python入门教程:一篇不错的Python入门教程

疯狂代码 http://CrazyCoder.cn/ ĵ:http:/CrazyCoder.cn/Python/Article69262.html
原文 http://www.hetland.org/python/instant-hacking.php
Instant Hacking[译文]
译者: 肯定来过
这是篇简短有关python设计语言入门教程原文在这里翻着词典翻译了来！
这是份对编程艺术简短介绍其中例子是用python写成(如果你已经知道了该如何编程但是想简单了解下python你可以查阅我另篇文章Instant Python)这篇文章已经被翻译为意大利、波兰、日本、塞尔维亚以及巴西葡萄亚语等许多种语言而且正在被翻译为韩语(译者:当然现在已经包括了中文版本只是作者并不知道)
这篇文章和如何闯入别人计算机系统的类东西无关我不关注那类事情所以请不要email问我那些东西
注意:要使此文中例子正确运行你应该把它们写在个文本文件中然后用解释器运行；不要试图直接在交互方式下运行它们 - - 不是所有都可以这样运行(不要问我和此有关具体细节最好查阅python文档或者email给help@python.org)
1. 运行环境
要用python写你必须先安装个python解释器它可以存在于大多数平台(包括Macosh、Unix和Windows)更多和此有关信息可以在python网站WebSite上找到你还应该有个文本编辑器(象emacs、notepad或者类似东西)

2. 编程是什么？
为计算机写其实就是给它系列指令告诉它去做什么计算机在某些方面就象是菜谱指导我们如何做菜那种例如[1]:
假日火腿沙拉
原料:
腌泡汁:
1/4杯酸橙汁
1/4杯低钠大豆酱油
1/4杯水
1大汤匙植物油
3/4茶匙小茴香
1/2茶匙牛至
1/4茶匙热胡椒粉
2片丁香、大蒜捣碎
沙拉:
1份(12盎司)罐装少钠午餐肉火腿切成条状
1个洋葱切片
胡椒粉切好生菜

12个樱桃西红柿切半
思路方法:
把腌泡汁装在有合适盖子广口瓶里摇匀用塑料袋装上火腿泼上腌泡汁封住袋口在电冰箱里腌制30分钟从塑料袋里取出火腿；准备2大汤匙腌泡汁在煮锅里煮下加上火腿、洋葱、绿色胡椒烧3到4分钟直到火腿熟了为止……
当然没有台计算机会懂这个……而且即便是懂大多数计算机也不可能烧制出份沙拉那么我们该如何让这些变得对计算机来说更为友好些呢？从根本上说依赖于两点:首先我们必须以计算机可以理解方式和的交流；其次还要和它谈论它能够做到事情
第点意味着我们必须使用种语言 - - 种已经为的准备好了解释器设计语言第 2点意味着我们不能期望计算机为我们做份沙拉 - - 但是我们可以让它做数字累加或者在屏幕上打印东西的类事情
3. Hello……
设计教程有个传统通常以在屏幕上打印“Hello, world!”这样做为开始对python来说这非常简单:
pr "Hello, world!"

它从根本上说很象上面菜谱(尽管要短得多！)它告诉计算机做什么:打印“Hello, world!”如果让它打印更多废话该如何做呢？很简单:
pr "Hello, world!"
pr "Goodbye, world!"

不比上个难是不是？但是不如何有趣……我们希望它可以处理更多元素就象沙拉菜谱那样那么我们都有哪些元素呢？首先有串象“Hello, world!”除此的外还有数字假设我们打算让计算机为我们计算矩形面积我们可以给它如下菜谱:
# The Area of a Rectangle
# Ingredients:
width = 20
height = 30
# Instructions:
area = width * height
pr area

你大概可以看出它同火腿沙拉菜谱相似性(尽管有些细微差别)但它是如何工作呢？首先以#开始行叫做注释事实上会被计算机忽略然而插入象这样小段注释对于增强你可读性来说是很重要
接下来看起来象 foo = bar 这样行叫做赋值对于 width = 20 这样情况来说就是告诉计算机从这里开始width就代表20了它还意味着个名字为“width”变量从此被创建了(如果它先前已经存在那么会被重新覆盖)所以我们以

后使用这个变量时候计算机就知道了它值因此
width * height
本质上同
20 * 30
样会计算出600这个结果然后赋给名称为“area”变量最后句在屏幕上打印出变量“area”值所以你看到这个运行最终结果仅仅是
600
注意:在某些设计语言中你必须在开始时候告诉计算机你将会用到哪些变量(就象沙拉中元素) - - 而python足够聪明所以你可以根据需要随时创建
4. 反馈
现在你可以执行些简单或者再复杂点计算了比方说你或许打算写段来计算圆形面积而不是矩形:

```
radius = 30
pr radius * radius * 3.14
```

然而这事实上并不比计算矩形面积那个更有意思至少在我看来是这样它有些僵硬如果我们看到半径为31圆该如何办？怎样让计算机知道？这有点象沙拉菜谱中:“烧3到4分钟直到火腿熟了为止”要知道何时烧熟我们必须检查我们需要反馈或者提示计算机如何知道我们圆形半径？同样需要输入资料……我们可以做是告诉计算机半径是多少:

```
radius = input("What is the radius?")
pr radius * radius * 3.14
```

现在变得漂亮些了……input是个被称为东西(很快你将学习创建你自己而input是python内建)仅仅写下
input
什么也不会做……你必须在它后面放上对括号所以input可以工作 - - 它会简单要求用户输入半径长度而上面那个版本对用户来说也许更友好些它首先打印出了个问题当我们将诸如提问串“What is the radius?”的类东西放在括号中时这个过程被称为参数传递括号中内容被称为参数在上个例子中我们传递了个提问作为参数以便input知道在获得答案前应该先打印什么
但是获得答案如何到达radius变量呢？input时会返回个值(象许多其它样)你不定非要使用这个值但象我们这种情况我们要使用它这样下面这两个表达式有着很大差别:

```
foo = input
bar = input
```

foo现在包含input本身(所以它事实上可以象foo("What is your age?")这样使用；这被称为动态)而bar包含用户键入值

5. 流程

现在我们可以编写执行简单任务(运算和打印)并且可以获得用户输入了这很有用但仍然局限在按顺序执行命令也就是说－－它们必须按照事先安排好顺序执行大多数火腿沙拉菜谱是象这样顺序或者线性叙述但是如果我们打算让计算机检查沙拉是否烧好该怎样告诉它呢？如果烧好了那么应该从烘箱里把它取出来－－否则话应该接着让它烧更长段时间什么我们如何表达这个？
我们想做其实是控制流程它可以从两个方向执行－－要么拿开火腿要不继续让它留在烘箱里我们可以选择条件是它是否烧好这被称为条件执行我们可以这样写:

```
temperature = input("What is the temperature of the spam?")
 temperature >; 50:
     pr "The salad is properly cooked."
:
     pr "Cook the salad some more."
```

意思很明显:如果温度超过50(摄氏度)那么打印出信息告诉用户烧好了否则告诉用户再烧制段时间
注意:缩进在python中很重要条件执行(还有循环执行以及定义－－见后面)中语句块必须被缩进(而且要缩进同等数量空格；个键相当于8个空格)以便解释器可以知道它们从哪里开始到哪里结束这同时也使变得更加可读
让我们回到先前面积计算问题能看出来这段做什么吗？

```
# Area calculation program
pr "Welcome to the Area calculation program"
pr "---------------------------------------"
pr
# Pr out the menu:
pr "Please select a shape:"
```

```
pr "1 Rectangle"
pr "2 Circle"
#Get the user's choice:
shape = input(">; ")
#Calculate the area:
 shape  1:
     height = input("Please enter the height: ")
     width = input("Please enter the width: ")
     area = height *width
     pr "The area is ", area
:
     radius = input("Please enter the radius: ")
     area = 3.14 * (radius**2)
     pr "The area is ", area
```

这个例子中新东西:
1. 只使用pr本身将打印出个空行
2. 检查两个值是否相等和=区别后者把表达式右侧值赋给左侧变量这是个非常重要差别！
3. **是python幂运算符 - - 因此半径平方被写成radius**2
4. pr能够打印出不止个东西只要用逗号把它们分开就可以了(它们在输出时会用单个空格分开)
这个很简单:它要个数字告诉它用户打算让它计算矩形或是圆形面积然后使用个语句(条件执行)来决定应当执行哪个语句块计算面积这两个语句块同先前面积计算例子中使用语句块本质上是样留意注释是如何使代码变得更加可读编程第条戒律就是:“你应当注释！”无论如何 - - 它都是个应该养成好习惯

练习1:
扩展上面使它包括正方形面积计算用户只要输入它条边长度就可以了做这个练习的前你需要了解件事:如果你有两个以上选择你可以象这样写:

```
 foo  1:
     # Do something...
el foo  2:
     # Do something ...
el foo  3:
     # If all  fails...
```

这里el是意思为 “ ” 神秘代码:)所以如foo等于1做某件事；否则如果foo等于2那么做另外些事等等你也可以在中加入其它选项 - - 象 3角形以及任意多边形随你便
6. 循环
顺序执行和条件执行仅仅是设计 3个基本语句块架构方式中两个第 3个则是循环执行在上个段落中我假设了种情况检查火腿是否烧好但很明显它并不适用如果下次检查时火腿仍然没烧好该如何办？我们如何知道需要检查多少次？事实上我们不知道而且我们也没必要知道我们可以要求计算机持续检查直到烧好了为止如何表达这个？你猜到了 - - 我们使用循环或者说是重复执行
python有两种循环类型:while循环和for循环for循环大概是最简单举个例子:

```
for food in "spam", "eggs", "tomatoes":
    pr "I love", food
```

它意思是:对于列表"spam", "eggs", "tomatoes"中每个元素都打印出你喜欢它循环中语句块为每个元素执行次而且每次执行当前元素都被赋给变量food(在这个例子中)另外个例子:

```
for number in range(1, 100):
    pr "Hello, world!"
    pr "Just", 100 - number, "more to go..."
pr "Hello, world"
pr "That was the last _disibledevent=>temperature = input("How hot is the spam?")
while temperature < hot_enouth:
    pr "Not hot enough... Cook it a bit more..."
    sleep(30)
    temperature = input("OK, How hot is it now?")
pr "It's hot enough - You're done!"
```

这个例子中新东西……
1. 有些有用被存储在模块中而且可以被导入此例中我们从python自带time模块中导入了sleep(它休止给定多少秒时间)(做你自己模块当然也是可能……)

练习2:
写个持续从用户获得数据然后相加直到它们和为100再写个从用户那里获得100个数据打印出它们和
Bigger Programs - Abstraction
如果想知道本书大致内容你不会翻遍所有页 - - 你只是看看目录是不是？它会列出书主要内容现在 - - 想像写本菜谱许多菜谱像“奶油火腿通心面”和“瑞士火腿馅饼”很可能包含相同东西比如火腿在这种情况下 - - 你肯定不会打算在每个菜谱里都重复叙述如何制作火腿(好了……你事实上可能不做火腿……但是为了做例子请忍受下:))你会把制作火腿菜谱单独放在个章节而仅仅在其它章节里引用它这样 - - 代替在每个菜谱里都完整描述你只要引用章节名称就可以了在计算机编程中这被称为抽象化
我们是不是已经象这样运行了某些东西？是我们没有详细告诉计算机如何从用户那里获得个答案(好了 - - 我们没有真这样做……同样地……我们也没有真正在做火腿:))而是简单使用了input - - 个来代替我们事实上可以构造我们自己来应用于这种类型抽象化中
假设我们希望找到小于给定正数最大整数例如给定2.7这个数应当是2这往往被称为给定数“底线(floor)”(这事实上可以用python内建来处理但是请再次忍受我拿它作例子……)我们该怎样做？个简单解决办法是从0开始试每个可能数:

```
number = input("What is the number?")
floor = 0
while floor <= number:
    floor = floor + 1
floor = floor - 1
pr "The floor of ", number, "is ", floor
```

注意当floor不再小于(或者等于)给定数时循环结束了；我们加了太多1给它因此我们必须为它减去1如果我们希望把它应用于完整数学运算该如何办呢？我们不得不为求每个数基数("floor"-ing)而写次完整循环这很不舒服……你可能猜到了我们代的以什么:把它放在我们自己中命名为“floor”：

```
def floor(number):
    result = 0
    while result <= number:
        result = result + 1
    result = result - 1
     result
```

这个例子中新东西……

1. 用关键字def定义名紧随其后并且要用括号把需要参数括起来
2. 如果要求返回个值要使用关键字来处理(它同时也自动结束定义)
定义了的后我们可以象这样使用它:

```
x = 2.7
y = floor(2.7)
```

执行后y值应该是2定义拥有多个参数也是可以:

```
def sum(x, y):
    x + y
```

练习3
写个用欧几里德思路方法寻找两个数个共同因数工作过程是这样:
1. 假设两个数a和ba大于b
2. 重复以下步骤直到b变成0:
1. a变为b值
2. b变成没有改变值的前a除以没有改变值的前b余数
3. 返回a最后个值
提示:
* 使用a和b作为参数
* 简单设定a大于b
* x除以z余数用表达式 x % z 来计算
* 两个变量可以象这样起赋值:x, y = y, y+1这里x被赋以值y(这意味着y值此前已经指定)而且y被递增了1
7. 深入
上面练习如何做？难吗？还不太清楚？别担心－－我还没完成我话题呢
我们构建时使用萃取思路方法称为过程抽象许多编程语言把关键字过程同样使用事实上这两个概念是不样但是在python中它们都被称为(它们或多或少以同样方式定义和使用)
和过程(在其它语言中)区别在哪里呢？嗯－－就像你在前面段落里看到那样可以返回个值区别就是过程并不返回这样值许多时候用这种思路方法把划分为两种类型－－返回值和不返回值－－是很有用
不返回值(过程)可以用作子或例行我们这些它们制造某些原料就象泡沫鲜奶的类我们可以在很多地方使用这个而不需要重写它代码(这被称为代码再利用－－以后你还会知道它意义不仅仅在这里)
这样(或过程)另个有用性体现在－－它改变了环境(例如把糖和奶油混在起搅拌它们整个外部状态就变化了)让我

们看个例子:

```
def hello(who):
    pr "Hello, ", who
hello("world")
# Prs out "Hello, world"
```

打印出内容是它方面作用这是这个唯需要做事它其实是个典型所谓过程但是……它事实上没有改变它运行环境是不是？它怎样才能改变呢？让我们试下:

```
# The *wrong* way of doing it
age = 0
def Age(a):
    age = a
Age(100)
pr age
# Prs "0"
```

错在哪儿？错在Age创建了它自己也被命名为age局部变量它只在Age内部可用那如何才可以避免出现这个问题呢？我们可以使用全局变量
注意:全局变量在python中不常用它们容易引起不好代码组织结构被称为意大利面代码我这里使用它们是为了引出更复杂点技术问题 - - 如果你可以请尽量避免使用它们

[color=#FF0000]未译完[/color]
rockety 回复于:2005-06-06 09:27:59
[color=red]译完了只是不小心没在我blog上贴全而且也没有给出缩进:oops: 今天并更正了感谢wolfg转贴并给出了正确缩进以下是其余部分:[/color]
通过告诉解释器个变量是全局(用象global age这样表达式做)我们事实上
告诉了它在的外使用这个变量而不是重新创建个新局部变量(所以和局部
相反它是全局)因此上面可以象这样重写:

```
# The correct, but not-so-good way of doing it
```

```
age=0
def Age(a):
    global age
Age(100)
pr age
# Prs "100"
```

了解对象(随后谈到)后你会发现更好解决这个问题办法是使用个有age属
性和Age思路方法对象在数据结构那段你也将会发现些改变它环境更好
例子
好了 - - 那么真正是什么样？什么是呢事实上？数学象种“机
器”获得输入然后计算结果它会每次返回同样结果如果每次提供它同样输入
例如:

```
def square(x):
 x*x
```

这和数学上f(x)=x*x 样它行为象个精确仅仅依赖于它输
入在任何情况下都不改变它环境
所以 - - 我这里描绘了两种构造思路方法:种类型更象是过程不返回任何结
果；另种更象是数学上(几乎)什么也不做就是为了返回个结果当然在
这两种极端事物的间做某些事情是可能尽管当改变事物时候它应该清楚它改
变了你可以通过标记它们名字区分它们例如为“纯粹”使用象square这样
名词而对类似过程那样使用象Age这样命令式名字
9. 更多类型 - 数据结构
现在 - - 你已经知道了不少:怎样输入输出怎样设计复杂运算法则()来执
行数学运算但是好戏还在后头呢
截止目前我们都在中使用了哪些成份呢？数字和串对不对？没意思种
类……现在让我们引入两 3个其它成份来让事情变得更有意思些
数据结构是种组织数据成份(惊奇吃惊……)单个数据没有什么真正数据
结构是不是？但是假设我们需要很多数放在起做为个成份 - - 那就需要某种结构
例如我们可能想要个数据列表那很容易:

[3, 6, 78, 93]
在循环那段我提到了列表但没真正描述它好 - - 这里说就是你如何创建它只
需要列出元素用逗号分开再加上方括号就行了
来看个计算素数(只能被1和它本身整除数)例子:

```
# Calculate all the primes below 1000
# (Not the best way to do it, but...)
result = [1]
candidates = range(3, 1000)
base = 2
product = base
while candidates:
    while product < 1000:
        product in candidates:
            candidates.remove(product)
        product = product+base
    result.append(base)
    base = candidates[0]
    product = base
    del candidates[0]
    result.append(base)
    pr result
```

这个例子中新东西……
内建range事实上返回个列表可以象所有其它列表那样使用(它包括第
个数但是不包括最后个数)
列表可以当作逻辑变量使用如果它非空则为true否则为false因此while
candidates意思是“while名称为candidates列表非空时”或者简单说“while存
在candidates时”
你可以用 someElement in somelist来检查个元素是否在列表中
你可以用someList.remove(someElement)来删除someList中someElement
你可以用someList.append(something)为个列表添加元素事实上你也可以使
用“+”(象someList = someList+[something])但是效率不是太高
你可以通过在列表名的后加上用括号括起来表示某元素位置数字(很奇怪列

表第1个元素位置是0)来获得列表某个元素因此someList[3]是someList
列表第 4个元素(依次类推)
你可以使用关键字del删除变量它也可以用来删除列表中元素(就象这里)
因此del someList[0]删除someList 列表中第个元素如果删除前列表是[1, 2,
3]删除后就变成了[2, 3]
在继续叙述索引列表中元素的前我简单解释下上面例子
这是古老算术个版本称为“The Sieve of Erastothenes”(类似这样)它考量
系列给定数字(在本例中是个列表)然后有组织删除已知不是素数数字如何知
道？只要看看它们是不是可以被分解为其它两个数就可以了
我们从个包含数字[2...999]候选列表开始 - - 我们知道1是素数(事实上它可能
是也可能不是看你问谁了)我们想得到小于1000所有素数(事实上我们候
选列表是[3...999]但是2也是候选数字它是我们第个base)我们还有个叫result列表它任何时间都包含着最新结果
最初时候它只包含1我们还有个叫base变量每次循环我们删除是它倍数数字(它总是候选列表中最小数)每次循环
的后我们知道剩下最小数是素数(所有可以分解数我们都删除了)
因此我们把它加入result并把它设为新base然后从列表里移除它(这样就不会对
它重复计算了)当候选列表为空时result列表将包含所有素数精巧吧哈!
研究下:第次循环有什么特别吗？那时base 是2但它样经过了筛选为什
么？为什么这不发生在其它base值身上？我们打算移除product时能否确定它在候选列
表中呢？为什么？
接下来是什么呢？哦是……索引还有切片它们是从python列表中获得单个
元素思路方法你已经见到了普通索引行为它相当简单事实上我已经告诉了你所有
你需要知道有关它东西除了件事:负数索引从列表末尾向前计算所以
someList[-1]是someList最后个元素someList[-2]是它的前个元素依次类
推
切片仍然对你来说是陌生它和索引相似除了切片可以获得列表中所有
元素而不仅仅是单个元素这如何做呢？象这样:

```
food = [“spam”, “spam”, “eggs”, “sausages”, “spam”]
pr food[2:4]
# Prs “['eggs', 'sausages']”
```

10. 继续抽象 - 对象和面向对象编程
现在有个比较热门词叫做“面向对象编程”
就象本段标题暗示那样面向对象编程仅仅是另外种抽象细节方式通过

命名将简单描述抽象为复杂操作在面向对象编程时我们不仅可以这样对待
还可以把它们做为对象(现在这肯定会让你吃惊哈！)例如如果编写烧火腿程
序我们不用编写很多过程来处理温度、时间、成份等等我们可以把它们结合为个火
腿对象或者也许我们可以再有炉子对象和时钟对象……那么象温度这类事物就变成
了火腿对象个属性而时间可以从时钟对象读取要使用我们做某些事我们
可以教给我们对象某些思路方法；比如炉子应当知道如何烹制火腿等
那么 - - 在python中我们如何做呢？我们不能直接制造个对象不能直接制造个
炉子而是做个菜谱来描述炉子应该是什么样这份菜谱因此就描述了个被我们称为
炉子类对象个非常简单炉子类可能是这样:

```
Oven:
   def insertSpam(self, spam):
      self.spam = spam
   def getSpam(self):
       self.spam
```

这看起来很难理解还是怎样呢？
这个例子中新东西……
对象类用关键字定义
类名称通常以大写字母开始而和变量(还有属性和思路方法)名称以小写字
母开始
思路方法(也就是让对象知道如何去做和操作)定义没有特别但是要在类
定义里面
所有对象思路方法应当有第个参数叫做self(或者类似……)原因很快就清楚
了
对象属性和思路方法可以这样来访问:mySpam.temperature = 2 或者dilbert.be_nice

我能猜到上面例子中某些东西你仍然不清楚例如什么是self？还有现在我们
有了对象菜谱(也就是类)我们怎样事实上构造个对象呢？
我们先颠倒下顺序对象通过象引用那样引用类名来创建:

```
myOven = Oven
```

myOven包含了个Oven对象通常叫做Oven类个例子假设我们也构造好了
个Spam类那么我们可象这样做:

```
mySpam = Spam
myOven.insertSpam(mySpam)
```

myOven.spam现在将包含mySpam如何回事？我们个对象某个思路方法
时第个参数通常称为self总是包含对象本身(巧妙哈！)这样self.spam =spam这行设置当前Oven对象spam属性
值为参数spam注意它们是两个区别事物尽管在这个例子中它们都被称为spam
11. 练习3答案
这是这个运算法则个非常简洁版本:

```
def euclid(a, b):
    while b:
        a, b = b, a%b
     a
```

12. 参考
[1] 假日火腿沙拉菜谱摘自獭ormel Foods坏缱影娌似住?