

《交通大数据技术》



2024/6/14

交通大数据技术

马晓磊
交通科学与工程学院
2024年

简介与工具准备



本节大纲

- **python历史**
- **Anaconda环境集成平台**

python语言的诞生

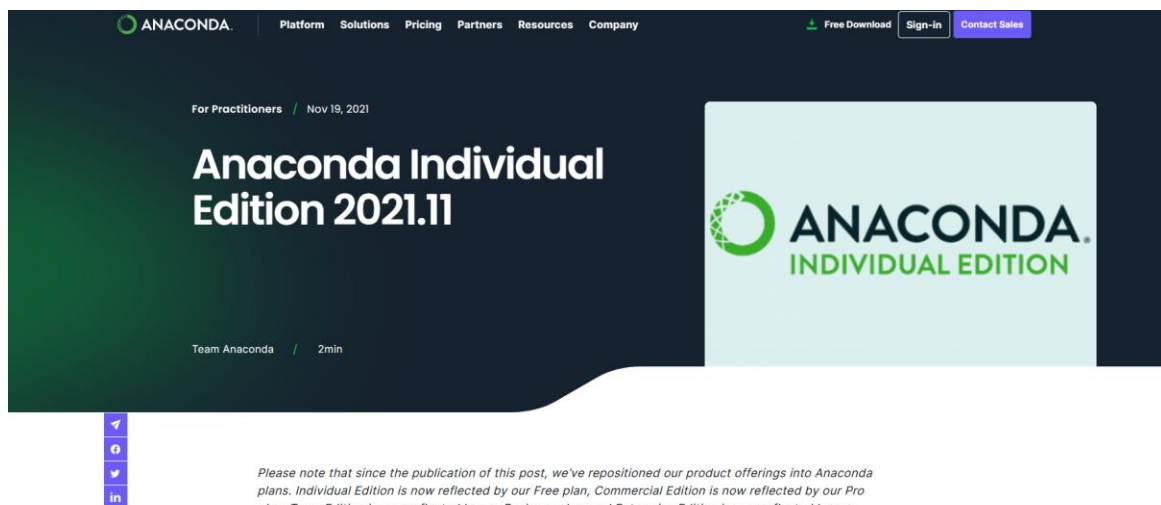
- Python语言的拥有者：Python Software Foundation(PSF)，非盈利组织
- Python语言的创立者：Guido van Rossum
 - 2002年——Python 2.x; 2008年——Python 3.x

python语言特点

- Python具有清晰、简洁的语法结构，使用自然语言元素，如常用英文关键字和直观的标点符号，使得代码易于阅读和理解，降低了学习门槛。严格的缩进规则强化了代码的整洁性和可读性，鼓励编写规范化的代码。
- 支持面向对象编程，允许定义类和对象，实现继承、封装和多态等特性，便于构建复杂的程序结构。

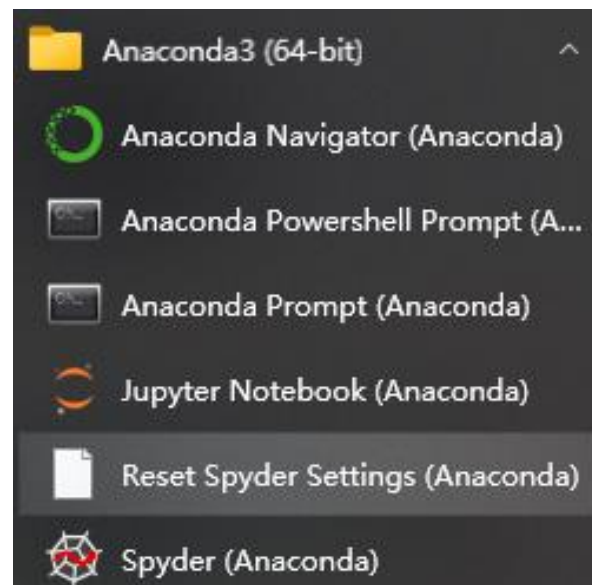
Anaconda环境集成平台

Anaconda 是一款巨大的 Python 环境集成平台，是必须下载的工具。其内含 Python 解释器、Jupyter Notebook 代码编辑器以及三方库。



Please note that since the publication of this post, we've repositioned our product offerings into Anaconda plans. Individual Edition is now reflected by our Free plan, Commercial Edition is now reflected by our Pro plan, Team Edition is now reflected by our Business plan, and Enterprise Edition is now reflected by our Enterprise plan. Click [here](#) to learn more about our plans for individuals, and click [here](#) to learn more about our plans for organizations.

Update: [Anaconda 2022.05 is now available](#)



python基础语法



本节大纲

- **python基本语法与编程规范**
- **python赋值**
- **python函数**

python基本语法与编程规范

缩进

- 类定义、函数定义、选择结构、循环结构、异常处理结构、with块，行尾的冒号表示缩进的开始
- python程序是依靠代码块的缩进来体现代码之间的逻辑关系的，缩进结束就表示一个代码块结束
- 同一个级别的代码块的缩进量必须相同
- 一般而言，以4个空格为基本缩进单位

```
with open (fn) as fp:
    for line in csv.reader(fp):
        if line:
            print(*line)
```

每个import语句只导入一个模块，最好按**标准库**、**扩展库**、**自定义库**的顺序依次导入

```
import csv
import random
import datetime
import pandas as pd
import matplotlib.pyplot as plt
```


python基本语法与编程规范

print()函数

- 向标准输出（通常是终端或控制台）输出指定的信息。
- Python中最常用的输出手段之一。

```
print("Hello, World!") # 输出字符串  
print(123, 456, sep=", ") # 输出数字, 指定分隔符  
print("End of line", end=">>>") # 更改结束字符
```

input()函数

- 从标准输入（通常是键盘）接收用户的输入，并以字符串形式返回。
- Python中实现人机交互的关键工具。

```
name = input("Please enter your name: ")  
age = int(input("Enter your age: ")) # 用户输入后需手动转换类型  
print(f"Hello, {name}! You are {age} years old.")
```

变量名=表达式，将右侧表达式的值绑定到左侧变量名的过程。

- 简单赋值示例

```
x = 5  
y = "Hello"  
z = True
```

- 列表、字典、元组等复合数据类型赋值

```
list_example = [1, 2, 3]  
dict_example = {'key': 'value'}  
tuple_example = (4, 5, 6)
```

函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。

- 函数能提高应用的模块性，和代码的重复利用率
 - Python提供了许多内建函数，比如print()
 - 可以自己创建函数，这被叫做用户自定义函数

如何自己创建函数？

- 函数代码块以 def 关键词开头，后接函数标识符名称和圆括号()
- 传入参数和自变量必须放在圆括号中间。圆括号之间可以用于定义参数
- 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明
- 函数内容以冒号起始，并且缩进
- return [表达式] 结束函数，选择性地返回一个值给调用方。不带表达式的return相当于返回 None。

语法

```
def functionname( parameters ):  
    "函数_文档字符串"  
    function_suite  
    return [expression]
```

实例

```
def printme( str ):  
    "打印传入的字符串到标准显示设备上"  
    print str  
    return
```

python变量类型



本节大纲

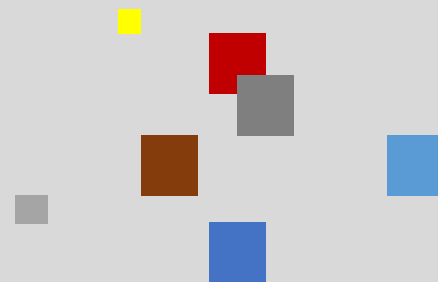
- **变量**
- **python关键字**
- **python字符串**
- **元组 Tuple, 列表 List, 字典 Dictionary**

在Python中，取名字必须遵循如下的命名规则。

- ◆ 变量名的第1个字符只能是英文字母或下划线。
- ◆ 变量名中的其余字符可以是英文字母或数字。
- ◆ 不能将Python自身留用的关键字（或保留字）作为变量名。

例2-1 下面给出的标识符，哪些可以作为变量名，哪些不可以？说明理由。

- (a) message_1 (b) 3cx
(c) _DCY (d) student_name
(e) 1_message (f) dmax C
(g) BBC\$TY



答：(a) (c) (d) 可以作为变量名；

(b) 不行，因为数字不能作为变量名的开始字符；

(e) 不行，因为数字不能作为变量名的开始字符；

(f) 不行，因为变量名中不能出现空格（dmax和C之间有一个空格）；

(g) 不行，因为变量名只能包含字母、数字、下划线，这里出现符号“\$”。



【说明】

Python语言是区分英文字母大小写的。也就是说，在Python里，即使英文字母全都相同，仅大小写不同，也代表不同的变量。因此，birthday、Birthday、BIRthday是3个不同的变量，而不是同一个变量。

python关键字

Python部分关键字表

and	break	class	def	del	continue	for
except	else	from	return	is	while	raise
import	if	as	elif	or	yield	finally
in	pass	with	True	False	None	

我们可以在“交互执行”模式下，通过以下方法来获得Python中的所有关键字名称：

```
>>>import keyword
>>>keyword.kwlist (或>>>print(keyword.kwlist))
```

这时，窗口里会输出Python中所有关键字的名字：

```
['false','None','True','and','as','assert','break','class','continue','def','del','elif','else','except',
'finally','for','from','global','if','import','in','is','lambda','nonlocal','not','or','pass','raise','re
turn','try','while','with','yield']
```


python关键字

在Python程序里，如果一不小心将关键字当成了变量名使用，会出现什么情况呢？

例2-2 如果不小心把关键字当成了变量名使用，Python会在窗口里给出出错信息。例如，在Sublime Text窗口中编写一个名为test3.py的程序，这里错误地把字符串“hello python world”赋给关键字for，如图2-3所示。

存储后在Python命令窗口运行该程序，Python在命令窗口输出图2-4所示的出错信息，即：

- 所列信息的第1行告诉我们，文件test3.py的第1行（line 1）有一个错误；
- 所列信息的第2行原封不动地列出了出错行的内容；
- 所列信息的第3行显示一个插入符（^），指示出出错的位置；
- 所列信息的最后一行显示该错误属于“语法错误”（SyntaxError: invalid syntax）。

python关键字



- 我们可以通过Python提供的这些信息检查程序，找出出错原因，对错误进行修正。
- Python以不同颜色区分输入的内容，最大限度地向用户提醒可能出现的问题，帮助程序员编写出正确的程序。

1. 字符串的定义

字符串中的元素仅限于一个一个的字符：英文字母、数字、空格，以及键盘上Python允许使用的其他字符。在Python里，把字符串定义为用单引号（'）或双引号（"）括起来的一系列字符。例如'this is a book.'、"this is a book."、'that is a string.'和"that is a string."都是正确的字符串。

设有变量str，将字符串'a0a1...ai-2ai-1'或字符串"a0a1...ai-2ai-1"赋给它，即：

```
str = 'a0a1...ai-2ai-1'或 str = "a0a1...ai-2ai-1"
```

那么，a0、a1、...、ai-2、ai-1就是组成该字符串的一个个字符，整个字符串就是变量str的值。若将单引号或双引号内的字符个数记为n，那么称n为字符串的“长度”。当一个字符串的长度n=0时，称其是一个“空串”，如str1=""和str2=""都是空字符串。

2. 字符串的“索引”

字符串中每个字符的序号称为它在字符串里的“索引”，不同的索引对应字符串中的不同字符。要注意，Python规定，索引从0开始，而不是从1开始。因此，字符串中第1个字符的索引值为0，第2个字符的索引值是1，依次类推。

3. 主串与子串

字符串中任意多个连续字符组成的子序列，称作该串的“子串”，字符串本身称为“主串”。子串的第1个字符在主串中的位置，称作该子串“在主串中的位置”。

4. 引号的使用

在字符串中使用引号时，要注意两点：

- 一是引号必须成对出现，因此要关注程序中与左引号配对的右引号在什么位置；
- 二是单引号只能与单引号配对，双引号只能与双引号配对，不能因为它们都可以定义字符串，就乱配对。在程序中，丢失“对”或乱配“对”，都会产生语法性的错误。

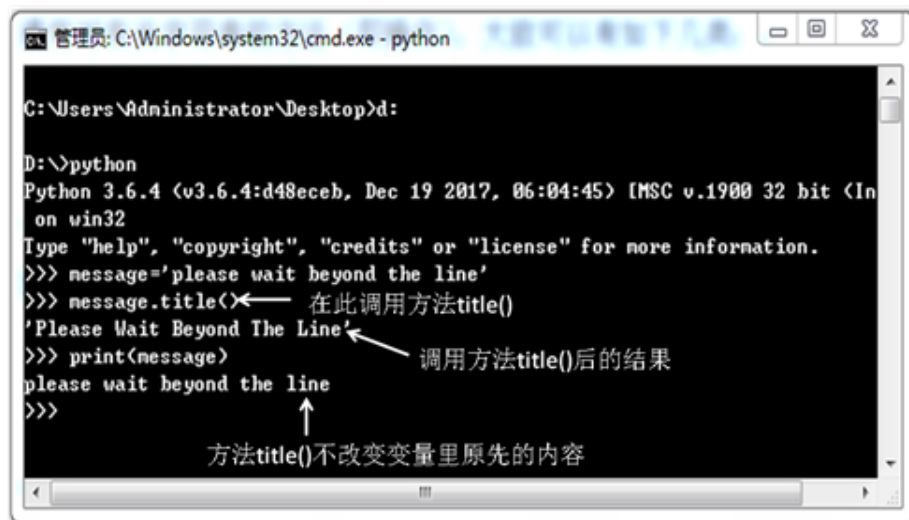
关于字符串的方法

1. title()

功能：将字符串变量里出现的每个单词的首字母改为大写。

用法：<变量名>.title()

例 输入字符串“please wait beyond the line”，将其赋给变量message，调用方法title()



```
管理员: C:\Windows\system32\cmd.exe - python
C:\Users\Administrator\Desktop>d:
D:\>python
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel) on win32]
Type "help", "copyright", "credits" or "license()" for more information.
>>> message='please wait beyond the line'
>>> message.title() ← 在此调用方法title()
'Please Wait Beyond The Line' ← 调用方法title()后的结果
>>> print(message)
please wait beyond the line
>>>
```

↑
方法title()不改变变量里原先的内容

- 该方法把字符串中的每一个单词的首字母改为大写。
- 该方法不改变字符串变量的原先内容。也就是说，原字符串单词的首字母仍保持调用title()前的状态不变。

关于字符串的方法

(1) 调用时，变量名和方法名之间要用点号 (.) 隔开，实现变量对方法的调用。

(2) 方法名后跟随的括号里面是用户调用该方法时，提供给方法的调用信息，即使不需要提供任何信息，括号也不能缺少。例如，字符串变量调用方法title()时，虽然不需要提供什么额外调用的信息，但后面跟随的括号绝不可不写。



【注意】

在print()的括号里可以是变量对方法的调用。例如：

```
message='please wait beyond the line.'
```

```
print(message.title())
```

这时，print()输出的是：

```
Please Wait Beyond The Line
```

关于字符串的方法

2. upper()

功能：将字符串变量里的所有字母都改为大写。

用法：<变量名>.upper()

3. lower()

功能：将字符串变量里的所有字母都改为小写。

用法：<变量名>.lower()

4. find()、rfind()

(1) find()

功能：find()用来查找子串在主串中的位置，返回子串首次出现在主串中时的第1个字符的索引编号；主串中不存在该子串时，查找失败，返回-1。

关于字符串的方法

用法: `<变量名>.find(sub, start, end)`

其中:



sub表示想要查找的子串, 该参数不可省略, 如果没有找到, 返回-1;



start表示开始查找的索引位置, 如省略该参数, 表示从主串的开头开始查找;



end表示结束查找的索引位置, 如果省略该参数, 表示从start开始, 一直查找到主串末尾。

(2) `rfind()`

功能: `rfind()`用来查找子串在主串中的位置, 返回子串最后一次出现在主串中时的第1个字符的索引编号; 主串中不存在该子串时, 查找失败, 返回-1。

字符串的“切片”

所谓字符串的“切片”，其实就是通过截取字符串的一个片段，形成一个子字符串。例如，有字符串 `I'm a student.`，它的任何一个片段，如 `'m a s`、`a stude`、`ent.`等，甚至它本身，都是它的一个“切片”。人们通过使用Python提供的方括号“`[]`”运算符，即可提取一个字符串中单个字符或局部范围的切片。

用法：<字符串>[start:end:step]

其中：



start表示切片起始位置的索引；



end表示切片终止位置的索引；



step表示切片索引的增、减值（步长）。

字符串的“切片”



`str[m]`表示按指定索引值`m`，获取字符串`str`中的某个切片；



`str[m:n]`表示从索引值`m`到`n-1`，获取字符串`str`中的某个切片；



`str[m:]`表示从索引值`m`开始，直到字符串`str`结束，获取切片；



`str[:n]`表示从索引值0开始，直到索引值`n-1`结束，获取字符串`str`中的切片；



`str[:]`表示复制整个字符串`str`作为切片；



`str[::-1]`表示将反转后的整个字符串`str`作为切片。

字符串的“切片”

这里解释以下两点。

第一

计算切片在字符串的位置时，索引从start（即m）开始，切片包含了start指示的字符；切片到end（即n）结束，但切片不包含end指示的字符。所以，切片真正“切”下来的字符，是从索引值start到end-1的部分。

Python为字符串提供了负索引，负索引从字符串的末尾算起，到字符串的头部结束。例如，字符串'I\m a student.'的索引和负索引如表2-3所示。

第二

元组 Tuple, 列表 List, 字典 Dictionary

元组

1. len()

- ◆ 功能：求元组中元素的个数，即元组的长度。
- ◆ 用法：len(<元组变量名>)

例如，定义一个元组yz=('Zong',78, 'male')，输入语句len(yz)，按Enter键换行后，窗口的输出是3，即该元组里有3个元素。这实际上也就是求元组长度的方法。

2. count()

- ◆ 功能：求元组中某元素出现的次数。
- ◆ 用法：<元组变量名>.count(<元组中的元素名>)

元组 Tuple, 列表 List, 字典 Dictionary

元组

编写程序如下:

```
yz1=('Zong da hua' ,78 , 'male',)
yz2=('Li wei',59,'male')
yz3=('chen chun ting',76,'female')
yz4=('ni nai hui',45,'female')
yz5=('jing tao hai',49,'male')
zyz=yz1+yz2+yz3+yz4+yz5          #拼接成一个大元组zyz
print('male=',zyz.count('male'),'female=',zyz.count('female'),end='\n')
print('End')
```

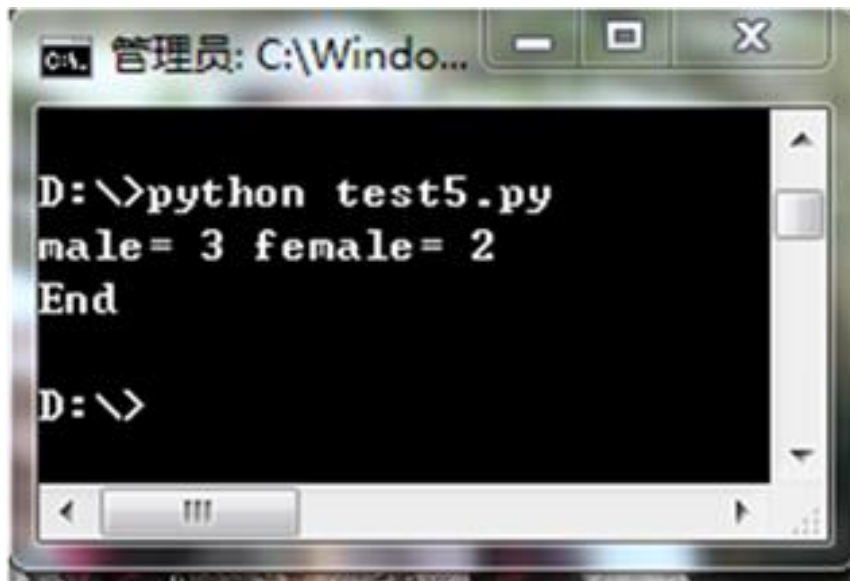
程序中, 利用+运算符, 先把5个人的信息拼装到一个大的元组zyz里, 然后语句:

```
print('male=',zyz.count('male'),'female=',zyz.count('female'),end='\n')
```

元组 Tuple, 列表 List, 字典 Dictionary

元组

两次由元组zyz调用方法count(), 统计出male (男) 和female (女) 的人数。图4-14所示是运行的结果。



```
D:\>python test5.py
male= 3 female= 2
End
D:\>
```

3. index()

- ◆ 功能: 获得某元素在元组中第1次出现的位置。
- ◆ 用法: <元组变量名>.index(<元素名>, i, j)

其中, <元组变量名>即是调用index()方法的元组名; <元素名>即元组中所要查找的元素, 该参数是必不可少的; i是查找的开始位置的索引, j是查找的结束位置的索引, i和j两个参数都可以省略, 省略时表明是从头查到尾。

元组 Tuple, 列表 List, 字典 Dictionary

元组

编写程序如下：

```
yz1=('Zong da hua',78,'male',)  
yz2=('Li wei',59,'male')  
yz3=('chen chun ting',76,'female')  
yz4=('ni nai hui',45,'female')  
yz5=('jing tao hai',49,'male')  
zyz=yz1+yz2+yz3+yz4+yz5  
print(zyz.index('male'))  
print(zyz.index('male',6))  
print('End')
```

程序中第1条输出语句： `print(zyz.index('male'))`

表示要查找元组zyz中元素male第一次出现位置的索引值，由于没有i和j，应该是从元组zyz的开头查起，因此结果是2。数一数元组zyz里第1次出现的元素“male”的索引值，确实是2。

元组 Tuple, 列表 List, 字典 Dictionary

元组

程序中的第2条输出语句：

```
print(zyz.index('male',6))
```

表示从索引值为6的地方开始查找元素 male 第一次出现位置的索引值。数一数元组 zyz 里从索引值6开始的第一次出现的元素“male”的索引值，确实是14。



A screenshot of a Windows command prompt window titled "管理员: C:\Windows\sy...". The window shows the execution of a Python script. The prompt is "D:\>python test5.py". The output of the script is displayed on the next three lines: "2", "14", and "End". The prompt "D:\>" is shown again on the next line.

元组 Tuple, 列表 List, 字典 Dictionary

列表

列表定义的一般形式是：

<变量名>=[<数据表>]

- ◆ <变量名>被定义为一个list（列表）型的变量，它由右边的中括号所确定。
- ◆ 中括号里的<数据表>列出了该列表可以取的所有元素值，每个数据之间用逗号分隔。

例4-6 定义一个名为staf的列表：

```
staf=[('Ni_lin','male'),40,['1978:born','2002:graduation','2002:enlist']]
```

元组 Tuple, 列表 List, 字典 Dictionary

列表

创建列表编写程序如下：

```
staf=[('Ni_lin','male'),40,['1978:born','2002:g  
raduation','2002:enlist']]  
print(type(staf))  
print(type(staf[0]))  
print(staf[0])  
print(staf[0][0])  
print(staf[0][1])  
print(type(staf[1]))  
print(staf[1])  
print(type(staf[2]))  
print(staf[2])  
print(staf[2][0])  
print(staf[2][1])  
print(staf[2][2])
```



```
管理员: C:\Windows\system32\cmd.exe  
D:\>python test4.py  
<class 'list'> ← Staf是列表型的  
<class 'tuple'>  
<'Ni_lin', 'male'> } Staf的第1个数据是元组型的  
Ni_lin  
male  
<class 'int'> } Staf的第2个数据是整数型的  
40  
<class 'list'>  
['1978:born', '2002:graduation', '2002:enlist'] } Staf的第3个数据是列表型的  
1978:born  
2002:graduation  
2002:enlist  
D:\>
```

元组 Tuple, 列表 List, 字典 Dictionary

列表

- 第1条语句“`print(type(staf))`”是输出所定义的`staf`的类型，由输出结果可以看出，它是一个`list`（列表）。
- 第2条语句“`print(type(staf[0]))`”是输出列表`staf`第1个元素的类型，输出结果表明它是`tuple`（元组）。
- 第3、第4、第5条输出语句，是把该元组的内容、元组每一个元素的内容输出。
- “`staf[0]`”代表了列表`staf`的第1个元素（也可以视它为第1个元素的“变量名”）。正因为如此，该元组的第1、2个元素应该表示为`staf[0][1]`及`staf[0][1]`。

其实，创建列表也可以通过函数`list()`，把元组、字符串、函数`range()`等括在括号里，将它们转换成为一个列表。例如：

```
>>> lb1=list((4,7,6,1,8,98,77))      # list将元组lb1转换成列表
>>> lb1
>>> [4,7,6,1,8,98,77]
>>> lb2=list(range(1,10))            # list将range(1,10)转换成列表
```

元组 Tuple, 列表 List, 字典 Dictionary

列表

与列表有关的方法：

1. append()

功能：将给出的元素作为参数，添加到调用该方法的列表的末尾。

用法：<列表名>.append(<元素>)

```
staf=[('Ni_lin','male'),40,['1978:born','2002:graduation','2002:enlist']]
staf[2].append('2012:teacher')
print('\n',staf[2])
print('\n',staf[2][3])
print('End')
```

元组 Tuple, 列表 List, 字典 Dictionary

列表

2. insert()

功能：将元素按照指定的索引位置，插入调用此方法的列表中。

用法：<列表名>.insert(i,<元素>)

```
staf=[('Ni_lin','male'),40,['1978:born','2002:graduation','2002:enlist']]
staf[2].insert(1,'1996:worker')
print('\n',staf[2])
print('\n',staf[2][1])
print('End')
```

元组 Tuple, 列表 List, 字典 Dictionary

列表

3. extend()

功能：将另一个列表中的所有元素添加到列表的末尾。

用法：<列表名>.extend(<新列表名>)

```
month=['January','February','March','April','May','June','July']
```

➤ 定义另一个列表others:

```
others=['August','September','October','November','December']
```

➤ 将列表others添加到列表month的末尾。要求输出调用方法extend()后，列表month内的所有元素（一行最多输出7个元素）。为此，编写小程序如下：

```
month=['January','February','March','April','May','June','July']      #原列表
others=['August','September','October','November','December']          #新列表
month.extend(others)                                                      #调用方法
```

元组 Tuple, 列表 List, 字典 Dictionary

字典

创建字典

在Python里，字典是通过大括号 “{}” 来定义的，其一般形式是：<变量名>={<数据表>}

其中，<变量名>被定义为一个dict（字典）型的变量，用大括号括起的<数据表>里，列出了该字典当前可能取的所有元素值，每个数据元素之间用逗号分隔。

作为字典这种数据类型，它具有如下的特点：



(1) 字典中的数据，都是一个一个“键-值”对，键与值之间用 “:” 冒号隔开，每个键都与一个值相关联，通过键来访问与之相关联的值；



(2) 字典中的“键-值”对与“键-值”对之间，也就是数据元素与数据元素之间，以逗号作为分隔符；



(3) 一个字典里包含的“键-值”对，其键必须是唯一的，即不能有相同名字的键，但不同键的值可以不同，也可以相同。

元组 Tuple, 列表 List, 字典 Dictionary

字典

- 日常生活中，最常用的“键-值”对，可能应该算是“姓名”和“电话号码”了，姓名是“键”，号码是“值”，通过姓名就能够找到他（或她）的电话号码，这就是字典最基本的用法：查找。
- 最常作为“键”的数据类型是字符串和整数，列表、字典等数据类型不能作为键，因为它们本身具有可变性；常作为值的数据类型是整数、字符串、元组、列表，甚至字典本身。

```
code={'Li pin':356712,'Chen yun':125458,'Wen ti':894563,  
'Lin ling':337682,'Ru nin':675446,'Tang xi':345890}  
print(type(code),'\n')  
print(code['Lin ling'],'\n')  
print(code['Tang xi'])
```


元组 Tuple, 列表 List, 字典 Dictionary

字典

与字典有关的方法：

1. iter()

- 功能：从字典中不断取出“键-值”对中的键，与for循环连用时，就能够遍历字典中所有元素。也就是说，该方法可以达到遍历字典元素的目的。
- 用法：iter(<字典名>)

例如，编写小程序如下：

```
code={ 'Li pin':356712,'Chen yun':125458,'Wen ti':894563,  
        'Lin ling':337682,'Ru nin':675446,'Tang xi':345890  }  
for name in iter(code):  
    print('%10s:%8d'%(name,code[name]))  
else:  
    print('End')
```

这里，code是字典，里面有6个“姓名”和“电话号码”的“键-值”对。小程序里，通过for循环，借助方法iter()遍历整个字典元素，即在循环：**for name in iter(code)**

元组 Tuple, 列表 List, 字典 Dictionary

字典

与字典有关的方法:

2. items()

- 功能: 方法items()能够不断从字典中返回一个“键-值”对, 借助for循环, 可以直接获得“键”和“值”(因此需要有两个变量来分别接收所得到的“键”和“值”), 从而更加便捷地遍历字典中的所有元素。
- 用法: <字典名>.items()

例如, 把上面的小程序改写为:

```
code={ 'Li pin':356712,'Chen yun':125458,'Wen ti':894563,  
        'Lin ling':337682,'Ru nin':675446,'Tang xi':345890 }
```

```
For name , num in code.items():
```

```
    print('key: ' +name)
```

```
    print('value: ' +str(num))
```

```
else:
```

```
    print('End')
```

程序中, 字典code直接调用方法items(), 每次从code里返回一个“键-值”对, 并将“键”赋给for循环中的变量name, 把“值”赋给变量num, 这就比使用方法iter()更加直接和方便。该小程序执行的结果与图4-34一样。

控制流



本节大纲

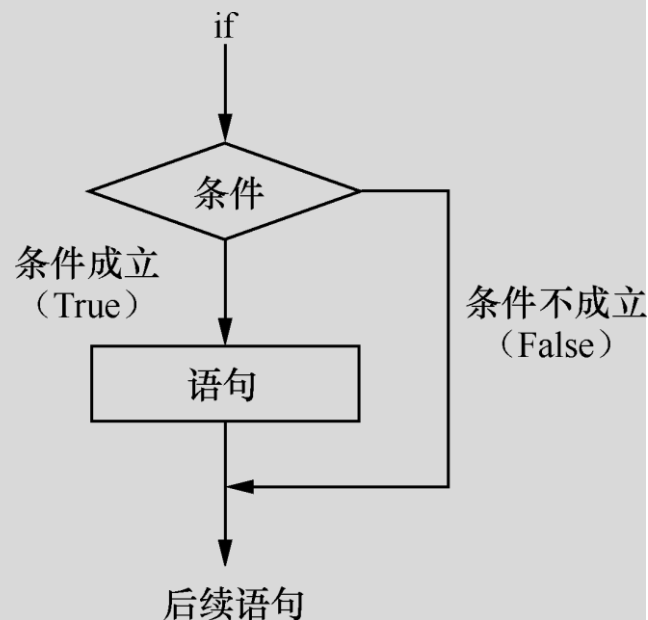
- **if-else**
- **for循环**
- **while循环**
- **循环中的break、continue**

if-else

if单分支选择语句的一般格式是：

```
if <条件> :  
    <语句>           #缩进语句块  
    <后续语句>
```

功能：程序在执行中遇到if时，若<条件>成立（取值True），则执行其后的缩进<语句>，执行完后去执行<后续语句>；否则，不执行缩进<语句>，直接执行<后续语句>。



if-else

注意以下几点。

(1) 在输入完if <条件>后面的冒号、按Enter键换行后，Sublime Text会让光标自动往右缩进，指明缩进语句块的输入起始位置，如图3-5所示。这是条件成立时要做的所有语句的输入处，因此可以是多条语句组成的语句块。



(2) 在Python程序里，无论在何处，冒号（:）都起到非常重要的作用。如果某行代码最后有冒号，那么下一行的程序代码必定缩进。Sublime Text会指明缩进的 actual 位置，为你完成缩进。

(3) 格式中的<后续语句>绝对不能够再缩进，应该将其恢复到原来的正常输入位置。

程序里，缩进的语句块限定了该语句块的作用范围，这个语句块的大小是由程序编制人员决定的，Sublime Text无法知道缩进何时结束。所以，程序设计者必须非常关注缩进的语句块的作用范围，不要出现提前结束缩进的情形，也不要扩大缩进的范围，那样程序执行时都不会得到预期的运行效果。缩进及取消缩进，都应该使用键盘上的Tab键，避免使用空格键。

例3-1 编写一个小程序，从键盘输入一个整数，然后输出其绝对值。

程序编写如下：

```
"""从键盘输入一个整数，并输出其绝对值"""
```

```
num=input('Input your integer:\n')
```

```
num=int(num)
```

```
if (num<0):
```

```
    num=-num #缩进语句块
```

```
print('The absolute value is:',num) # 结束缩进，恢复语句原来位置
```

分析如下。程序开始时是一条用3个双引号写的注释：

```
"""从键盘输入一个整数，并输出其绝对值"""
```


for-in循环是一种计次循环，通过计数来控制循环重复执行的次数。

for-in循环语句的一般格式是：

```
for <迭代变量> in <序列>:  
    <循环体>  
else:  
    <后续语句>
```

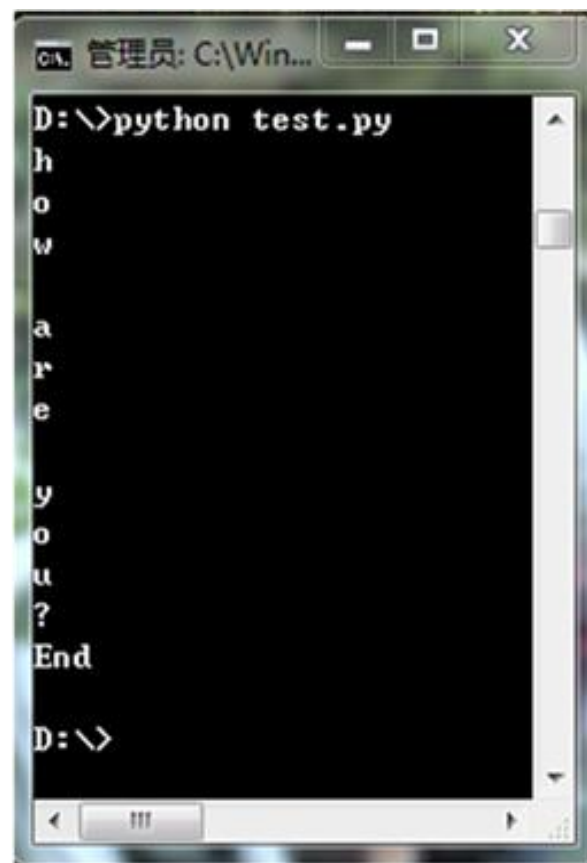
所谓“迭代”，即不停地进行有规律的替换。在程序设计里，同一个变量，用不同的值进行替代，就被称为一个“迭代变量”。其实，这里用“变量”也是可以的，只是添加了“迭代”二字，可以更形象地传达该变量的使用特征。

for循环

编写如下程序，利用for循环和range()函数，输出字符串“how are you?”中的所有字符：

```
s='how are you?'  
y=len(s)  
for x in range(y):  
    print(s[x])           #缩进  
else:  
    print('End')          #缩进
```

程序中，通过函数len()计算出字符串s的长度,并将其存放在变量y里。进入for循环，调用函数range(y)。该函数不断把0、1、2.....赋给迭代变量x，并去执行循环体“print(s[x])”。也就是不断将字符串的切片s[0]、s[1]、s[2].....等输出。

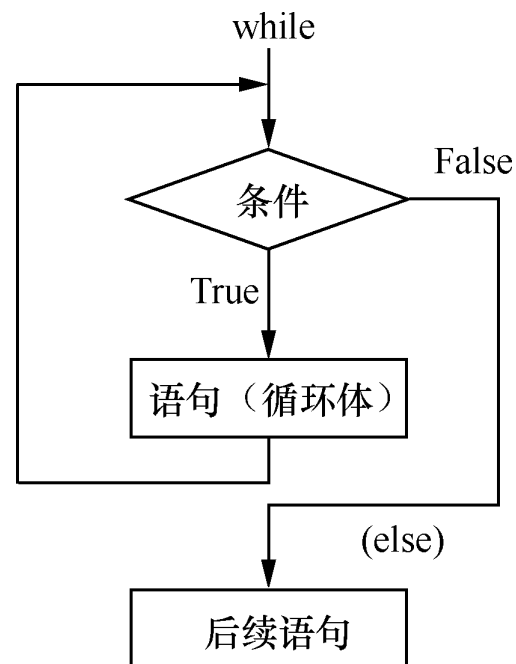


while循环

while循环语句的一般格式是：

```
while <条件> :  
    <语句(循环体)>  
else :  
    <后续语句>
```

功能：当执行遇到while时，判断<条件>是否取值True，即是否成立。如果成立，就执行缩进的循环体，转而又去判断<条件>的取值；如果<条件>取值False，即不成立，则去执行<后续语句>。

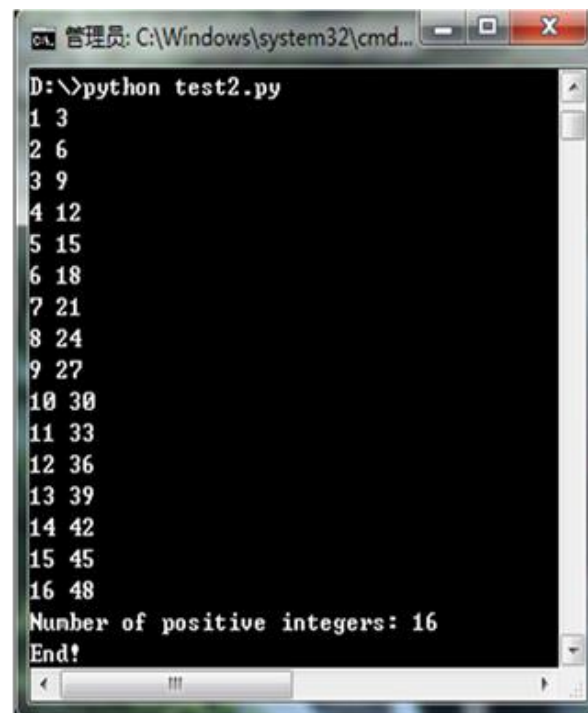


while循环

编写一个程序，输出1 ~ 50的所有能被3整除的正整数。

程序编写如下：

```
num=1      #控制循环执行的变量初值
i=0        #记录这种正整数的个数
while (num<=50):
    if (num%3==0):
        i+=1
        print(i,num)
        num+=1
    else:
        num+=1
else:
    print('Number of positive integers is in total=',i)
    print('End!')
```



```
D:\>python test2.py
1 3
2 6
3 9
4 12
5 15
6 18
7 21
8 24
9 27
10 30
11 33
12 36
13 39
14 42
15 45
16 48
Number of positive integers: 16
End!
```

图3-18所示是该程序运行的结果。



循环中的break、continue

1. break语句

功能：在循环体中，如果遇到break，就立即中断整个循环，离开循环体，去执行该循环的<后续语句>。也就是说，一旦break语句被执行，将会使整个循环提前结束。

2. continue语句

功能：continue一定是出现在循环体内的。遇到它时，就跳过循环体中它后面原本应该执行的其他语句（如果还有的话），提前结束本次循环，直接返回到循环体起始处，判断循环控制条件，以决定是否进入下一次循环。也就是说，continue语句的作用是终止本次循环，忽略循环体内continue语句后面的所有语句，直接回到循环的顶端，以进入可能的下一次循环。

函数 Function



本节大纲

- **内置函数**
- **练习题**

内置函数

1. 内置函数 (Bulit-In Function, BIF)

内置函数是Python的“贴身”函数，Python在哪里，内置函数就跟到哪里，用户无须经过任何特殊手续，就可以放心大胆地去调用它们。例如，input()、print()等都属于内置函数。

名称	说明	举例
int()	把字符串型数字转换成数字	x='12' y=int(x) y=12
hex()	把整数转换为字符串形式的十六进制数	x=125 y=hex(x) y='0x7d'
oct()	把整数转换为字符串形式的八进制数	x=24 y=oct(x) y='0o30'
float()	把数字转换成浮点数形式	x=78 y=float(x) y=78.0
str()	把数字转换成字符串形式	x=123 y=str(x) y='123'
iter()	不断读取参数中的元素，直至按序读完	data=[44,66,25] iter(data)
range()	依照索引，不断将调用者中的元素输出	list(range(4,10))
len()	返回调用者中元素的个数（长度）	num=[78,42,93] len(num)

续表

名称	说明	举例
<code>slice()</code>	利用[]运算符，提取字符串中的切片	<code>wd='function' wd1=slice(2,5,3)</code>
<code>sorted()</code>	对调用者的元素进行升序或降序排序	<code>data=[44,66,25] sorted(data)</code>
<code>type()</code>	返回调用者的类型	<code>x=56 type(x)</code>
<code>print()</code>	输出字符到屏幕上	
<code>input()</code>	获取输入的数据	

2. 标准函数库 (Standard Library)

Python分门别类地提供很多标准函数库，每个库里的众多函数，都专注于完成某一个方面所需要的功能。例如，`math`库提供的是数学方面需要的功能模块；`cmath`库涉及的是处理复数的模块；`random`库是各种产生随机数值的功能模块；`time`、`calendar`库涉及的是有关时间、日历等的模块；等等。

模块



本节大纲

- **标准模块**
- **第三方模块**

标准模块

模块	描述
os	操作系统管理
sys	解释器交互
platform	操作系统信息
glob	查找文件
shutil	文件管理
random	随机数
subprocess	执行Shell命令
pickle	对象数据持久化
json	JSON编码和解码
time	时间访问和转换
datetime	日期和时间
urllib	HTTP访问

官方文档标准库列表: <https://docs.python.org/zh-cn/3.8/library/index.html>

os库

方法	描述
os.name	返回操作系统类型
os.environ	以字典形式返回系统变量
os.putenv(key, value)	改变或添加环境变量
os.listdir(path=' . ')	列表形式列出目录下所有目录和文件名
os.getcwd()	获取当前路径
os.chdir(path)	改变当前工作目录到指定目录
os.mkdir(path [, mode=0777])	创建目录
os.makedirs(path [, mode=0777])	递归创建目录
os.rmdir(path)	移除空目录，不能删除有文件的目录
os.remove(path)	移除文件
os.rename(old, new)	重命名文件或目录
os.stat(path)	获取文件或目录属性
os.chown(path, uid, gid)	改变文件或目录所有者
os.chmod(path, mode)	改变文件访问权限
os.symlink(src, dst)	创建软链接
os.unlink(path)	移除软链接
os.getuid()	返回当前进程UID
os.getlogin()	返回登录用户名
os.getpid()	返回当前进程ID
os.kill(pid, sig)	发送一个信号给进程
os.walk(path)	目录树生成器，生成一个三组 (dirpath, dirnames, filenames)
os.system(command)	执行shell命令，不能存储结果
os.popen(command [, mode='r' [, bufsize]])	打开管道来自shell命令，并返回一个文件对象

sys库

方法	描述
sys.argv	从程序外部传递参数 argv[0] #代表本身名字 argv[1] #第一个参数 argv[2] #第二个参数 argv[3] #第三个参数 argv[N] #第N个参数 argv #参数以空格分隔存储到列表
sys.exit([status])	退出Python解释器
sys.path	当前Python解释器查找模块搜索的路径，列表返回。
sys.getdefaultencoding()	获取系统当前编码
sys.platform	返回操作系统类型
sys.version	获取Python版本

<https://blog.csdn.net/angqiang>

.....

第三方模块

可以在 The Python Package Index (PyPI) 软件库（官网主页：<https://pypi.org/>）查询、下载 和 发布 Python包或库。



第三方模块

网络爬虫

requests: <https://pypi.org/project/requests/> 简洁且简单的处理HTTP请求的第三方库

scrapy: <https://scrapy.org/> 快速、高层次的Web获取框架

数据分析

numpy: <http://www.numpy.org/> 开源数值计算扩展第三方库

scipy: <https://pypi.org/project/scipy/> 专为科学以及工程计算的第三方库

pandas: <http://pandas.pydata.org/> 可高效地操作大型数据集的第三方库

pdfminer: <https://pypi.org/project/pdfminer/> 从PDF文档中提取各类信息的第三方库

openpyxl: <https://pypi.org/project/openpyxl/> 处理Microsoft Excel文档的Python第三方库

python-docx: <https://pypi.org/project/python-docx/> 处理Microsoft Word文档的Python第三方库

beautifulsoup4: <https://pypi.org/project/beautifulsoup4/> 从HTML和XML文件中解析出数据的第三方库

用户图形界面

PyQt5: <https://pypi.org/project/PyQt5/> 成熟的商业级GUI第三方库

wxpython: <https://pypi.org/project/wxPython/> 优秀的GUI图形库

pygtk: <https://pypi.org/project/PyGTK/> 轻松创建具有图形用户界面程序的第三方库

机器学习

Scikit-learn: <https://scikit-learn.org/stable/> 简单且高效的数据挖掘和数据分析工具

Tensorflow: <https://pypi.org/project/tensorflow/> 人工智能学习系统

Theano: <http://deeplearning.net/software/theano/> 执行深度学习中大规模神经网络算法的运算