## Python入门教程 超详细1小时学会Python

**为什么使用Python**

    假设我们有这么一项任务:简单测试局域网中的电脑是否连通.这些电脑的ip范围从192.168.0.101到192.168.0.200.
    思路:用shell编程.(Linux通常是bash而Windows是批处理脚本).例如,在Windows上用ping ip 的命令依次测试各个机器并得到控制台输出.由于ping通的时候控制台文本通常是"Reply from ... " 而不通的时候文本是"time out ... " ,所以,在结果中进行字符串查找,即可知道该机器是否连通.
    实现:Java代码如下:

String cmd="cmd.exe ping ";
String ipprefix="192.168.10.";
int begin=101;
int end=200;
Process p=null;
for(int i=begin;i<end;i++){
     p= Runtime.getRuntime().exec(cmd+i);
     String line = null;
     BufferedReader reader = new BufferedReader(new InputStreamReader(p.getInputStream()));
     while((line = reader.readLine()) != null)
     {
         //Handling line , may logs it.
     }
    reader.close();
    p.destroy();
}
    这段代码运行得很好,问题是为了运行这段代码,你还需要做一些额外的工作.这些额外的工作包括:

1. 编写一个类文件
2. 编写一个main方法
3. 将之编译成字节代码
4. 由于字节代码不能直接运行,你需要再写个小小的bat或者bash脚本来运行.

    当然,用C/C++同样能完成这项工作.但C/C++不是跨平台语言.在这个足够简单的例子中也许看不出C/C++和Java实现的区别,但在一些更为复杂的场景,比如要将连通与否的信息记录到网络数据库.由于Linux和Windows的网络接口实现方式不同,你不得不写两个函数的版本.用Java就没有这样的顾虑.
    同样的工作用Python实现如下:
import subprocess
cmd="cmd.exe"
begin=101
end=200
while begin<end:
    p=subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,
                   stdin=subprocess.PIPE,
                   stderr=subprocess.PIPE)
    p.stdin.write("ping 192.168.1."+str(begin)+"\n")
    p.stdin.close()
    p.wait()
    print "execution result: %s"%p.stdout.read()
    对比Java,Python的实现更为简洁,你编写的时间更快.你不需要写main函数,并且这个程序保存之后可以直接运行.另外,和Java一样,Python也是跨平台的.
    有经验的C/Java程序员可能会争论说用C/Java写会比Python写得快.这个观点见仁见智.我的想法是当你同时掌握Java和Python之后,你会发现用Python写这类程序的速度会比Java快上许多.例如操作本地文件时你仅需要一行代码而不需要Java的许多流包装类.各种语言有其天然的适合的应用范围.用Python处理一些简短程序类似与操作系统的交互编程工作最省时省力.

**Python应用场合**

    足够简单的任务,例如一些shell编程.如果你喜欢用Python设计大型商业网站或者设计复杂的游戏,悉听尊便.

2 快速入门

**2.1 Hello world**

    安装完Python之后(我本机的版本是2.5.4),打开IDLE(Python GUI) , 该程序是Python语言解释器,你写的语句能够立即运行.我们写下一句著名的程序语句:

print "Hello,world!"

    并按回车.你就能看到这句被K&R引入到程序世界的名言.
    在解释器中选择"File"--"New Window" 或快捷键 Ctrl+N , 打开一个新的编辑器.写下如下语句:

print "Hello,world!"
raw\_input("Press enter key to close this window");
    保存为a.py文件.按F5,你就可以看到程序的运行结果了.这是Python的第二种运行方式.
    找到你保存的a.py文件,双击.也可以看到程序结果.Python的程序能够直接运行,对比Java,这是一个优势.

**2.2 国际化支持**

    我们换一种方式来问候世界.新建一个编辑器并写如下代码:

print "欢迎来到奥运中国!"
raw\_input("Press enter key to close this window");
    在你保存代码的时候,Python会提示你是否改变文件的字符集,结果如下:

# -\*- coding: cp936 -\*-

print "欢迎来到奥运中国!"
raw\_input("Press enter key to close this window");
    将该字符集改为我们更熟悉的形式:

# -\*- coding: GBK -\*-

print "欢迎来到奥运中国!" # 使用中文的例子
raw\_input("Press enter key to close this window");
    程序一样运行良好.

**2.3 方便易用的计算器**

    用微软附带的计算器来计数实在太麻烦了.打开Python解释器,直接进行计算:

a=100.0
b=201.1
c=2343
print (a+b+c)/c

**2.4 字符串,ASCII和UNICODE**
    可以如下打印出预定义输出格式的字符串:

print """
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
"""
    字符串是怎么访问的?请看这个例子:

word="abcdefg"
a=word[2]
print "a is: "+a
b=word[1:3]
print "b is: "+b # index 1 and 2 elements of word.
c=word[:2]
print "c is: "+c # index 0 and 1 elements of word.
d=word[0:]
print "d is: "+d # All elements of word.
e=word[:2]+word[2:]
print "e is: "+e # All elements of word.
f=word[-1]
print "f is: "+f # The last elements of word.
g=word[-4:-2]
print "g is: "+g # index 3 and 4 elements of word.
h=word[-2:]
print "h is: "+h # The last two elements.
i=word[:-2]
print "i is: "+i # Everything except the last two characters
l=len(word)
print "Length of word is: "+ str(l)
    请注意ASCII和UNICODE字符串的区别:

print "Input your Chinese name:"
s=raw\_input("Press enter to be continued");
print "Your name is  : " +s;
l=len(s)
print "Length of your Chinese name in asc codes is:"+str(l);
a=unicode(s,"GBK")
l=len(a)
print "I'm sorry we should use unicode char!Characters number of your Chinese \
name in unicode is:"+str(l);

**2.5 使用List**

    类似Java里的List,这是一种方便易用的数据类型:

word=['a','b','c','d','e','f','g']
a=word[2]
print "a is: "+a
b=word[1:3]
print "b is: "
print b # index 1 and 2 elements of word.
c=word[:2]
print "c is: "
print c # index 0 and 1 elements of word.
d=word[0:]
print "d is: "
print d # All elements of word.
e=word[:2]+word[2:]
print "e is: "
print e # All elements of word.
f=word[-1]
print "f is: "
print f # The last elements of word.
g=word[-4:-2]
print "g is: "
print g # index 3 and 4 elements of word.
h=word[-2:]
print "h is: "
print h # The last two elements.
i=word[:-2]
print "i is: "
print i # Everything except the last two characters
l=len(word)
print "Length of word is: "+ str(l)
print "Adds new element"
word.append('h')
print word

**2.6 条件和循环语句**

# Multi-way decision
x=int(raw\_input("Please enter an integer:"))
if x<0:
    x=0
    print "Negative changed to zero"
elif x==0:
    print "Zero"
else:
    print "More"
# Loops List
a = ['cat', 'window', 'defenestrate']
for x in a:
    print x, len(x)

**2.7 如何定义函数**

# Define and invoke function.
def sum(a,b):
    return a+b
func = sum
r = func(5,6)
print r
# Defines function with default argument
def add(a,b=2):
    return a+b
r=add(1)
print r
r=add(1,5)
print r
    并且,介绍一个方便好用的函数:

# The range() function
a =range(5,10)
print a
a = range(-2,-7)
print a
a = range(-7,-2)
print a
a = range(-2,-11,-3) # The 3rd parameter stands for step
print a

**2.8 文件I/O**

spath="D:/download/baa.txt"
f=open(spath,"w") # Opens file for writing.Creates this file doesn't exist.
f.write("First line 1.\n")
f.writelines("First line 2.")
f.close()
f=open(spath,"r") # Opens file for reading
for line in f:
    print line
f.close()

**2.9 异常处理**

s=raw\_input("Input your age:")
if s =="":
    raise Exception("Input must no be empty.")
try:
    i=int(s)
except ValueError:
    print "Could not convert data to an integer."
except:
    print "Unknown exception!"
else: # It is useful for code that must be executed if the try clause does not raise an exception
    print "You are %d" % i," years old"
finally: # Clean up action
    print "Goodbye!"

**2.10 类和继承**

class Base:
    def \_\_init\_\_(self):
        self.data = []
    def add(self, x):
        self.data.append(x)
    def addtwice(self, x):
        self.add(x)
        self.add(x)
# Child extends Base
class Child(Base):
    def plus(self,a,b):
        return a+b
oChild =Child()
oChild.add("str1")
print oChild.data
print oChild.plus(2,3)

**2.11 包机制**

    每一个.py文件称为一个module,module之间可以互相导入.请参看以下例子:

# a.py
def add\_func(a,b):
    return a+b

# b.py
from a import add\_func # Also can be : import a

print "Import add\_func from module a"
print "Result of 1 plus 2 is: "
print add\_func(1,2)    # If using "import a" , then here should be "a.add\_func"
    module可以定义在包里面.Python定义包的方式稍微有点古怪,假设我们有一个parent文件夹,该文件夹有一个child子文件夹.child中有一个module a.py . 如何让Python知道这个文件层次结构?很简单,每个目录都放一个名为\_init\_.py 的文件.该文件内容可以为空.这个层次结构如下所示:

parent
  --\_\_init\_.py
  --child
    -- \_\_init\_.py
    --a.py
b.py
    那么Python如何找到我们定义的module?在标准包sys中,path属性记录了Python的包路径.你可以将之打印出来:

import sys
print sys.path
    通常我们可以将module的包路径放到环境变量PYTHONPATH中,该环境变量会自动添加到sys.path属性.另一种方便的方法是编程中直接指定我们的module路径到sys.path 中:

import sys
sys.path.append('D:\\download')
from parent.child.a import add\_func
print sys.path
print "Import add\_func from module a"
print "Result of 1 plus 2 is: "
print add\_func(1,2)

总结

    你会发现这个教程相当的简单.许多Python特性在代码中以隐含方式提出,这些特性包括:Python不需要显式声明数据类型,关键字说明,字符串函数的解释等等.我认为一个熟练的程序员应该对这些概念相当了解,这样在你挤出宝贵的一小时阅读这篇短短的教程之后,你能够通过已有知识的迁移类比尽快熟悉Python,然后尽快能用它开始编程.
    当然,1小时学会Python颇有哗众取宠之嫌.确切的说,编程语言包括语法和标准库.语法相当于武术招式,而标准库应用实践经验则类似于内功,需要长期锻炼.Python学习了Java的长处,提供了大量极方便易用的标准库供程序员"拿来主义".(这也是Python成功的原因),在开篇我们看到了Python如何调用Windows cmd的例子,以后我会尽量写上各标准库的用法和一些应用技巧,让大家真正掌握Python.
    但不管怎样,至少你现在会用Python代替繁琐的批处理写程序了.希望那些真的能在一小时内读完本文并开始使用Python的程序员会喜欢这篇小文章,谢谢!