一个元素  
print(tuple4[-1])  
  
#修改元组，元素不可变，元素里面的内容可变  
tuple4 = (1, 2, 3, 4, [5, 6, 7])  
#tuple4[0] = 100 # 报错，元组不能变  
tuple4[-1][0] = 50  
print(tuple4) #(1, 2, 3, 4, [50, 6, 7])  
#删除元组  
tuple5 = (1, 2, 3, 4, 5)  
del tuple5  
# print(tuple5) #NameError: name 'tuple5' is not defined  
  
#元组的操作  
#元组相加  
tuple6 = (1, 2, 3)  
tuple7 = (4, 5, 6)  
tuple8 = tuple6 + tuple7  
print(tuple8)  
#元组相乘  
print(tuple6 \* 3)  
#判断元素是否元组  
print(1 in tuple6) #True  
#元组的截取 格式 元组名[开始下标：结束下标]  
#从开始下标开始截取，截取到结束下标之前  
print(tuple8[2: 5]) #(3, 4, 5)  
print(tuple8[2:]) #(3, 4, 5, 6)  
print(tuple8[:5]) #(1, 2, 3, 4, 5)  
#二维元组:元素为一维元组的元组  
tuple9 = ((1, 2),(3, 4),(5, 6))  
print(tuple9[2][0]) #5  
  
#元组的方法  
tuple10 = (1, 2, 3, 4, 5)  
#len(tuple10) 返回元组中元素的个数  
#max(tuple10) 返回元组中的最大值  
#min(tuple10) 返回元组中的最小值  
#将列表转成元组  
list = [1, 2, 3]  
tuple11 = tuple(list)  
print(tuple11) #(1, 2, 3)  
# 元组的遍历  
for i in (1, 2, 3, 4):  
 print(i)

#### 字典

*"""  
概述:  
使用键-值对（key-value）存储，具有极快的查找速度  
注意：字典是无序的  
特性：  
1.字典中的key必须唯一  
2.key必须是不可变对象  
3.字符串、整数等都是不可变的，key作为key  
4.list可变，不能作为key  
和list比较  
1.查找和插入的速度极快，不会随着key—value的增加而变慢  
2.需要占用大量的内存，内存浪费慢  
"""*"""  
思考：保存多个学生的姓名和成绩  
使用字典，学生姓名为key，成绩作为值  
"""  
dict1 = {"tom":60, "lilei":70}  
  
#元素的访问  
#获取：字典名[key]  
print(dict1["lilei"])  
# print(dict1.get["tom"])  
# print(dict1.get["liuyaj"])  
#添加 修改就是删除  
dict1["123"] = 90  
dict1["lilei"] = 80  
print(dict1["lilei"]) #80  
print(dict1["123"]) #90  
print(dict1) #{'tom': 60, 'lilei': 80, '123': 90}  
#删除  
dict1.pop("tom")  
print(dict1) #{'lilei': 80, '123': 90}  
#遍历  
for key in dict1:  
 print(key, dict1[key]) #lilei 80 123 90  
  
for k, v in dict1.items():  
 print(k, v) #lilei 80 123 90  
  
for i, v2 in enumerate(dict1):  
 print(i, v2) # 0 lilei 1 123

#### 例题1：时间下一秒

timestr = input("请输入时间")  
#12:59:59  
timelist = timestr.split(":")  
  
h = int(timelist[0])  
m = int(timelist[1])  
s = int(timelist[2])  
s += 1  
if s == 60:  
 m += 1  
 s = 0  
 if m == 60:  
 h += 1  
 m = 0  
 if h == 24:  
 h = 0  
print("%.2d:%.2d:%.2d" %(h, m, s))

### 1.6函数基础

#### set

*"""  
set:类似dict，是一组key的集合，不存储value  
本质：无序和无重复元素的集合  
重复的元素会自动被过滤  
# 创建  
创建set需要一个list或者tuple或者dict作为输入集合  
经常用于去重  
"""*s1 = set([1, 2, 3, 5, 4, 2])  
print(s1) #{1, 2, 3, 4, 5}  
s2 = set((1, 2, 3, 5, 4, 2))  
print(s2) #{1, 2, 3, 4, 5}  
s3 = set({1: "good",2: "nice"})  
print(s3) # {1, 2}  
#添加  
s4 = set([1, 2, 3, 5, 4, 2])  
s4.add(6)  
print(s4) #{1, 2, 3, 4, 5, 6}  
# s4.add([7, 8, 9]) #set的元素不能是列表，因为列表是可变的  
# print(s4) # TypeError: unhashable type: 'list'  
s4.add((7, 8, 9)) #{1, 2, 3, 4, 5, 6, (7, 8, 9)}  
print(s4)  
# s4.add({7:"a"}) # #set的元素不能是字典，因为字典是可变的  
#插入整个list、tuple、字符串， 打碎插入  
s5 = set([1, 2, 3, 5, 4, 2])  
s5.update([6, 7, 8])  
print(s5) #{1, 2, 3, 4, 5, 6, 7, 8}  
s5.update((5, 10))  
print(s5) #{1, 2, 3, 4, 5, 6, 7, 8, 10}  
#删除  
s6 = set([1, 2, 3, 5, 4, 2])  
s6.remove(3)  
print(s6) #{1, 2, 4, 5}  
#遍历  
s7 = set([1, 2, 3, 5, 4, 2])  
for i in s7:  
 print(i)  
# set没有索引  
for index, data in enumerate(s7):  
 print(index, data)  
  
s8 = set([1, 2, 3])  
s9 = set([2, 3, 4])  
#交集  
a1 = s8 & s9  
print(a1) #{2, 3}  
print(type(a1))  
#并集  
a2 = s8 | s9  
print(a2) #{1, 2, 3, 4}

#### 类型转换

# list --> set  
l1 = [1, 2, 3, 4, 5]  
s1 = set(l1)  
#tuple --> set  
t2 = (1, 2, 3, 4, 5)  
s2 = set(t2)  
#dict --> set 很少转换  
# set --> list  
s3 = {1, 2, 3, 4}  
print(type(s3))  
l3 = list(s3)  
print(l3)  
#set --> tuple  
s4 = {1, 2, 3, 4}  
t4 = tuple(s4)  
print(t4)

#### list去重复

l = [1, 2, 3, 4, 5, 4, 3]  
"""  
s = set(l)  
l1 = list(s)  
print(l1)  
"""  
l = list(set(l))  
print(l)

#### 迭代器

*"""  
可迭代对象：可以直接作用于for循环的对象，统称为可迭代对象（Iterable）。可以用isinstance（）判断  
是否是Iterable对象  
可以直接作用于for的数据类型一般分两种  
1、数集数据类型：如list、tuple、dict、set、string  
2、是generator，包括生成器和带yield的generator function  
迭代器  
"""*from collections import Iterable  
print(isinstance([], Iterable)) #True  
print(isinstance((x for x in range(10)), Iterable))  
"""  
迭代器:不但可以作用与for循环，还可以被next()函数不断调用并返回下一个值，直到最后  
跑出一个StopIteration错误，表示无法继续返回下一个值  
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对象）  
可以使用isinstance()函数判断一个对象是否是Iterator对象  
"""  
from collections import Iterator  
print(isinstance([], Iterator))  
print(isinstance((x for x in range(10)), Iterable)) #迭代器  
  
l = (x for x in range(4))  
print(next(l))  
print(next(l))  
print(next(l))  
print(next(l))  
# print(next(l)) # StopIteration  
  
#转成Iterator对象  
a = iter([1, 2, 3, 4])  
print(next(a))  
print(next(a))  
  
print(isinstance(iter(()), Iterator)) # True(元组转Iterator)

#### 函数概述

*"""  
认识函数：在一个项目中，某些功能会反复的使用。那么会将功能封装成函数，当我们要使用功能  
的时候直接调用函数即可  
本质：函数就是对功能的封装  
优点：  
1、简化代码结构，增加了代码的复用度（重复使用的程度）  
2、如果想修改某些功能或者调试某个BUG，只需修改对应的函数即可  
"""*"""  
定义函数：  
格式： def 函数名(参数1，参数2，……参数n):  
 语句  
 return 表达式  
def: 函数代码块以def关键字开始  
函数名：遵循标识符规则  
参数列表：任何传入函数的参数和变量必须放在圆括号之间，用逗号分隔。函数从函数的调用者那里  
获取的信息  
冒号：函数内容（封装功能）以冒号开始，并且缩进  
语句：函数封装的功能  
return：一般用于结束函数，并返回信息给函数的调用者  
表达式：要返回给函数的调用者的信息  
注意：最后的return表达式，可以不写，相当于return None  
"""  
  
"""  
函数的调用  
格式：函数名（参数列表）  
函数名：是要使用的功能的函数的函数名  
参数列表：函数调用者给函数传递的信息  
函数调用的本质：实参给形参赋值的过程  
  
"""  
def myprint():  
 print("good man ")  
 print("nice man ")  
  
myprint()

**最简单的函数（无参无返回值）**

def myprint():  
 print("good man ")  
 print("nice man ")  
  
myprint()

#### 函数的参数

#需求：编写一个函数，给函数一个字符串和年龄，在函数内部打印出来  
#形参（形式参数）：定义函数时，小括号中的变量，本质是变量  
#参数必须按顺序传递，个数对应  
def myprint(str, age):  
 print(str, age)  
#实参（实际参数）：调用函数时给函数传递的数据，本质是值  
myprint("good man", 16)

#### 函数的返回值

#编写函数，实现功能：给函数两个数，返回者两个数的和  
def add(num1, num2):  
 add = num1 + num2  
 return add  
#执行完return语句，函数就结束了，后面的代码不会执行  
 print("\*\*\*")  
print(add(1, 2))

#### 传递参数

*"""  
值传递：传递的是不可变类型（数字，字符串，元组）  
"""*def func1(num):  
 num = 10  
temp = 20  
func1(temp) #num = temp  
print(temp) #20  
  
"""  
引用传递：传递的是可变类型（list、dict、set）  
"""  
def func2(list):  
 list[0] = 10  
li = [1, 2, 3, 4]  
func2(li)  
print(li)

#### 关键字参数

*"""  
概念：允许函数调用时的参数的顺序与定义时不一致  
"""*def myprint(str, age):  
 print(str, age)  
#使用关键字参数  
myprint(age = 16, str = "good man")

#### 默认参数

*"""  
概述：调用函数时，如果没有传递参数，则使用默认参数  
注意：如果使用默认参数，最好将默认参数放到最后  
"""*def myprint(str ="good man", age = 18):  
 print(str, age)  
  
myprint("good man ")

#### 不定长参数

*"""  
概述：能处理比定义时更多的参数  
# 加了星号（\*）的变量存放所有未命名的变量参数，如果在函数调用时没有给定参数，它就是一个空元组  
"""*def function(name, \*arr):  
 print(name)  
 for x in arr:  
 print(x)  
function("good","nice","123")  
  
def mysum(\*i):  
 sum = 0  
 for k in i:  
 sum += k  
 return sum  
print(mysum(1,2,3,4))  
  
# \*\*代表键值对的参数字典，和\*所代表的意义一样  
def func2(\*\*kwargs):  
 print(kwargs)  
 print(type(kwargs))  
# 必须传递的是关键字参数  
func2(a = 10, b = 2, c = 3)  
  
  
#可以接受任意参数  
def func3(\*agr, \*\*kwargs):  
 pass #代表一个空语句

#### 匿名参数

*"""  
概念：不使用def这样的语句定义函数，使用lambda来创建匿名函数  
特点：  
1、lambda只是一个表达式，函数体比def简单  
2、lambda的主体是一个表达式，而不是代码块，仅仅只能在lambda表达式中封装简单的逻辑  
3、lambda函数有自己的命名空间，且不能访问自有参数列表之外的或者全局命名空间的参数  
4、虽然lambda是一个表达式，只能写一行，与C、C++内联函数不同。  
格式：lambda 参数1， 参数2，……, 参数n:expression  
  
"""*sum = lambda num1, num2:num1 + num2  
print(sum(1, 2))

例1 九九乘法表：

j = 1  
for i in range(1,10):  
 for j in range (1,i+1):  
 sum = i \* j  
 print("%d \* %d = %d" %(i, j, sum),end=' ')  
 print("")

print("%d \* %d = %d" %(i, j, sum),end=' ') 同一行输出

### 1.7、装饰器&偏函数与作用域与异常处理与文件读写

**Python 标识符**

在 Python 里，标识符由字母、数字、下划线组成。

以下划线开头的标识符是有特殊意义的。以单下划线开头 **\_foo** 的代表不能直接访问的类属性，需通过类提供的接口进行访问，不能用 **from xxx import \*** 而导入。

以双下划线开头的 **\_\_foo** 代表类的私有成员，以双下划线开头和结尾的 **\_\_foo\_\_** 代表 Python 里特殊方法专用的标识，如 **\_\_init\_\_()** 代表类的构造函数。

Python有五个标准的数据类型：

* Numbers（数字）String（字符串）List（列表）Tuple（元组）Set（集合）Dictionary（字典）
* **不可变数据（3 个）：**Number（数字）、String（字符串）、Tuple（元组）；
* **可变数据（3 个）：**List（列表）、Dictionary（字典）、Set（集合）。

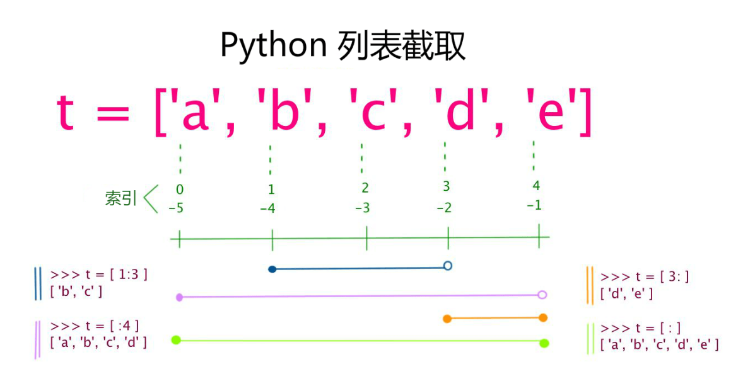
字符串或串(String)是由数字、字母、下划线组成的一串字符。str="hello,world"

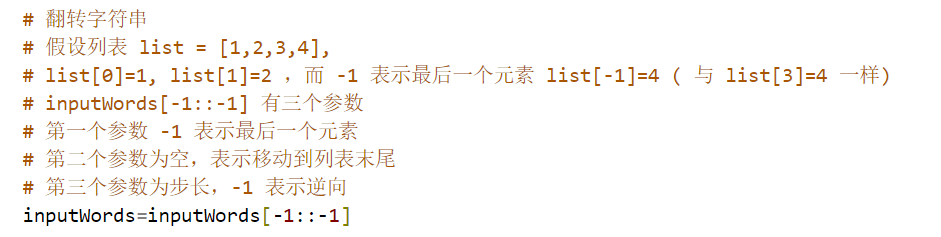
列表用 **[ ]** 标识，是 python 最通用的复合数据类型。list=['hello',2,3,4,'world']

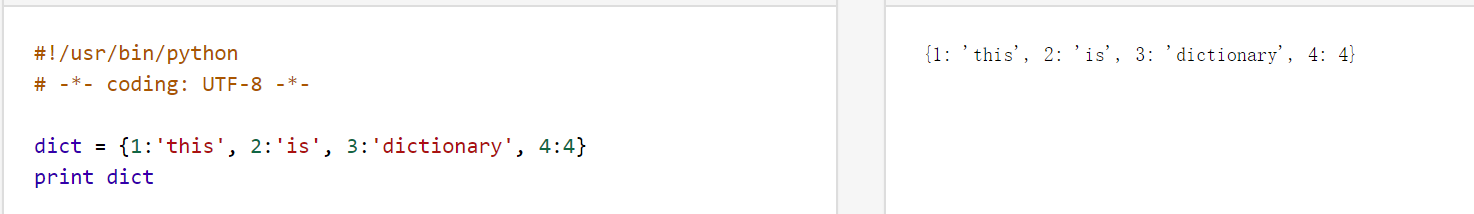
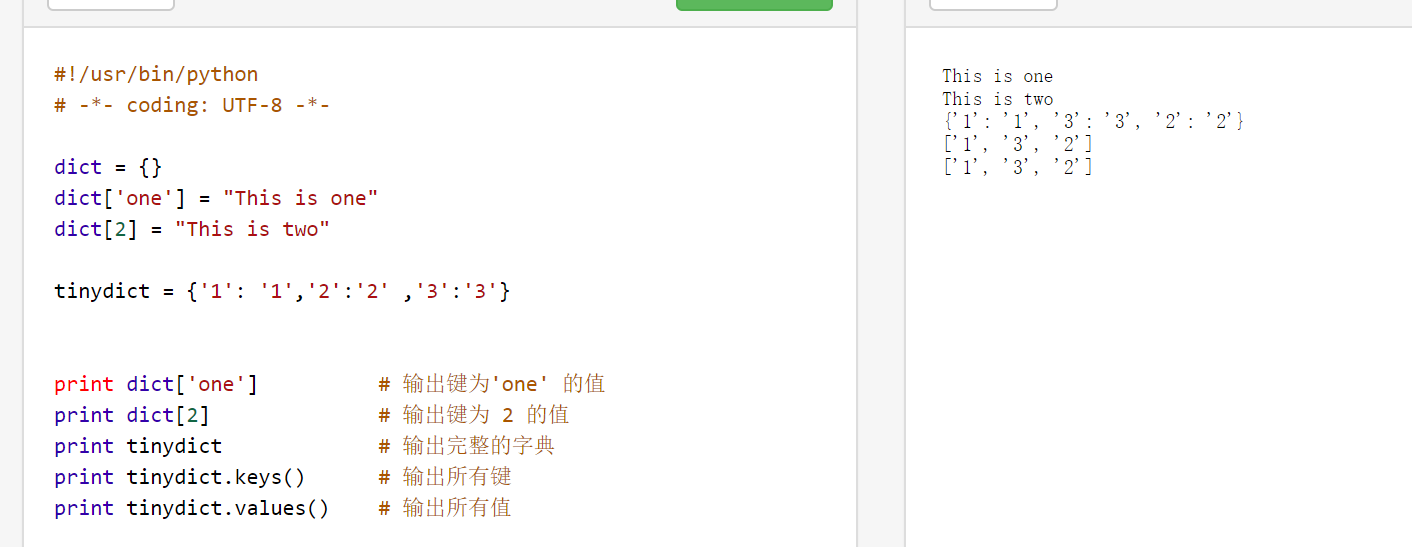
元组用 **()** 标识。内部元素用逗号隔开。但是元组不能二次赋值，相当于只读列表。tuple=('hello',2,3,4,'world')

字典用"{ }"标识。字典由索引(key)和它对应的值value组成。

取值顺序：

* 从左到右索引默认0开始的，最大范围是字符串长度少1
* 从右到左索引默认-1开始的，最大范围是字符串开头
* 下标可以为空表示取到头或尾





字典输出顺序？？

## 数据类型

数据类型 分为数字型和非数字型。

数字型包括整型，长整型，浮点型，复数型；

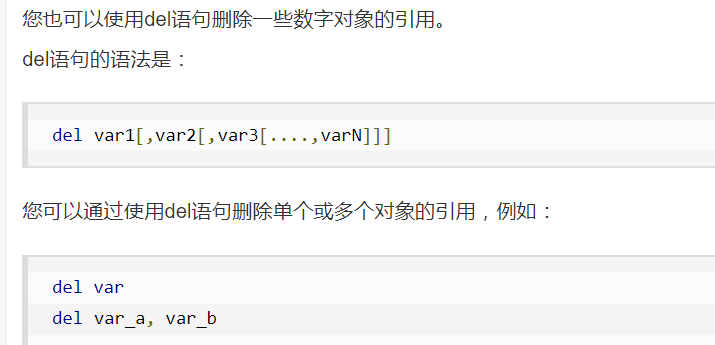
非数字型包括字符串，列表，元组和字典 ；

非数字型的共同点：都可以使用切片、链接（+）、重复（\*）、取值（a[]）等相关运算;

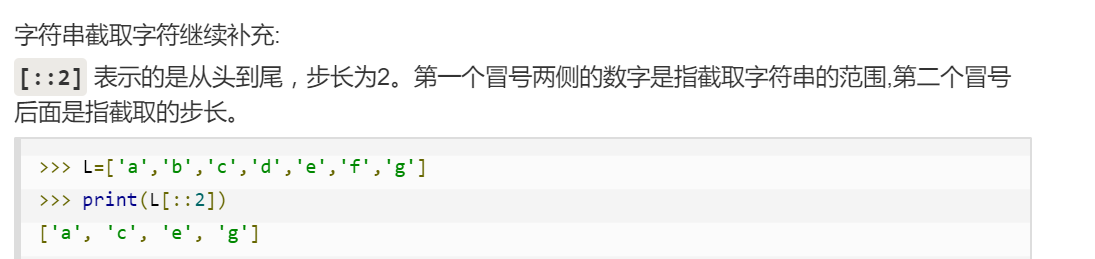
非数字型的不同点：

列表 可以直接赋值，元组不可以赋值，字典按照 **dict[k]=v** 的方式赋值。





### 字符串



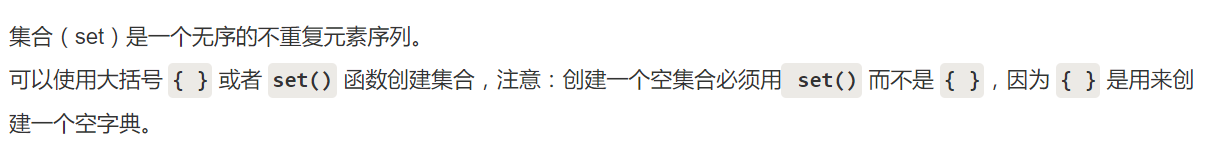
### 列表



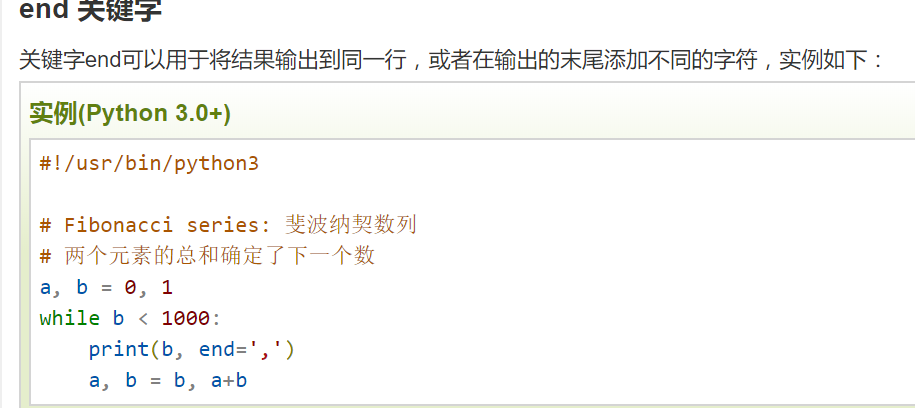
### 元组



### 集合



a, b = b, a+b



## 条件控制

a = int(input(‘请输入一个数字’))

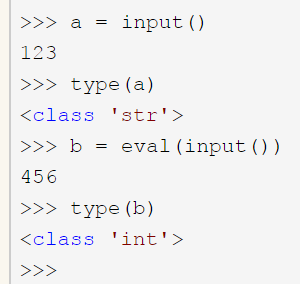
import random

~~x = random.choice(0, 100)~~  % x为0-99取一个数

x = random.choice(range(100))

a = int(input(‘请输入一个数字’))

a = eval(input(‘请输入一个数字’)) %判读输入的数字



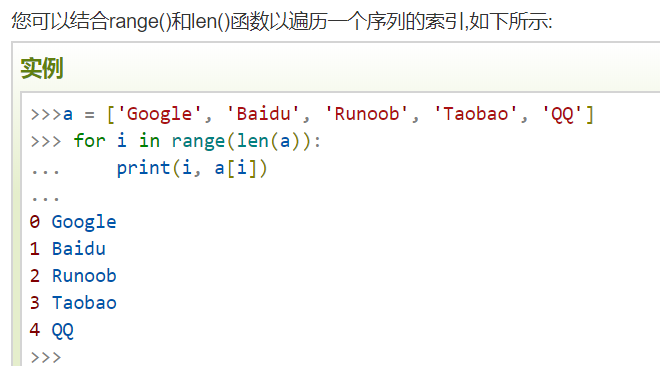
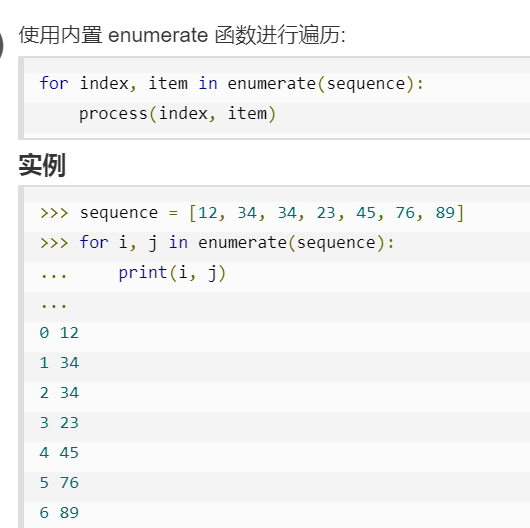


## 循环语句

循环次数用 counter = 1 flag = 1

counter += 1

输出 print(‘1 到 %d 之和为: %d’ % （n, sum）)



## 迭代器与生成器

迭代是Python最强大的功能之一，是访问集合元素的一种方式。

迭代器是一个可以记住遍历的位置的对象。

迭代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束。迭代器只能往前不会后退。

迭代器有两个基本的方法：**iter()** 和 **next()**。

字符串，列表或元组对象都可用于创建迭代器：