Python程序设计

陈远祥

chenyxmail@gmail.com

北京邮电大学 电子工程学院

上周主要内容

- 组合数据类型及其操作

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

- 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

- 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。

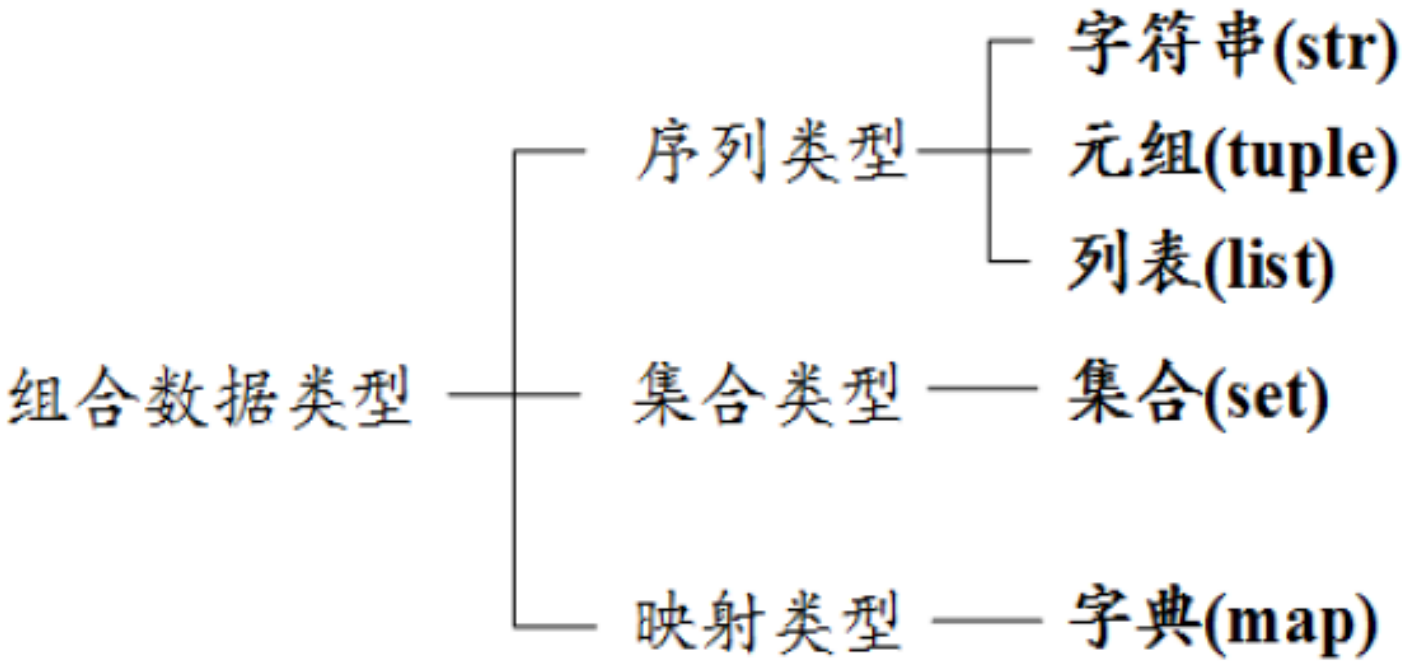
字典应用

- 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

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

组合数据类型



集合

- 集合（set）是无序可变集合，使用一对大括号界定
- 集合类型与数学中集合的概念一致，即包含0个或多个数据项的无序组合。集合中元素不可重复，元素类型只能是固定数据类型，例如：整数、浮点数、字符串、元组等
- 列表、字典和集合类型本身都是可变数据类型，不能作为集合的元素出现

集合

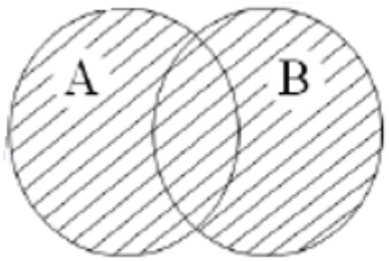
- 由于集合是无序组合，它没有索引和位置的概念，不能分片，集合中元素可以动态增加或删除
- 集合用大括号（{}）表示，可以用赋值语句生成一个集合
  - ✓ #集合

集合

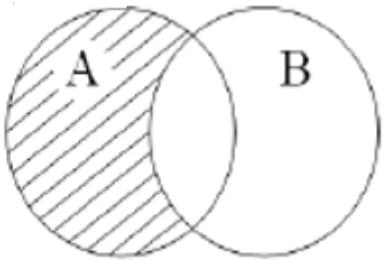
- set(x)函数可以用于生成集合
- 由于集合元素是无序的，集合的打印效果与定义顺序可以不一致
- 由于集合元素独一无二，使用集合类型能够过滤掉重复元素

集合

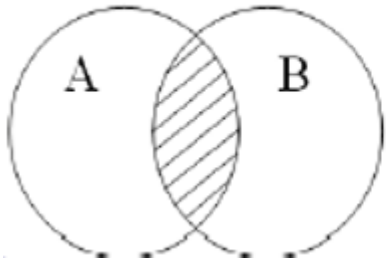
- 集合类型的4种基本操作，交集（&）、并集（|）、差集（-）、补集（^），操作逻辑与数学定义相同



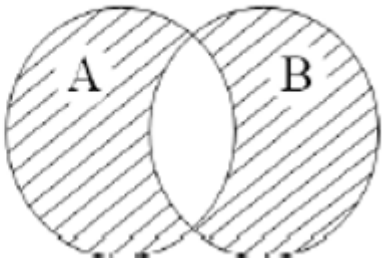
A|B



A-B



A&B



A^B

集合

- 集合类型有10个操作符

操作符	描述
S – T 或 S.difference(T)	返回一个新集合，包括在集合S中但不在集合T中的元素
S-=T或S.difference_update(T)	更新集合S，包括在集合S中但不在集合T中的元素
S & T或S.intersection(T)	返回一个新集合，包括同时在集合S和T中的元素
S&=T或S.intersection_update(T)	更新集合S，包括同时在集合S和T中的元素。
S^T或s.symmetric_difference(T)	返回一个新集合，包括集合S和T中元素，但不包括同时在其中的元素
S=^T或s.symmetric_difference_update(T)	更新集合S，包括集合S和T中元素，但不包括同时在其中的元素
S|T或S.union(T)	返回一个新集合，包括集合S和T中所有元素
S=|T或S.update(T)	更新集合S，包括集合S和T中所有元素
S<=T或S.issubset(T)	如果S与T相同或S是T的子集，返回True，否则返回False，可以用S<T判断S是否是T的真子集
S>=T或S.issuperset(T)	如果S与T相同或S是T的超集，返回True，否则返回False，可以用S>T判断S是否是T的真超集

集合

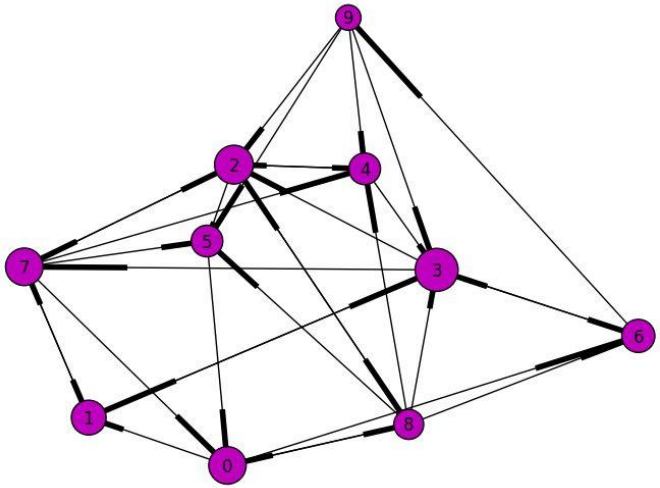
- 集合类型有10个操作函数或方法

函数或方法	描述
S.add(x)	如果数据项x不在集合S中，将x增加到s
S.clear()	移除S中所有数据项
S.copy()	返回集合S的一个拷贝
S.pop()	随机返回集合S中的一个元素，如果S为空，产生KeyError异常
S.discard(x)	如果x在集合S中，移除该元素；如果x不在，不报错
S.remove(x)	如果x在集合S中，移除该元素；不在产生KeyError异常
S.isdisjoint(T)	如果集合S与T没有相同元素，返回True
len(S)	返回集合S元素个数
x in S	如果x是S的元素，返回True，否则返回False
x not in S	如果x不是S的元素，返回True，否则返回False

集合

- 集合类型主要用于三个场景：成员关系测试、元素去重和删除数据项
- 集合类型与其他类型最大的不同在于它不包含重复元素，因此，当需要对一维数据进行去重或进行数据重复处理时，一般通过集合来完成
  - ✓ #集合

jieba库



jieba

- jieba是Python中一个重要的第三方中文分词函数库
- 第三方库，需要安装

函数	描述
jieba.cut(s)	精确模式，返回一个可迭代的数据类型
jieba.cut(s, cut_all=True)	全模式，输出文本s中所有可能单词
jieba.cut_for_search(s)	搜索引擎模式，适合搜索引擎建立索引的分词结果
jieba.lcut(s)	精确模式，返回一个列表类型，建议使用
jieba.lcut(s, cut_all=True)	全模式，返回一个列表类型，建议使用
jieba.lcut_for_search(s)	搜索引擎模式，返回一个列表类型，建议使用
jieba.add_word(w)	向分词词典中增加新词w

jieba

- 支持三种分词模式：
  - ✓ 精确模式，试图将句子最精确地切开，适合文本分析
  - ✓ 全模式，把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义
  - ✓ 搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词
  - ✓ #jieba库

jieba

- 文本词频统计
  - ✓ 三国人物

三国人物1.py - F:\北邮课程\Python程序设计\备课\第五周\三国人物1.py (3.6.4)

File Edit Format Run Options Window Help

```
import jieba
f=open("三国演义.txt","rb")
txt=f.read()
words  = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1:  #排除单个字符的分词结果
        continue
    else:
        counts[word] = counts.get(word,0) + 1

items = list(counts.items())
items.sort(key=lambda items:items[1], reverse=True)
for i in range(20):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

jieba

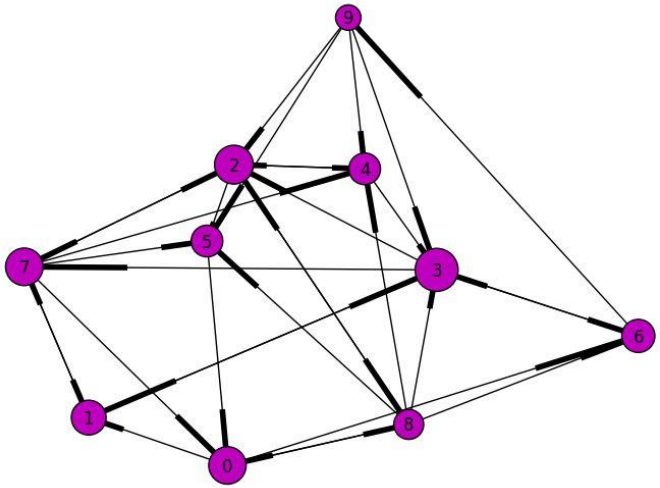
文本词频统计

三国人物2.py - F:\北邮课程\Python程序设计\备课\第五周\三国人物2.py (3.6.4)

File Edit Format Run Options Window Help

```
import jieba
excludes = {"将军","却说","荆州","二人","不可","不能","如此"}
f=open("三国演义.txt","rb")
txt=f.read()
words  = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1:
        continue
    elif word == "诸葛亮" or word == "孔明曰":
        rword = "孔明"
    elif word == "关公" or word == "云长":
        rword = "关羽"
    elif word == "玄德" or word == "玄德曰":
        rword = "刘备"
    elif word == "孟德" or word == "丞相":
        rword = "曹操"
    else:
        rword = word
    counts[rword] = counts.get(rword,0) + 1
for word in excludes:
    del(counts[word])
items = list(counts.items())
items.sort(key=lambda items:items[1], reverse=True)
for i in range(10):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

字符串格式化



字符串格式化

- 为什么需要字符串格式化？

  一个程序希望输出如下内容：

  “2016-12-31：计算机PYTHON的CPU占用率为10%”

- 下划线内容可能会变化，需要由特定函数运算结果进行填充，最终形成上述格式字符串作为输出结果

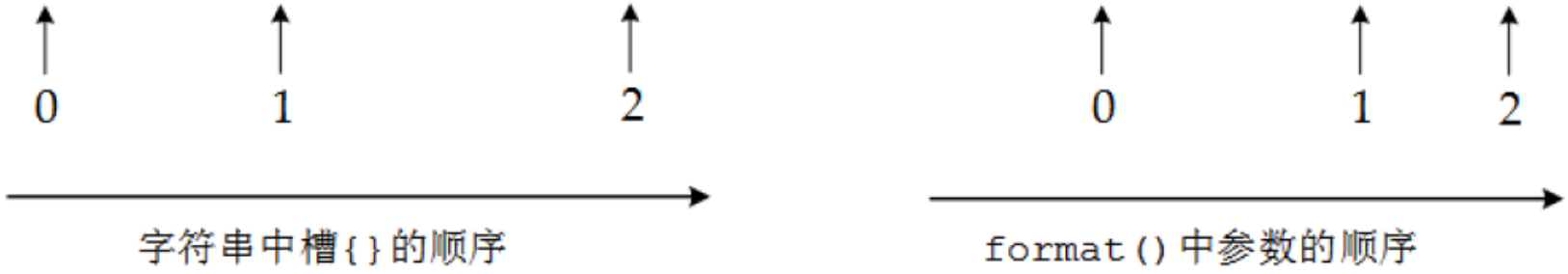
- 字符串格式化用于解决字符串和变量同时输出时的格式安排

字符串格式化

- format()方法的基本使用
- 字符串format()方法的基本使用格式是：

  <模板字符串>.format(<逗号分隔的参数>)

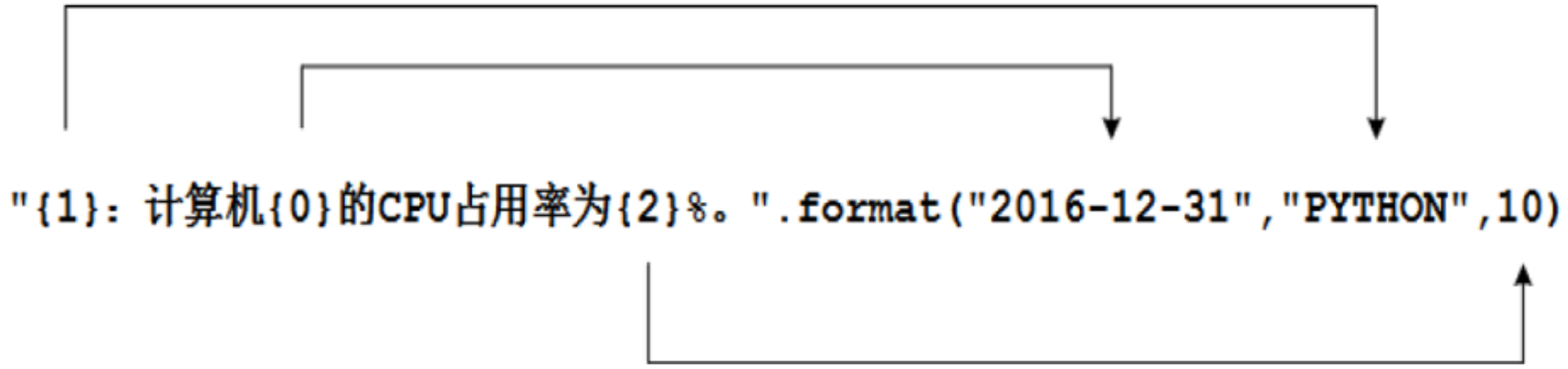
```
"{ }：计算机{ }的CPU占用率为{ }%。".format("2016-12-31","PYTHON",10)
```



- ✓ #格式化

字符串格式化

- 如果大括号中指定了使用参数的序号，按照序号对应参数替换



字符串格式化

- format()方法中模板字符串的槽除了包括参数序号，还可以包括格式控制信息。此时，槽的内部样式如下：{<参数序号>:<格式控制标记>}
- 格式控制标记用来控制参数显示时的格式。格式控制标记包括：<填充><对齐><宽度>,<.精度><类型>6个字段，这些字段都是可选的，可以组合使用

:	<填充>	<对齐>	<宽度>	,	<.精度>	<类型>
引导符号	用于填充的单个字符	< 左对齐 > 右对齐 ^ 居中对齐	槽的设定输出宽度	数字的千位分隔符 适用于整数和浮点数	浮点数小数部分的精度 或 字符串的最大输出长度	整数类型 b, c, d, o, x, X 浮点数类型 e, E, f, %

✓ 格式化1

字符串格式化

- 文本进度条
  - ✓ 基本思想是按照任务执行百分比将整个任务划分为100个单位，每执行N%输出一次进度条。每一行输出包含进度百分比，代表已完成的部分(**)和未完成的部分(..)的两种字符，以及一个跟随完成度前进的小箭头，风格如下：

75% [**********************************->.....]

  - ✓ 利用print()函数实现简单的非刷新文本进度条
  - ✓ 进度条1

字符串格式化

- 文本进度条
  - ✓ 进度条1

File Edit Format Run Options Window Help

```
import time#引入时间模块
scale = 20 #从0%开始到100%, 共21个进度条
print("------执行开始------")
for i in range(scale+1):
    a, b = '**' * i,'..' * (scale - i)
    c = (i/scale)*100
    print("{:>3.0f}%[{}->{}]" .format (c, a, b))
    time.sleep(0.5)#让程序休眠
print("------执行结束------")
```

字符串格式化

- 单行动态刷新
  - ✓ 进度条2

File Edit Format Run Options Window Help

```
import time
for i in range(101):
    print("\r{:2}%".format(i), end="")#表示将输出的内容返回到第一个指针
    time.sleep(0.05)
```

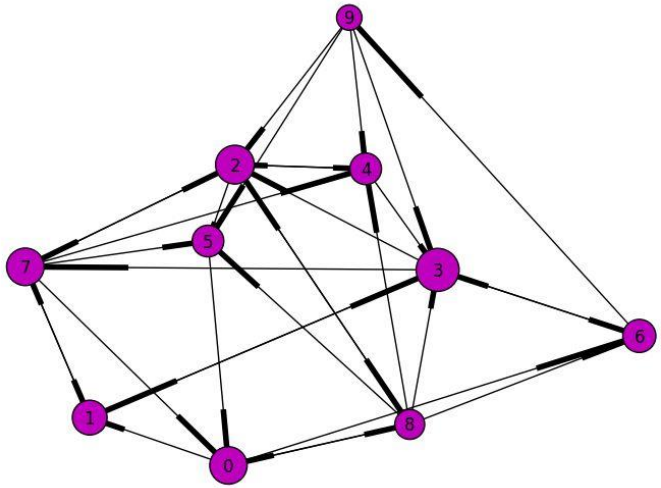
字符串格式化

- 单行动态刷新
  - ✓ 进度条3

File Edit Format Run Options Window Help

```
import time
scale = 50
print("执行开始".center(scale//2,'-'))
t = time.clock()
for i in range(scale+1):
    a = '*' * i
    b = '.' * (scale - i)
    c = (i/scale)*100
    t -= time.clock()
    print("\r{:^3.0f}%[{}->{}]{:.2f}s".format(c,a,b,-t),\
          end='')
    time.sleep(1)
print("\n"+"执行结束".center(scale//2,'-'))
```

模式匹配与正则表达式



模式匹配

- 模式匹配是数据结构中字符串的一种基本运算，给定一个子串，要求在某个字符串中找出与该子串相同的所有子串，这就是模式匹配
- 假设P是给定的子串，T是待查找的字符串，要求从T中找出与P相同的所有子串。P称为模式，T称为目标。如果T中存在一个或多个模式为P的子串，就给出该子串在T中的位置，称为匹配成功；否则匹配失败

模式匹配

- 查找电话号码
  - ✓ 电话号码：415-555-4242
  - ✓ 模式：3个数字，一个短横线，3个数字，一个短横线，再是4个数字
- 如何检查字符串中是否模式匹配
  - ✓ isPhoneNumber1

File Edit Format Run Options Window Help

```
def isPhoneNumber(text):
    if len(text) != 12:
        return False
    for i in range(0, 3):
        if not text[i].isdecimal():
            return False
        if text[3] != '-':
            return False
        for i in range(4, 7):
            if not text[i].isdecimal():
                return False
            if text[7] != '-':
                return False
            for i in range(8, 12):
                if not text[i].isdecimal():
                    return False
                return True
```

模式匹配

- 从一长串字符串中查找电话号码
  - ✓ isPhoneNumber2

isPhoneNumber2.py - F:\北邮课程\Python程序设计\备课\第五周\isPhoneNumber2.py (3.6.4)

File Edit Format Run Options Window Help

```
def isPhoneNumber(text):
    if len(text) != 12:
        return False
    for i in range(0, 3):
        if not text[i].isdecimal():
            return False
        if text[3] != '-':
            return False
        for i in range(4, 7):
            if not text[i].isdecimal():
                return False
            if text[7] != '-':
                return False
            for i in range(8, 12):
                if not text[i].isdecimal():
                    return False
                return True
message = 'Call me at 415-555-1011 tomorrow. 415-555-9999 is my office.'
for i in range(len(message)):
    chunk = message[i:i+12]
    if isPhoneNumber(chunk):
        print('Phone number found: ' + chunk)
print('Done')
```

正则表达式

- 正则表达式，简称为regex，是文本模式的描述方法
  - ✓ 例：\d是一个正则表达式，表示一位数字字符，即任何一位0到9的数字
  - ✓ \d\d\d-\d\d\d-\d\d\d\d，匹配前面isPhoneNumber()函数匹配的同样文本：3 个数字、一个短横线、3 个数字、一个短横线、4 个数字

正则表达式

- 创建正则表达式对象
  - ✓ Python 中所有正则表达式的函数都在re模块中。在交互式环境中输入以下代码，导入该模块：

```
import re
```

  - ✓ 向 re.compile()传入一个字符串值，表示正则表达式，它将返回一个regex模式对象

```
phoneNumRegex=re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
```

正则表达式

- 匹配regex对象
  - ✓ regex对象的search()方法查找传入的字符串，寻找该正则表达式的所有匹配
  - ✓ 如果字符串中没有找到该正则表达式模式，search()方法将返回None。如果找到了该模式，search()方法将返回一个Match对
  - ✓ match 对象有一个group()方法，它返回被查找字符串中实际匹配的文本
  - ✓ phoneNumRegex1

phoneNumRegex1.py - F:\北邮课程\Python程序设计\备课\第五周\phoneNumRegex1.py (3.6.4)

File Edit Format Run Options Window Help

```
import re
phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')#创建
mo = phoneNumRegex.search('My number is 41-555-4242.')#匹配
print('Phone number found: ' + mo.group())#返回值
```

正则表达式

- 转义字符：倒斜杠（\）。字符串'\n'表示一个换行字符，而不是倒斜杠加上一个小写的n。你需要输入转义字符\\，才能打印出一个倒斜杠
- 在字符串的第一个引号之前加上r，可以将该字符串标记为原始字符串，它不包括转义字符
- 因为正则表达式常常使用倒斜杠，向re.compile()函数传入原始字符串就很方便，而不是输入额外得到斜杠。输入r'\d\d\d-\d\d\d-\d\d\d\d'，比输入'\\d\\d\\d-\\d\\d\\d-\\d\\d\\d\\d'要容易得多

用正则表达式匹配更多模式

- 利用括号分组
  - ✓ 将区号从电话号码中分离，添加括号将在正则表达式中创建“分组”

    (\d\d\d)-(\d\d\d-\d\d\d\d)
  - ✓ 然后可以使用group()匹配对象方法，从一个分组中获取匹配的文本
  - ✓ 第一对括号是第1 组。第二对括号是第2 组。向group()传入整数1或2，可以取得匹配文本的不同部分。向group()方法传入0或不传入参数，将返回整个匹配的文本
  - ✓ #group

用正则表达式匹配更多模式

- 用管道匹配多个分组
  - ✓ 字符|称为“管道”。希望匹配许多表达式中的一个时，就可以使用它。例如，正则表达式 r'Batman|Tina Fey' 将匹配 'Batman' 或 'Tina Fey'
  - ✓ 如果 Batman 和Tina Fey 都出现在被查找的字符串中，第一次出现的匹配文本，将作为Match对象返回
  - ✓ #多个分组

用正则表达式匹配更多模式

- 匹配'Batman'、'Batmobile'、'Batcopter'和'Batbat'中任意一个。因为所有这些字符串都以Bat开始，所以如果能够只指定一次前缀，就很方便。这可以通过括号实现多个分组
  - ✓#多个分组

用正则表达式匹配更多模式

- 用问号实现可选匹配
  - ✓ 不论这段文本在不在，正则表达式都会认为匹配。字符?表明它前面的分组在这个模式中是可选的

用正则表达式匹配更多模式

- 用星号匹配零次或多次
  - ✓*（称为星号）意味着“匹配零次或多次”，即星号之前的分组，可以在文本中出现任意次。它可以完全不存在，或一次又一次地重复
  - ✓+（加号）则意味着“匹配一次或多次”。星号不要求分组出现在匹配的字符串中，但加号不同，加号前面的分组必须“至少出现一次”

用正则表达式匹配更多模式

- 用花括号匹配特定次数
  - ✓ 如果想要一个分组重复特定次数，就在正则表达式中该分组的后面，跟上花括号包围的数字。例如，正则表达式(Ha){3}将匹配字符串'HaHaHa'，但不会匹配'HaHa'，因为后者只重复了(Ha)分组两次
  - ✓ 可以指定一个范围，即在花括号中写下一个最小值、一个逗号和一个最大值。例如，正则表达式(Ha){3,5}将匹配'HaHaHa'、'HaHaHaHa'和'HaHaHaHaHa'

字符分类

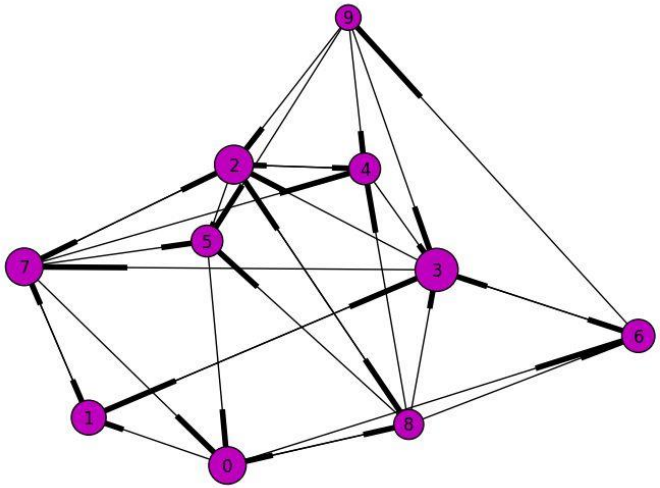
- 缩写字符分类
  - ✓#字符分类

缩写字符分类	表示
\d	0 到 9 的任何数字
\D	除 0 到 9 的数字以外的任何字符
\w	任何字母、数字或下划线字符（可以认为是匹配“单词”字符）
\W	除字母、数字和下划线以外的任何字符
\s	空格、制表符或换行符（可以认为是匹配“空白”字符）
\S	除空格、制表符和换行符以外的任何字符

替换字符串

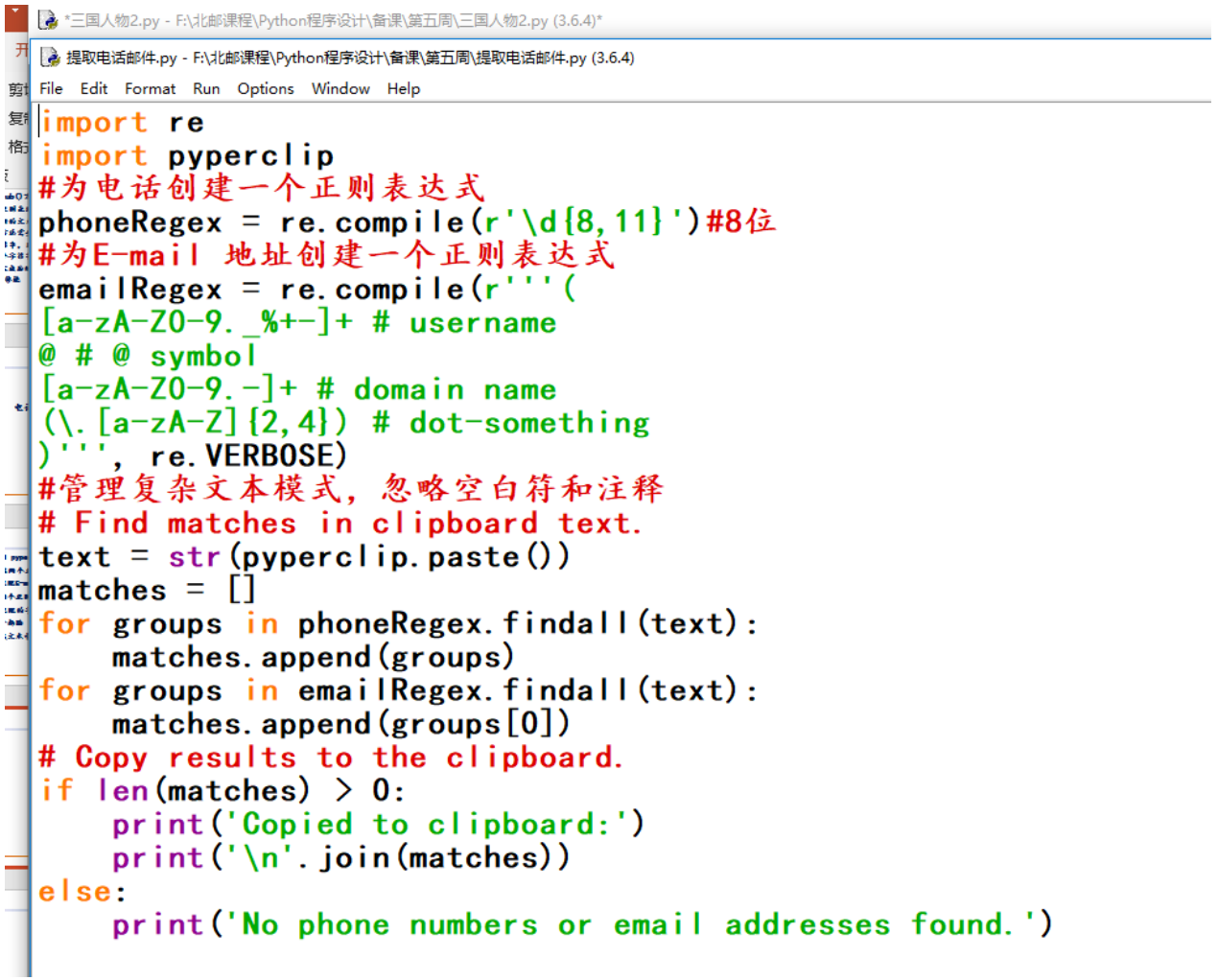
- 用sub()方法替换字符串
  - ✓ 正则表达式不仅能找到文本模式，而且能够用新的文本替换掉这些模式。Regex对象的sub()方法需要传入两个参数。第一个参数是一个字符串，用于取代发现的匹配。第二个参数是一个字符串，即正则表达式。sub()方法返回替换完成后的字符串
  - ✓ #替换

电话号码和E-mail地址提取

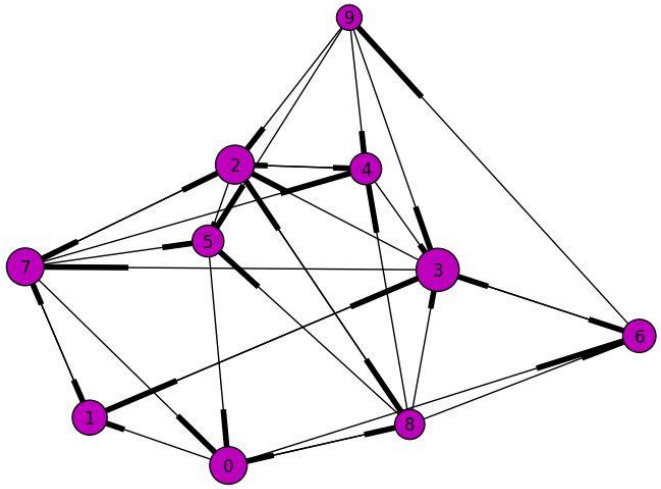


步骤

- 使用 pyperclip 模块复制和粘贴字符串
- 创建两个正则表达式，一个匹配电话号码，另一个匹配E-mail地址
- 对两个正则表达式，找到所有的匹配
- 将匹配的字符串整理好格式，放在一个字符串中，用于粘贴
- 如果文本中没有找到匹配，显示某种消息


三国人物2.py - F:\北邮课程\Python程序设计\备课\第五周\三国人物2.py (3.6.4)
提取电话邮件.py - F:\北邮课程\Python程序设计\备课\第五周\提取电话邮件.py (3.6.4)
File Edit Format Run Options Window Help
import re
import pyperclip
#为电话创建一个正则表达式
phoneRegex = re.compile(r'\d{8,11}')#8位
#为E-mail 地址创建一个正则表达式
emailRegex = re.compile(r'''(
[a-zA-Z0-9._%+-]+ # username
@ # @ symbol
[a-zA-Z0-9.-]+ # domain name
(\.[a-zA-Z]{2,4}) # dot-something
)''', re.VERBOSE)
#管理复杂文本模式，忽略空白符和注释
# Find matches in clipboard text.
text = str(pyperclip.paste())
matches = []
for groups in phoneRegex.findall(text):
matches.append(groups)
for groups in emailRegex.findall(text):
matches.append(groups[0])
# Copy results to the clipboard.
if len(matches) > 0:
print('Copied to clipboard:')
print('\n'.join(matches))
else:
print('No phone numbers or email addresses found.')

从网页中提取图片



步骤

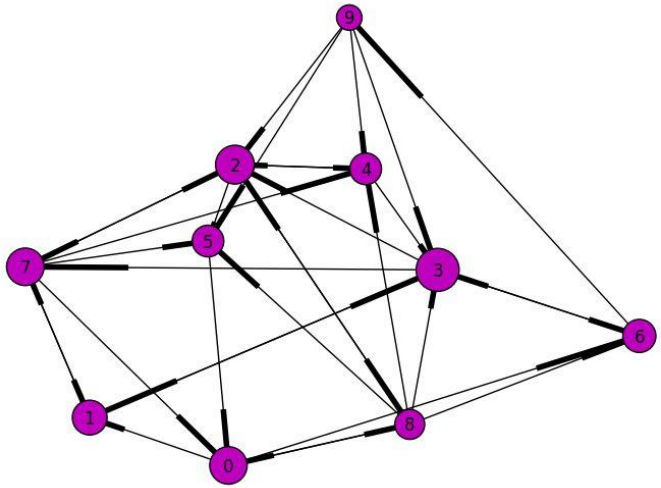
- 1、获取网页信息
- 2、获取图片
- 3、下载图片

图片.py - F:\北邮课程\Python程序设计\备课\第五周\抓取图片\图片.py (3.6.4)

File Edit Format Run Options Window Help

```
#抓取图片
import re
import urllib #urllib库，操作URL的功能
#获取网页信息
url="https://bj.lianjia.com/zufang/"
page = urllib.request.urlopen(url)#抓取网页的源代码
html = page.read()#bytes类型
html=html.decode('utf-8')#str类型
#获取图片，正则表达式
reg=r'data-img="(.*?\.jpg)" alt'#源代码格式图片
imgre=re.compile(reg)
imglist = re.findall(imgre,html)
#抓取页面图片并保存到本地
x=0
for img in imglist:
    urllib.request.urlretrieve(img,'%s.jpg' % x)#直接将远程数据下载到本地
    x+=1
```

课后作业



- 1、从网页（链家租房）获取房源信息
- 2、提取小区名字，房屋面积和价格
- 3、按价格或者面积对房源进行排序

此作业成绩重要构成部分！

谢谢