程序设计

代码复用和模块化设计

主要有如下作用：

代码重用：我们知道当一段代码需要用到两次的时候，我们就需要写一个函数了这是一个道理。

避免变量名的冲突：每个模块都将变量名封装进了自己包含的软件包，这可以避免变量名的冲突。除非使用精确导入。

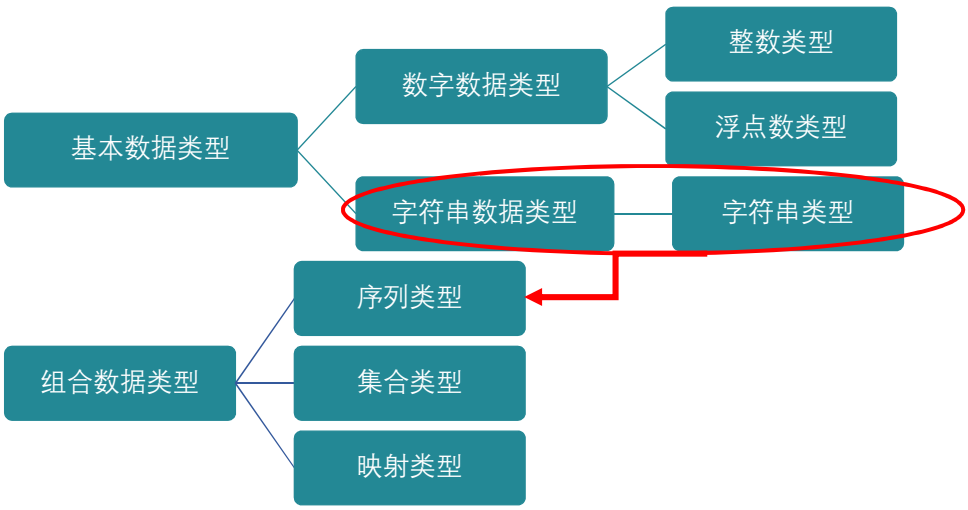
代码复用和模块化设计

import A

from A import B

import A as B

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)。

序列类型

序列类型有12个通用的操作符和函数

操作符	描述
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
s + t	连接s和t
s * n 或 n * s	将序列s复制n次
s[i]	索引，返回序列的第i个元素
s[i: j]	分片，返回包含序列s第i到j个元素的子序列（不包含第j个元素）
s[i: j: k]	步骤分片，返回包含序列s第i到j个元素以j为步数的子序列
len(s)	序列s的元素个数（长度）
min(s)	序列s中的最小元素
max(s)	序列s中的最大元素
s.index(x[, i[, j]])	序列s中从i开始到j位置中第一次出现元素x的位置
s.count(x)	序列s中出现x的总次数

列表的定义

- 列表是值的序列。
- 在字符串中，这些值是字符，在列表中，它可以是任何类型。
- 列表中的值称为元素（element），有时也叫列表项（item）。
- 列表用中括号（[]）表示。

列表的定义

```
[10, 20, 30 ,40]

['一月','二月', '三月']

['Spam', 2.0, 5, [10, 20]]
```

列表的操作——添加

在列表末尾添加，使用.append()方法；

在列表中插入元素，使用insert()方法，需指定新元素的索引和值；

列表的操作——修改

与字符串不同，列表元素的值是可变的：

```
a = ['Spam', 2.0, 5, [10, 20]]
a[1] = 3
```

列表的操作——删除

使用del语句删除元素；

使用pop()方法删除列表末尾的元素，并返回其值；

使用pop()方法删除指定位置的元素；

使用remove()方法，根据值删除元素；

列表的操作——其他方法

使用方法 sort()对列表进行永久性排序；

使用函数 sorted()对列表进行临时排序；

使用reverse()方法倒序打印列表；

使用len()函数，确定列表长度。

列表的遍历

如何进行列表的遍历?

```
a = ['Spam', 2.0, 5, [10, 20]]
for i in a:
    print(id)
```

创建数值列表

如何使用range()函数创建数值列表?

列表切片遍历操作

[start: stop: step]

如果要遍历列表的部分元素，可在for循环中使用切片。

列表的复制

见代码。

元组

元组

列表非常适合用于存储在程序运行期间可能变化的数据集。列表是可以修改的，这对处理网站的用户列表或游戏中的角色列表至关重要。然而，有时候你需要创建一系列不可修改的元素，元组可以满足这种需求。 Python将不能修改的值称为不可变的，而不可变的列表被称为元组。

元组

元组是一组值的序列。其中的值可以是任意类型，使用整数索引其位置，因此元组与列表非常相似。而重要的不同之处在于元组的不可变性。

元组

元组看起来犹如列表，但使用圆括号而不是方括号来标识。定义元组后，就可以使用索引来访问其元素，就像访问列表元素一样。

```
a = tuple()
a = ('student','teacher',12, [12,'a','c'])
a[1]
a[1] = '0'
```

元组

元组看起来犹如列表，但使用圆括号而不是方括号来标识。定义元组后，就可以使用索引来访问其元素，就像访问列表元素一样。

```
a = tuple()
a = ('student','teacher',12, [12,'a','c'])
a[1]
a[1] = '0'
```

元组

如何“修改”元组?

元组的作用

1. 函数多返回值
2. 多变量同步赋值
3. 遍历循环

字典

字典概念

字典类似于列表，但更加通用；

在列表中，下标是整数，在字典中，下标可以是任意类型；

字典包含下标（键）集合和值集合；

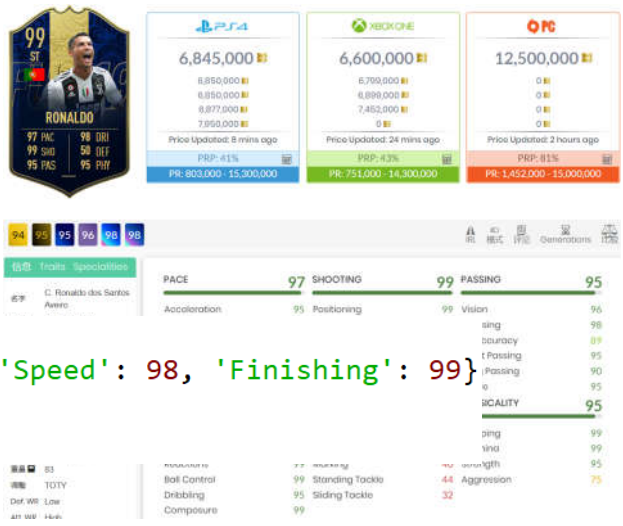
键与值之间的关联被称为键值对；

字典体现的是键值对的映射；

字典可以高效模拟现实世界的情景。

字典案例

游戏中，C罗的各个属性值是不同的，可以用字典存储部分属性的值：




```
# 字典案例
CR_7 = {'Club': 'Juventus', 'Speed': 98, 'Finishing': 99}
print(CR_7['Club'])
print(CR_7['Speed'])
```

字典的创建

```
# 字典案例
CR_7 = {'Club': 'Juventus', 'Speed': 98, 'Finishing': 99}
print(CR_7['Club'])
print(CR_7['Speed'])
```

用花括号{}中的键-值对表示；

键值之间用冒号分隔，键-值对之间用逗号分隔；

字典值的访问

```
# 字典案例
CR_7 = {'Club': 'Juventus', 'Speed': 98, 'Finishing': 99}
print(CR_7['Club'])
print(CR_7['Speed'])
```

要获取与键相关联的值，可依次指定字典名和放在方括号内的键

字典键-值对的添加、修改、删除

字典是一种动态结构，可随时在其中添加键—值对；

要添加键—值对，可依次指定字典名、用方括号括起的键和相关联的值；

```
# 字典案例
CR_7 = {'Club': '尤文图斯', 'Speed': 98, 'Finishing': 99}
print(CR_7['Club'])
print(CR_7['Speed'])

# 字典的添加（添加键值对）
CR_7['Heading'] = 95
CR_7['Nationality'] = '葡萄牙'
```

字典键-值对的添加、修改、删除

创建空字典：使用字典来存储用户提供的数据或在编写能自动生成大量键—值对的代码时，通常都需要先定义一个空字典。

```
# 创建空字典
Messi = {}
```

字典键-值对的添加、修改、删除

修改字典中的值，可依次指定字典名、用方括号括起的键以及与该键相关联的新值。

```
# 修改字典值
CR_7['Speed'] = 90
```

字典键-值对的添加、修改、删除

删除键值对，可使用del语句将相应的键—值对彻底删除；

使用del语句时，必须指定字典名和要删除的键；

字典遍历

遍历字典有多种方式：可遍历字典的所有键—值对、键或值：

1. 要编写用于遍历字典的for循环，可声明两个变量，用于存储键—值对中的键和值。对于这两个变量，可使用任何名称。使用items()方法访问键值对
2. 在不需要使用字典中的值时，使用方法keys()遍历键。
3. 遍历值可使用方法values()，返回一个值列表，而不包含任何键。

字典遍历

```
for key,value in CR_7.items():
    print(key)
    print(value)
```

```
for value in CR_7:
    print(value)
```

```
for value in CR_7.values():
    print(value)
```

```
for key in CR_7.keys():
    print(key)
```

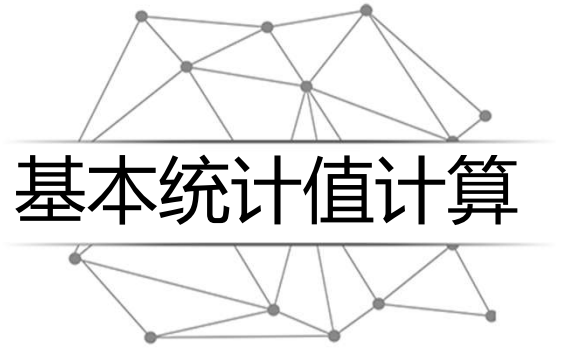
字典嵌套

将一系列字典存储在列表中，或将列表作为值存储在字典中，这称为嵌套。

1. 在列表中嵌套字典
2. 在字典中嵌套列表
3. 在字典中嵌套字典

字典操作总结

函数和方法	描述
<d>.keys()	返回所有的键信息
<d>.values()	返回所有的值信息
<d>.items()	返回所有的键值对
<d>.get(<key>,<default>)	键存在则返回相应值，否则返回默认值
<d>.pop(<key>,<default>)	键存在则返回相应值，同时删除键值对，否则返回默认值
<d>.popitem()	随机从字典中取出一个键值对，以元组(key, value)形式返回
<d>.clear()	删除所有的键值对
del <d>[<key>]	删除字典中某一个键值对
<key> in <d>	如果键在字典中返回True，否则返回False



基本统计值计算

基本统计值的计算

以最简单的统计问题为例，求解一组不定长数据的基本统计值，即平均值、标准差、中位数。

一组数据表示为$S=s_0, s_1, \ldots, s_{n-1}$，其算术平均值、标准差分别表示为：

$$m=\left(\sum_{i=0}^{n-1} s_i\right)/n \quad \text{和} \quad d=\sqrt{\left(\sum_{i=0}^{n-1}(s_i-m)^2\right)/(n-1)}$$

基本统计值的计算

由于平均数、标准差和中位数是三个不同的计算目标，使用函数方式编写计算程序。

getNum()函数从用户输入获得数据

mean()函数计算平均值

dev()函数计算标准差

median()函数计算中位数

基本统计值的计算

实例代码9.1 e9.1CalStatistics.py

```
#e9.1CalStatistics.py
from math import sqrt
def getNum():        #获取用户输入
    nums = []
    iNumStr = input("请输入数字(直接输入回车退出): ")
    while iNumStr != "":
        nums.append(eval(iNumStr))
        iNumStr = input("请输入数字(直接输入回车退出): ")
    return nums
def mean(numbers):   #计算平均值
    s = 0.0
    for num in numbers:
        s = s + num
    return s / len(numbers)
```

基本统计值的计算

实例代码9.1　　e9.1CalStatistics.py

```
15 def dev(numbers, mean): #计算方差
16     sdev = 0.0
17     for num in numbers:
18         sdev = sdev + (num - mean)**2
19     return sqrt(sdev / (len(numbers)-1))
20 def median(numbers):    #计算中位数
21     sorted(numbers)
22     size = len(numbers)
23     if size % 2 == 0:
24         med = (numbers[size//2-1] + numbers[size//2])/2
25     else:
26         med = numbers[size//2]
27     return med
28 n =  getNum()  #主体函数
29 m =  mean(n)
30 print("平均值:{},方差:{:.2},中位数:{}.".format(m,\
    dev(n,m),median(n)))
```

基本统计值的计算

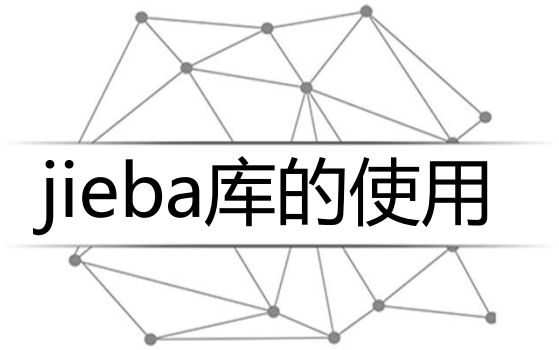
```
>>>
请输入数字(直接输入回车退出): 99
请输入数字(直接输入回车退出): 98
请输入数字(直接输入回车退出): 97
请输入数字(直接输入回车退出): 96
请输入数字(直接输入回车退出): 95
请输入数字(直接输入回车退出):
平均值:97.0,方差:1.6,中位数:97.
```

程序先后调用getNum()、mean()、dev()和median()函数。利用函数的模块化设计能够复用代码并增加代码的可读性。每个函数内部都采用了简单的语句。

基本统计值的计算

列表在实现基本数据统计时发挥了重要作用，表现在：

- 列表是一个动态长度的数据结构，可以根据需求增加或减少元素；
- 列表的一系列方法或操作符为计算提供了简单的元素运算手段；
- 列表提供了对每个元素的简单访问方式及所有元素的遍历方式。



jieba库的使用

jieba库的概述

jieba是Python中一个重要的第三方中文分词函数库

```
>>>import jieba
>>>jieba.lcut("中国是一个伟大的国家")
['中国', '是', '一个', '伟大', '的', '国家']
```

jieba库是第三方库，不是安装包自带，需要通过pip指令安装

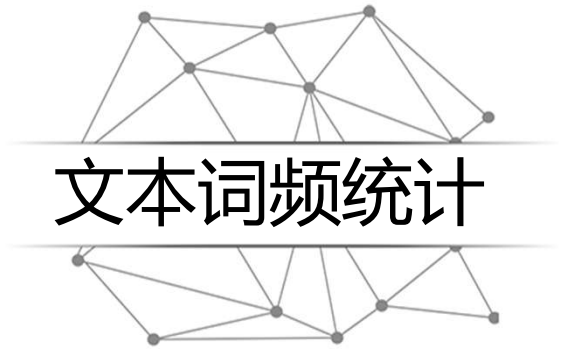
```
:\>pip install jieba    # 或者 pip3 install jieba
```

jieba库的解析

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库的解析

```
>>>import jieba
>>>jieba.lcut("中华人民共和国是一个伟大的国家")
['中华人民共和国', '是', '一个', '伟大', '的', '国家']
>>>jieba.lcut("中华人民共和国是一个伟大的国家", cut_all=True)
['中华', '中华人民', '中华人民共和国', '华人', '人民', '人民共和国', '共和', '共和国', '国是', '一个', '伟大', '的', '国家']
>>>jieba.lcut_for_search("中华人民共和国是一个伟大的国家")
['中华', '华人', '人民', '共和', '共和国', '中华人民共和国', '是', '一个', '伟大', '的', '国家']
```



文本词频统计

《Hamlet》英文词频统计

实例代码10.1　　e10.1CalHamlet.py

```
#e10.1CalHamlet.py
def getText():
txt = open("hamlet.txt", "r").read()
txt = txt.lower()
for ch in '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~':
    txt = txt.replace(ch, " ")    #将文本中特殊字符替换为空格
return txt
hamletTxt = getText()
words  = hamletTxt.split()
counts = {}
for word in words:
    counts[word] = counts.get(word,0) + 1
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(10):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

```
for word in words:
    counts[word] = counts.get(word,0) + 1

items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
```

《Hamlet》英文词频统计

```
>>>
the         1138
and          965
to           754
of           669
you          550
a            542
i            542
my           514
hamlet      462
in           436
```

观察输出结果可以看到，高频单词大多数是冠词、代词、连接词等语法型词汇，并不能代表文章的含义。进一步，可以采用集合类型构建一个排除词汇库excludes，在输出结果中排除这个词汇库中内容。

实例代码10.2　　e10.2CalHamlet.py

```
#e10.2CalHamlet.py
excludes = {"the","and","of","you","a","i","my","in"}
def getText():
txt = open("hamlet.txt", "r").read()
txt = txt.lower()
for ch in '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~':
    txt = txt.replace(ch, " ")    #将文本中特殊字符替换为空格
return txt
hamletTxt = getText()
words  = hamletTxt.split()
counts = {}
for word in words:
counts[word] = counts.get(word,0) + 1
for word in excludes:
del(counts[word])
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(10):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

《Hamlet》英文词频统计

运行程序后，输出结果如下

```
>>>
to          754
hamlet      462
it          416
that        391
is          340
not         314
lord        309
his         296
this        295
but         269
```

《三国演义》人物出场统计

实例代码10.3　　e10.3CalThreeKingdoms.py

```
#e10.3CalThreeKingdoms.py
import jieba
txt = open("三国演义.txt", "r", encoding='utf-8').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 x:x[1], reverse=True)
for i in range(15):
    word, count = items[i]
print ("{0:<10}{1:>5}".format(word, count))
```

《三国演义》人物出场统计

```
>>>
曹操          953
孔明          836
将军          772
却说          656
玄德          585
关公          510
丞相          491
二人          469
不可          440
荆州          425
玄德曰         390
孔明曰         390
不能          384
如此          378
张飞          358
```

《三国演义》人物出场统计

观察输出结果，同一个人物会有不同的名字，这种情况需要整合处理。同时，与英文词频统计类似，需要排除一些人名无关词汇，如“却说”、“将军”等。

《三国演义》人物出场统计

实例代码10.4 e10.4CalThreeKingdoms.py

```
#e10.4CalThreeKingdoms.py
import jieba
excludes = {"将军","却说","荆州","二人","不可","不能","如此"}
txt = open("三国演义.txt", "r", encoding='utf-8').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"
```

《三国演义》人物出场统计

实例代码10.4 e10.4CalThreeKingdoms.py

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

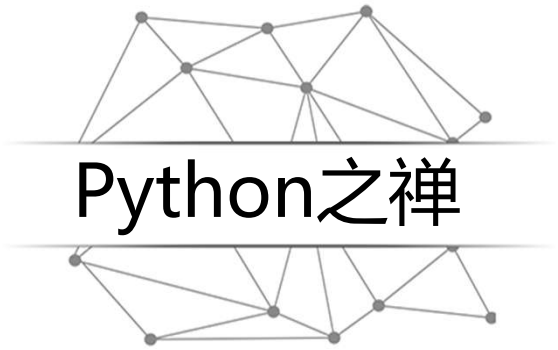
《三国演义》人物出场统计

输出排序前5的单词，运行程序后，输出结果如下：

```
>>>
曹操          1451
孔明          1383
刘备          1252
关羽           784
张飞           358
```

请继续完善程序，排除更多无关词汇干扰，总结出场最多的20个人物都有哪些。这里，给出参考答案。

曹操（1451）、孔明（1383）、刘备（1252）、关羽（784）、张飞（358）、吕布（300）、赵云（278）、孙权（264）、司马懿（221）、周瑜（217）、袁绍（191）、马超（185）、魏延（180）、黄忠（168）、姜维（151）、马岱（127）、庞德（122）、孟获（122）、刘表（120）、夏侯惇（116）



Python之禅

什么样的程序是好的？如何编写漂亮的代码？这都是学习编程一段时间最经常提出的问题，却最难回答。程序设计语言如同自然语言，好的代码就像文学作品，不仅达意，更要优美。那什么是“好”？什么是“优美”？领悟编程代码优美的过程类似参禅，除了不断练习，也需要理解一些原则。

Python之禅

Python编译器以函数库的形式内置了一个有趣的文件，被称为“Python之禅”（The Zen of Python）。当调用如下一行语句后，会出现一段有趣的运行结果。

```
>>>import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Python之禅　　作者：Tim Peters	译者心得
•优美胜于丑陋	以编写优美代码为目标，不多解释
•明了胜于隐晦	优美代码应该清晰明了，规范统一
•简洁胜于复杂	优美代码应该逻辑简洁，避免复杂逻辑
•复杂胜于凌乱	如果必须采用复杂逻辑，接口关系也要清晰
•扁平胜于嵌套	优美代码应该是扁平的，避免太多层次嵌套
•间隔胜于紧凑	优美代码间隔要适当，每行代码解决适度问题
•可读性很重要	优美代码必须是可读且易读的
•即便假借特例的实用性之名，也不要违背上述规则	上述规则是至高无上的
•除非你确定需要，任何错误都应该有应对	捕获异常，不让程序留有因错误退出的可能
•当存在多种可能，不要尝试去猜测	
•只要你不是Guido，对于问题尽量找一种，最好是唯一明显的解决方案	不要试图给出多种方案，找到一种实现它，几乎所有人都没有Guido那么牛
•做也许好过不做，但不假思索就动手还不如不做	编程之前要有思考
•如果你无法向人描述你的实现方案，那肯定不是一个好方案	能说清楚的往往才是对的
•如果实现方案容易解释，可能是个好方案	
•命名空间是绝妙的理念，要多运用	适合复杂程序编程

Python之禅

除了Python之禅所表达的Python设计理念，该程序还有另一段魅力：

实例代码11.1 this.py

```
s = """Gur Mra bs Clguba, ol Gvz Crgref

Ornhgvshy vf orggre guna htyl.
Rkcyvpvg vf orggre guna vzcyvpvg.
Fvzcyr vf orggre guna pbzcyrk.
Pbzcyrk vf orggre guna pbzcyvpngrq.
Syng vf orggre guna arfgrq.
Fcnefr vf orggre guna qrafr.
Ernqnovyvgl pbhagf.
Fcrpvny pnfrf nera'g fcrpvny rabhtu gb oernx gur ehyrf.
Nygubhtu cenpgvpnyvgl orngf chevgl.
Reebef fubhyq arire cnff fvyragyl.
Hayrff rkcyvpvgyl fvyraprq.
```

Python之禅

实例代码11.1 this.py

```
Va gur snpr bs nzovthvgl, ershfr gur grzcgngvba gb thrff.
Gurer fubhyq or bar-- naq cersrenoyl bayl bar --boivbhf jnl gb qb vg.
Nygubhtu gung jnl znl abg or boivbhf ng svefg hayrff lbh'er Qhgpu.
Abj vf orggre guna arire.
Nygubhtu arire vf bsgra orggre guna *evtug* abj.
Vs gur vzcyrzragngvba vf uneq gb rkcynva, vg'f n onq vqrn.
Vs gur vzcyrzragngvba vf rnfl gb rkcynva, vg znl or n tbbq vqrn.
Anzrfcnprf ner bar ubaxvat terng vqrn -- yrg'f qb zber bs gubfr!"""

d = {}
for c in (65, 97):
    for i in range(26):
        d[chr(i+c)] = chr((i+13) % 26 + c)

print("".join([d.get(c, c) for c in s]))
```

Python之禅

密文：A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
原文：N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
密文：a b c d e f g h i j k l m n o p q r s t u v w x y z
原文：n o p q r s t u v w x y z a b c d e f g h i j k l m

这个算法可以看作是凯撒密码的一种扩展，相比凯撒密码，采用循环移动13个位置，加密和解密可以用同一个程序。