Python程序设计

陈远祥

chenyxmail@gmail.com

北京邮电大学 电子工程学院

Anaconda

- Anaconda： anaconda指的是一个开源的Python发行版本，其包含了conda、Python等180多个科学包及其依赖项
- Anaconda和Jupyter notebook已成为数据分析的标准环境，简单来说，Anaconda是包管理器和环境管理器，Jupyter notebook 可以将数据分析的代码、图像和文档全部组合到一个web文档中

Anaconda

- Anaconda在英文中是“蟒蛇”，Nicki Minaj（妮琪·米娜）有首歌就叫《Anaconda》，表示像蟒蛇一样性感妖娆的身体

Anaconda

Anaconda

- 你可能已经安装了 Python，那么为什么还需要 Anaconda？
  - ✓ Anaconda 附带了一大批常用数据科学包，它附带了 conda、Python 和 150 多个科学包及其依赖项。因此你可以立即开始处理数据

Anaconda

- 管理包
  - ✓ Anaconda是在conda（一个包管理器和环境管理器）上发展出来的
  - ✓ 在数据分析中，你会用到很多第三方的包，而conda（包管理器）可以很好的帮助你在计算机上安装和管理这些包，包括安装、卸载和更新包

Anaconda

- 管理环境，为什么需要管理环境呢？
  - ✓ 比如你在A项目中用了 Python 2，而新的项目B要求使用Python 3，而同时安装两个Python版本可能会造成许多混乱和错误。这时候 conda就可以帮助你为不同的项目建立不同的运行环境
  - ✓ 还有很多项目使用的包版本不同，比如不同的pandas版本，不可能同时安装两个 Numpy 版本，你要做的应该是，为每个 Numpy 版本创建一个环境，然后项目的对应环境中工作，这时候conda就可以帮你做到

Anaconda

- 调试环境，Spyder编辑器
  - ✓ 对于初学者非常有用

Anaconda

- Anaconda 可用于多个平台（ Windows、Mac OS X 和 Linux）
- 你可以在下面地址上找到安装程序和安装说明。根据你的操作系统选择对应的版本下载
- 下载地址
  - ✓ https://www.continuum.io/downloads

Anaconda

- Anaconda 的下载文件比较大（约 500 MB），因为它附带了 Python 中最常用的数据科学包。如果计算机上已经安装了 Python，安装不会对你有任何影响。实际上，脚本和程序使用的默认 Python 是 Anaconda 附带的 Python

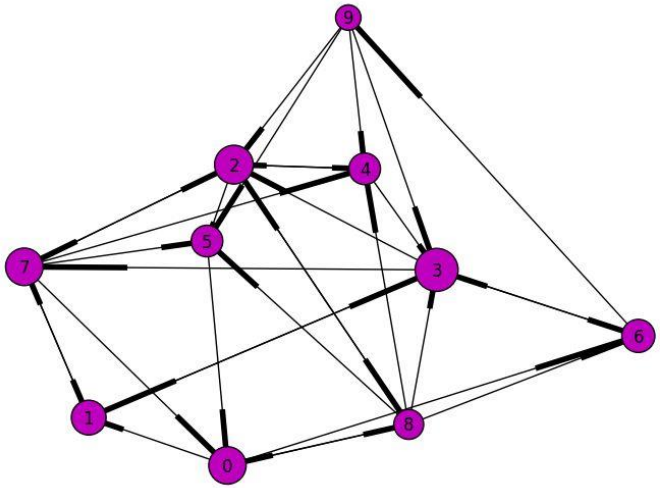
上周主要内容

- 函数和代码的复用

本周主要内容

- 组合数据类型及其操作

简单数据类型



简单数据类型

- 空（None）

表示该值是一个空对象，空值是Python里一个特殊的值，用None表示。None不能理解为0，因为0是有意义的，而None是一个特殊的空值

简单数据类型

- 布尔类型（Boolean）

一个布尔值只有True、False两种值，要么是True，要么是False，在Python中，可以直接用True、False表示布尔值（请注意大小写）

简单数据类型

- Python提供了常用的数字类型：整数、浮点数以及与之相关的语法和操作
- 允许使用八进制、十六进制常量
- 提供了复数类型
- 提供了无穷精度的长度类型（只要内存空间允许，可以增长成为任意位数的整数）

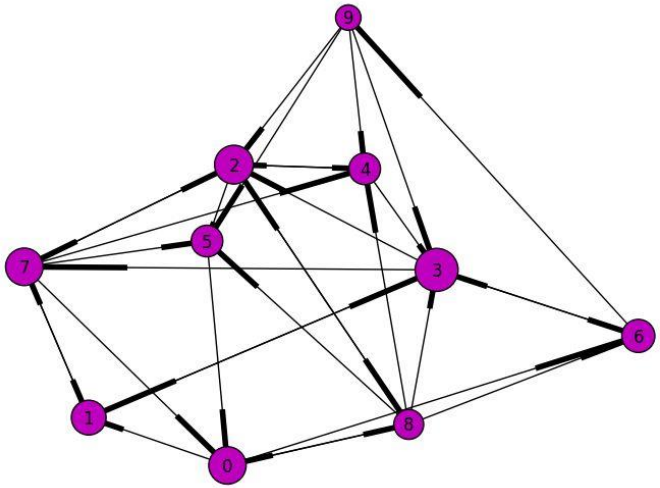
简单数据类型

- 表达式操作符：+、-、*、/、**
- 内置数学函数：pow、abs
- 公用模块：random、math等
- 专业扩展NumPy：矩阵、向量处理等

数学内置函数和内置模块

- random模块：用于产生随机数
  - ✓ #random

组合数据类型



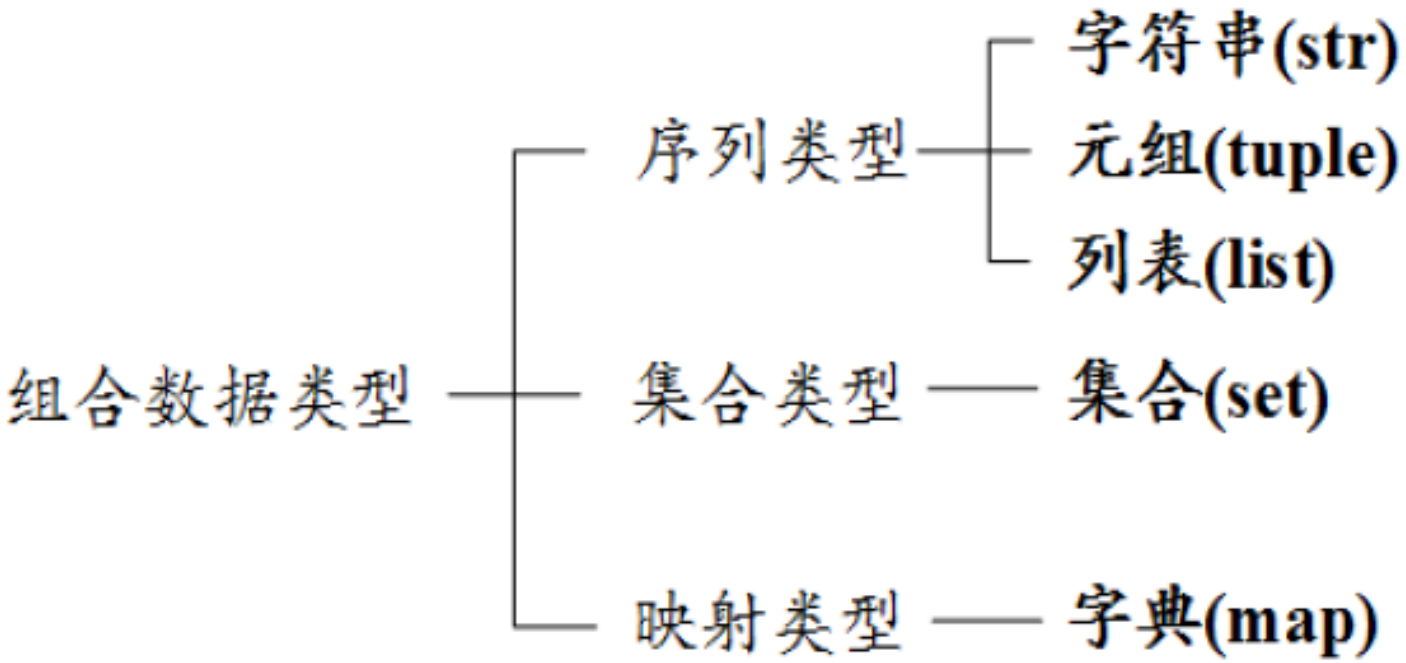
组合数据类型

- 计算机不仅对单个变量表示的数据进行处理，更多情况，计算机需要对一组数据进行批量处理。一些例子包括：
  - ✓ 给定一组单词{python, data, function}，计算并输出每个单词的长度
  - ✓ 给定一个学院学生信息，统计一下男女生比例
  - ✓ 一次实验产生了很多组数据，对这些大量数据进行分析

组合数据类型

- 组合数据类型能够将多个同类型或不同类型的数据组织起来，通过单一的表示使数据操作更有序更容易
- 根据数据之间的关系，组合数据类型可以分为三类：
  - ✓ 序列类型、集合类型和映射类型

组合数据类型



组合数据类型

- 序列类型是一个元素向量，元素之间存在先后关系，通过序号访问，元素之间不排他
- 集合类型是一个元素集合，元素之间无序，相同元素在集合中唯一存在
- 映射类型是“键-值”数据项的组合，每个元素是一个键值对，表示为(key,value)

字符串定义

- 字符串在Python被看成是单个字符的序列，具有序列对象的特殊功能，字符串是固定的，不可变的
- 可在字符串中使用单引号和双引号，注意要搭配。如‘boy’，”girl”等
- 字符串内部的一个反斜杠“\”可允许把字符串放于多行
- 也可以使用三个’或”使字符串跨行

字符串基本操作

- 可以用for语句在一个字符串中进行迭代，并使用in表达式操作符进行成员关系的测试，这实际上是一种搜索

```
>>> s = 'hello'
>>> for c in s:
...     print c,
...
h e l l o
>>> "h" in s
True
>>> "b" in s
False
```

- for循环指派了一个变量去获取一个序列其中的元素，并对每一个元素执行一个或多个语句，变量c相当于在字符串中步进的指针

索引和分片的总结

- 分片（s[i:j）提取对应的部分作为一个序列，上边界并不包含在内
- s[1:]获取了从偏移为1直到末尾之间的元素
- s[:-1]获取从偏移为0直到但不包含最后一个元素之间的元素
- s[:]获取从偏移为0直到末尾之间的所有元素

字符串转化

- Python不允许字符串和数字直接相加
- 有意设计的，因为+既能够进行加法运算也能够进行合并运算，这样的语法会变得模棱两可，因此，Python将其作为错误处理，在Python中，如果让操作变得复杂或含糊，就会避免这样的语法

字符串转化

- 如果用户从文件或用户界面得到一个作为字符串的数字，怎么把这个字符串变为数字型呢？这就用到类型的转换函数
  - ✓ str()
  - ✓ int()
  - ✓ float()
  - ✓ eval('12 + 3')：将字符串str当成有效的表达式来求值并返回计算结果

修改字符串

- 缺省情况下，字符串对象是“不可变序列”，不可变的意思是不能实地的修改一个字符串
  - ✓ #字符串修改
- 那如何改变一个字符串呢？这就要利用合并、分片这样的工具来建立并赋值给一个新的字符串；必要的话，可以将结果赋值给字符串最初的变量名

修改字符串

- 每修改一次字符串就生成一个新的字符串对象，这看起来好像会造成效率下降，其实，在Python内部会自动对不再使用的字符串进行垃圾回收，所以，新的对象重用了前面已有字符串的空间
- Python的效率比我们想象的要好

List列表

- 列表是Python中最具灵活性的有序集合对象类型。和字符串不同的是，列表可以包含任何种类的对象：数字、字符串、自定义对象甚至其他列表
- 与其他高级语言的数组列表是可变对象相似，支持在原处修改，可以通过指定的偏移值和分片、列表方法调用、删除语句等方法实现

列表的主要性质

- 任意对象的有序集合
  - ✓ 从功能是看，列表就是收集其他对象的地方，可以他们看成数组；同时，列表所包含的每一项都保持了从左到右的位置顺序（也就是说，它们是序列）
- 通过偏移读取
  - ✓ 和字符串一样，可以通过列表对象的偏移对其进行索引，从而读取对象的一部分内容。当然也可以执行诸如分片和合并之类的操作

列表的主要性质

- 可变长度、异构以及任意嵌套
  - ✓ 和字符串不同，列表可以根据需要增长或缩短（长度可变），并且可以包含任何类型的对象，并支持任意的嵌套
- 可变序列
  - ✓ 列表支持在原处的修改，也可以响应所有指针对字符串序列的操作，如索引、分片以及合并

常用列表常量和操作

操作	解释
`L1=[]`	一个空的列表
`L2 = [0, 1, 2, 3]`	四元素列表
`L3 = ['abc',10,['def', 'ghi']]`	嵌套列表
`L2[i]`	索引
`L3[i][j]`	索引的索引
`L2[i:j]`	分片
`len(L2)`	求长度
`L1 + L2`	合并
`L2 * 3`	重复

List对象的操作

方法	描述
append(x)	在列表尾部追加单个对象x。使用多个参数会引起异常。
count(x)	返回对象x在列表中出现的次数。
extend(L)	将列表L中的表项添加到列表中。返回None。
index(x)	返回列表中匹配对象x的第一个列表项的索引。无匹配元素时产生异常。
insert(i,x)	在索引为i的元素前插入对象x。如list.insert(0,x)在第一项前插入对象。返回None。
pop(x)	删除列表中索引为x的表项，并返回该表项的值。若未指定索引，pop返回列表最后一项。
sort()	对列表排序，返回none。bisect模块可用于排序列表项的添加和删除。
remove(x)	删除列表中匹配对象x的第一个元素。匹配元素时产生异常。返回None。
reverse()	颠倒列表元素的顺序。

List元素的增加

- 可以使用“+”运算符来实现将元素添加到列表中的功能

```
alist = [3,4,5]
alist = alist+[6]
```

- 这并不是真的为列表添加元素，而是创建一个新列表，并将原列表中的元素和新元素依次复制到新列表的内存空间
- 由于涉及大量元素的复制，该操作速度较慢，在涉及大量元素添加时不建议使用该方法

List元素的增加

- 使用列表对象的append()方法，原地修改列表，是真正意义上的在列表尾部添加元素，速度较快，也是推荐使用的方法
  - ✓ append时间

File Edit Format Run Options Window Help

```
import time #调用time模块
result1 = []#创建空列表
start = time.time()
#返回当前时间的时间戳
#1970年1月1日以来到现在所经过的秒数
#1970年1月1日是Unix纪元，
#而很多操作系统和编程语言都和unix或多或少
#有些关系，所以就约定从这天开始的秒数来
#表示Unix纪元之后的时刻
for i in range(100000):
    result1 = result1 + [i]
print(len(result1), ',', time.time()-start)
result2 = []
start = time.time()
for i in range(100000):
    result2.append(i)
print(len(result2), ',', time.time()-start)
```

List元素的增加

- 自动内存管理方式；

当为对象修改值时，并不是真的直接修改变量的值，而是使变量指向新的值，这对于Python所有类型的变量都是一样的

  - ✓ #id

List元素的增加

- 对于列表、集合、字典等可变序列类型而言，列表中包含的是元素值的引用，而不是直接包含元素值
- 如果直接修改序列变量的值，则与Python普通变量的情况是一样的，而如果是通过下标来修改序列中元素的值或通过可变序列对象自身提供的方法来增加和删除元素时，序列对象在内存中的起始地址是不变的，仅仅是被改变值的元素地址发生变化
  - ✓ id2

List元素的删除

- 使用列表对象的remove()方法删除首次出现的指定元素，如果列表中不存在要删除的元素，则抛出异常
  - ✓ remove1

List元素的删除

- 删除列表中指定元素的所有重复
  - ✓ 循环+remove
  - ✓ remove2
  - ✓ remove3

File Edit Format Run Options Window Help

```
#x = [1, 2, 1, 2, 1]
x = [1, 2, 1, 1, 2]
for i in x:
        if i == 1:
                x.remove(i)
```

File Edit Format Run Options Window Help

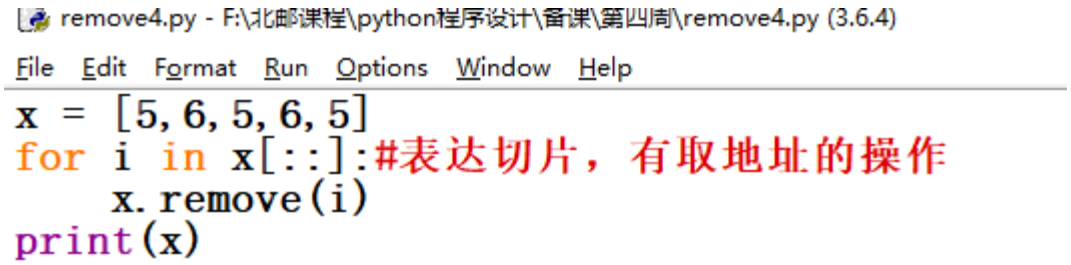
```
x = [1, 2, 3, 4, 5]
for i in x:#x[0]索引
    x.remove(i)
print(x)
```

List元素的删除

- 列表的自动内存管理功能，在删除列表元素时，Python会自动对列表内存进行收缩并移动列表元素以保证所有元素之间没有空隙，增加列表元素时也会自动扩展内存并对元素进行移动以保证元素之间没有空隙。每当插入或删除一个元素之后，该元素位置后面所有元素的索引就都改变了

List元素的删除

- 正确的代码
  - ✓ remove4



把列表当作堆栈使用

- 列表方法使得列表可以很方便的做为一个堆栈来使用，堆栈作为特定的数据结构，最先进入的元素最后一个被释放（后进先出）
- 用append() 方法可以把一个元素添加到堆栈顶。用不指定索引的pop() 方法可以把一个元素从堆栈顶释放出来
  - ✓ #堆栈

把列表当作队列使用

- 也可以把列表当做队列使用，队列作为特定的数据结构，最先进入的元素最先释放（先进先出）
- 使用append()方法可以把元素添加到队列最后，以0为参数调用pop() 方法可以把最先进入的元素释放出来
  - ✓ #队列

元组

- 一个元组由数个逗号分隔的值组成
- 元组在输出时总是有括号的，以便于正确表达嵌套结构
- 在输入时，有或没有括号都可以，不过经常括号都是必须的（如果元组是一个更大的表达式的一部分）
- 构造包含零个或一个元素的元组：为了适应这种情况，语法上有一些额外的改变。一对空的括号可以创建空元组；要创建一个单元素元组可以在值后面跟一个逗号

元组

- 元组就像字符串，不可改变：不能给元组的一个独立的元素赋值（尽管可以通过联接和切片来模仿）
- 元组有很多用途：例如(x, y)坐标点，数据库中的员工记录等等
- 可以通过包含可变对象来创建元组，例如列表
  - ✓ `lst = [1, 2, 3]`
  - ✓ `t = tuple(lst)`

元组

- 元组中的数据一旦定义就不允许更改
- 元组没有append()、extend()和insert()等方法，无法向元组中添加元素
- 元组没有remove()或pop()方法，也无法对元组元素进行del操作，不能从元组中删除元素
- 内建的tuple( )函数接受一个列表参数，并返回一个包含同样元素的元组，而list( )函数接受一个元组参数并返回一个列表。从效果上看，tuple( )冻结列表，而list( )融化元组

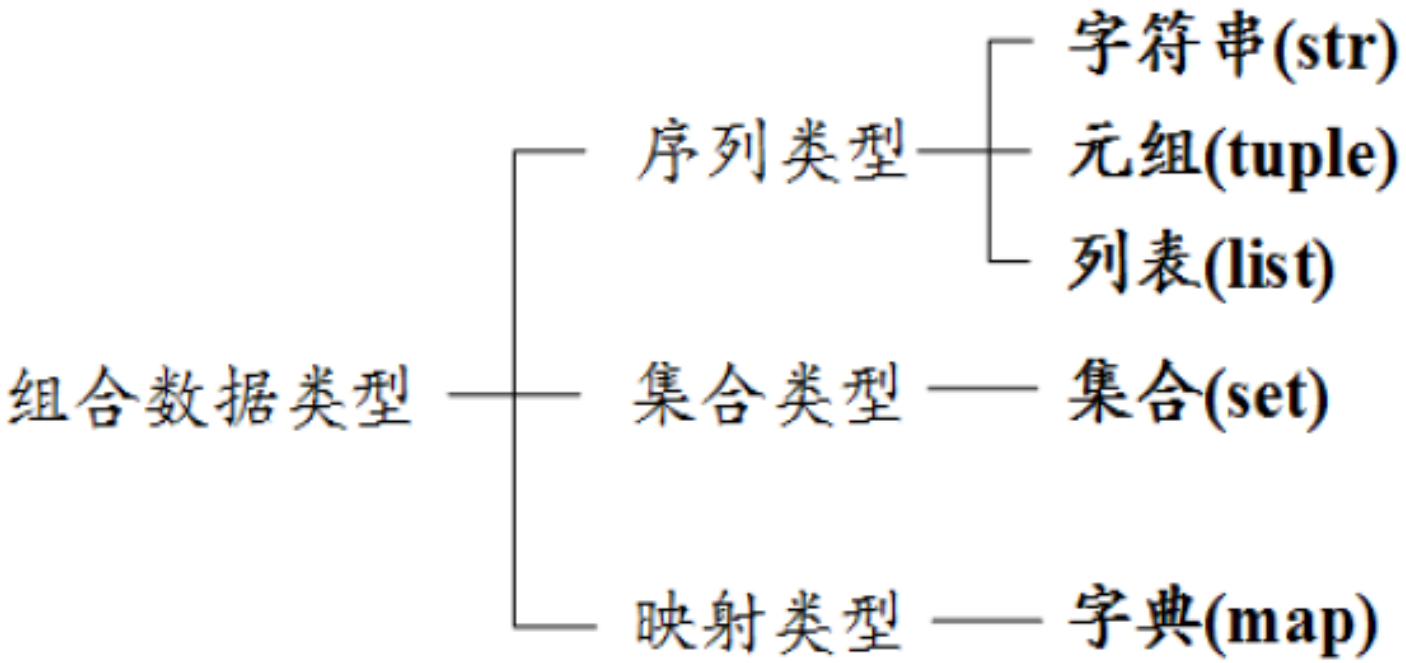
元组封装和解封

- 语句t = 1，2，3 是元组封装的一个例子：值1，2，3被封装进元组。其逆操作是这样：

```
>>> t = (1, 2, 3)
>>> x, y, z = t
>>> print x, y, z
1 2 3
```

- 这个调用被称为序列拆封非常合适，序列拆封要求左侧的变量数目与序列的元素个数相同

组合数据类型



字典

- 通过任意键信息查找一组数据中值信息的过程叫映射，Python语言中通过字典实现映射
- 字典（Dictionary）：字典在某些语言中可能称为“联合内存” 或“联合数组”
- 字典类似于通过联系人名字查找地址和联系人详细情况的地址簿，即：我们把键（名字）和值（详细情况）联系在一起
- 键必须是唯一的，就像如果有两个人恰巧同名的话，将无法找到正确的信息

字典

- 举例：检索学生信息
  - ✓ “<键><值>对”
  - ✓ 键（即身份证号码）
  - ✓ 值（即学生信息）
- “键值对”例子
  - ✓ 姓名和电话号码
  - ✓ 用户名和密码
  - ✓ 国家名称和首都等

字典

- Python语言中的字典可以通过大括号（{}）建立，建立模式如下：

  {<键1>:<值1>,<键2>:<值2>,…,<键n>:<值n>}

- 其中，键和值通过冒号连接，不同键值对通过逗号隔开
  - ✓ #字典1

字典

- 字典最主要的用法是查找与特定键相对应的值，这通过索引符号来实现
- 一般来说，字典中键值对的访问模式如下，采用中括号格式：

  <值>=<字典变量>[<键>]

- 字典中对某个键值的修改可以通过中括号的访问和赋值实现

字典

- 序列是以连续的整数为索引，与此不同的是，字典以关键字为索引
- 关键字可以是任意不可变类型，通常用字符串或数值。如果元组中只包含字符串和数字，它可以做为关键字，如果它直接或间接的包含了可变对象，就不能当做关键字
- 不能用列表做关键字，因为列表可以用它们的append()和extend()方法，或者用切片、或者通过检索变量来即时改变
- 基本说来，应该只使用简单的对象作为键

字典

- 字典是无序的(关键字:值)对集合，关键字必须是互不相同的
- 如果使用一个已经存在的关键字存储新的值或对象，以前为该关键字分配的值就会被遗忘
- 试图析取从一个不存在的关键字中读取值会导致错误

字典

- 通过中括号可以增加新的元素
- 直接使用大括号（{}）可以创建一个空的字典，并通过中括号（[]）向其增加元素
- 字典的用del来删除(关键字:值)对
- 不允许同一个键出现两次。创建时如果同一个键被赋值两次，后一个值会被记住

字典

方法	描述
keys()	返回字典中键的列表
values()	返回字典中值的列表。
items()	返回tuples的列表。每个tuple由字典的键和相应值组成。
clear()	删除字典的所有条目。
copy()	返回字典高层结构的一个拷贝，但不复制嵌入结构，而只复制对那些结构的引用。
update(x)	用字典x中的键值对更新字典内容。
get(x[,y])	返回键x，若未找到该键返回none，若提供y，则未找到x时返回y。

字典应用

- 随机产生1000个字符串，统计每个字符出现次数
  - ✓ #统计字符串出现次数

统计字符串出现的字数.py - F:\北邮课程\python程序设计\备课\第四周\统计字符串出现的字数.py (3.6.4)

File Edit Format Run Options Window Help

```
import string#导入string模块
import random#导入random模块
x = string.ascii_letters + string.digits + string.punctuation#生成字符串
y = [random.choice(x) for i in range(1000)]#随机选择1000个数字
"""y = []
for i in range(N):
y.append(random.choice(range(n)))
"""
z = ''.join(y)#将列表转换为字符串
d = dict()
for ch in z:
        d[ch] = d.get(ch, 0) + 1#不断统计，更新键的值,之前没有值则计1，有则取出
```

字典应用

- 建立通讯录程序，实现查找和更新联系人信息
  - ✓ #通讯录

File Edit Format Run Options Window Help

```
print('''|---欢迎进入通讯录程序---|
|---1、 查询联系人资料---|
|---2、 插入新的联系人---|
|---3、 删除已有联系人---|
|---4、 退出通讯录程序---|''')
addressBook={}#字典，定义通讯录
while 1:
    temp=input('请输入指令代码：')
    if not temp.isdigit():#检测字符串是否只由数字组成
        print("输入的指令错误，请按照提示输入")
        continue
    item=int(temp)#转换为数字
    if item==4:
        print("|---感谢使用通讯录程序---|")
        break
    name = input("请输入联系人姓名:")
    if item==1:
        if name in addressBook:
            print(name,':',addressBook[name])
            continue
        else:
            print("该联系人不存在！")
    if item==2:
        if name in addressBook:
            print("您输入的姓名在通讯录中已存在-->>",name,":",addressBook[name])
            isEdit=input("是否修改联系人资料(Y/N)：")
            if isEdit=='Y' or 'y':
                userphone = input("请输入联系人电话：")
                addressBook[name]=userphone
                print("联系人修改成功")
                continue
            else:
                continue
        else:
            userphone=input("请输入联系人电话：")
            addressBook[name]=userphone
            print("联系人加入成功！")
            continue

    if item==3:
        if name in addressBook:
            del addressBook[name]
            print("删除成功！")
            continue
        else:
            print("联系人不存在")
```

谢谢