Python 脚本编程及系统大规模自动化运维-Python语句和语法

Python 脚本编程及系统大规模自动化运维-Python语句和语法
- python语句
  - python的语句集
  - 语法格式
  - 赋值、表达式和打印
    - 赋值语句的形式
    - 表达式语句
    - 打印操作
      - python3.0
      - python2.6
      - 打印流重定向
      - 格式化打印
      - 课堂练习
  - if
    - 语法格式
    - 判断
    - 课堂练习
  - for
    - 语法格式
    - 课堂练习
  - while
    - 语法格式
    - 课堂练习
  - break\continue\pass和循环else
  - 循环计数器：range
  - 课堂练习
  - 函数
    - 函数的定义和调用
    - 参数
    - 局部变量
    - 全局变量
    - 作用域
    - Unbound异常
    - 参数默认值
    - 返回值
    - 多值返回
    - Doc
    - Lambda
    - 练习

> Python程序结构

- 程序由模块构成
- 模块包含语句
- 语句包含表达式

- 表达式建立并处理对象

python语句

python的语句集

Statement语句	**Role**角色	**Example**例子
Assignment赋值	Creating references创建引用值	a, b = 'good', 'bad'
Calls and other expressions调用	Running functions执行函数	log.write("spam, ham")
print calls打印调用	Printing objects打印对象	print('The Killer', joke)
if / elif / else	Selecting actions选择动作	if "python" in text: print(text)
for / else	Iteration序列迭代	for x in mylist: print(x)
while / else	General loops一般循环	while X > Y: print('hello')
pass	Empty placeholder空占位符	while True: pass
break	Loop exit循环退出	while True: if exittest(): break
continue	Loop continue循环继续	while True: if skiptest(): continue
def	Functions and methods函数和方法	def f(a, b, c=1, *d): print(a+b+c+d[0])
return	Functions results函数结果	def f(a, b, c=1, *d): return a+b+c+d[0]
yield	Generator functions生成器函数	def gen(n): for i in n: yield i*2
global	Namespaces命名空间	x = 'old' def function(): global x, y x = 'new'
nonlocal	Namespaces (3.X)	def outer():x = 'old' def function(): nonlocal x; x = 'new'
import	Module access模块访问	import sys
from	Attribute access属性访问	from sys import stdin
class	Building objects创建对象	class Subclass(Superclass): staticData = [] def method(self): pass
try / except / finally	Catching exceptions捕捉异常	try: action() except: print('action error')
raise	Triggering exceptions触发异常	raise EndSearch(location)
assert	Debugging checks调试检查	assert X > Y, 'X too small'
with / as	Context managers (3.X, 2.6+)环境管理器	with open('data') as myfile: process(myfile)
del	Deleting references删除引用	del data[k] del data[i:j] del obj.attr del variable

以上表格中需要注意的是：

- print在python3.0是一个内置的函数，2式，也是一个保留字
- python2.6中nonlocal不可用
- python3.0的exec代码执行内置函数是一条语句，具有特定语法
- python2.5 try/except 和 try/finally语句合并了

语法格式

1. python增加了冒号(:)
2. python中括号()是可选的
3. 终止行就是终止语句，不需要分号终止
4. 缩进的结束就是代码块的结束
5. 特殊实例`if x > y : print(x)`，只有简单语句才可
6. 建议单个制表符，两个或四个空格的缩进，不要混合使用制表符和空格

赋值、表达式和打印

- 赋值语句建立对象引用值 phython变量更像指针，而不是数据存储区域
- 变量名在首次赋值时会被创建
- 变量名在引用前必须先赋值
- 执行隐式赋值的一些操作

赋值语句的形式

运算	解释
`spam = 'Spam'`	基本形式
`spam,ham = 'yum','YUM'`	元组赋值运算
`[spam,ham] = ['yum','YUM']`	列表赋值运算
`a,b,c,d = 'spam'`	序列赋值运算，通用性
`a,*b = 'spam'`	扩展的序列解包python3.0
`spam = ham = 'lunch'`	多目标赋值运算
`spams += 42`	增强赋值运算相当于`spams = spams + 42`

表达式语句

通常在两种情况下表达式作用于语句：

- 调用函数和方法
- 爱交互模式提示符下打印值

常见的ython表达式语句

运算	解释
spam(eggs,ham)	函数调用
spam.ham(eggs)	方法调用
spam	在交互模式解释器内打印变量
print(a,b,c,sep='')	python3.0中的打印操作
yield x ** 2	产生表达式的语句

打印操作

在python中，print语句可以实现打印————只是对程序员友好的标准输出流的接口而已。

从技术角度，就是将一个或多个对象转化为其文本表达形式，然后发送给标准输出或另一个类似文件的流。
python中，打印与文件和流的概念紧密相连。

- 文件对象法 eg.写入文件 `file.wirte(str)` 和文件方法不同，在使用打印操作的时候，不需要把对象转换成字符串
- 标准输出流 stdout 只是发送一个程序的文本输出的默认地方。eg. `sys.stdout`

打印是python3.0和python2.6差异最明显的地方之一。

- python3.0中，打印是一个内置函数，用关键字参数来表示特定模式
- python2.6中，打印是语句，拥有自己的特定语法

python3.0

```
print([object, ...][,sep=' '][, end='\n'][, file=sys.stdout])
```

- 方括号中的项为可选的，并且可能会在一个给丁的调用中省略
- =后面的值都给出了参数的默认值
- 把字符串sep所分隔的一个或多个对象的文本表示
- 后面跟着的字符串end，都打印到流file中

python2.6

```
print_stmt ::= "print" ([expression ("," expression)* [","]]
| ">>" expression [("," expression)+ [","]])
```

python2.6语句	**python3.0**对等形式	说明
print x,y	print(x,y)	把对象的文本形式打印到sys.stdout，在各项之间添加一个空格，在末尾添加一个行末。
print x,y,	print(x,y,end='')	同样，但是不会在文本末尾添加行末
print >> afile,x,y	print(x,y,file=afile)	把文本发送到myfile.write而不是sys.stdout.write

打印流重定向

print语句值是python的人性化地特性，提供了sys.stdout对象地简单接口，再加上一些默认地格式设置。

```
In [2]: print "hello"
hello

In [3]: import sys

In [4]: sys.stdout.write('hello')
hello
In [5]: sys.stdout.write('hello\n')
hello
```

`print x,y` 等价于

```
import sys
sys.stdout.write(str(X)+' '+str(Y)+'\n')
```

> 往文件log.txt中追加内容 2.6

```
In [7]: log=open('log.txt','a')

In [8]: print >> log,'hello word'

In [9]: log.close()

In [10]: open('log.txt').read()
Out[10]: '123\nhello word\n'

In [11]: print open('log.txt').read()
123
hello word
```

格式化打印

```
In [12]: x='a'

In [13]: y='b'

In [14]: z='c'

In [15]: print('%s %s %s' % (x,y,z))
a b c

In [16]: print('{0} {1} {2}'.format(x,y,z))
a b c
```

课堂练习

> ipython2.7中完成

1. 举出三种可以把三个变量赋值成相同值的方法

2. 打印出hello word，分别使用sys.stdou()和print
3. 将hello word追加到文件myfile.txt中，并输出myfile.txt的所有行

> 答案

1. `a=b=c=0` 或 `a=0;b=0;c=0` 或 `a,b,c=0,0,0`
2.

```
print "hello word"
import sys
sys.stdout.write('hello'+' '+'word'+'\n')
```

3.

```
myfile=open('myfile.txt','a')
print >> myfile,"hello word"
myfile.close()
print open('myfile.txt','r').read()
```

---

if

语法格式

```
if <test1>:
    <statements1>
elif <test2>:
    <statements2>
else:
    <statements2>
```

判断

数字的判断	符号
等于	==
大于	>
小于	<
大于等于	>=
小于等于	<=
不等于	!=

字符的判断	符号
等于	==
包含	in
不包含	not in

逻辑运算	符号
与	and
或	or

课堂练习

1. 写一个脚本，判断用户是否存在，如果存在则删除。若不存在，就提示不存在。
2. 三个数字比大小，输出最大的
3. 三个数字比大小,并且按从大到小排列

1.

```
#!/usr/bin/env python
import subprocess
test=subprocess.call(['id','booboo'])
if test==0:
    subprocess.call(['userdel','-r','booboo'])
else:
    print('no such user')
```

2.

```
#!/usr/bin/env python
a = raw_input("plz input a:")
b = raw_input("plz input b:")
c = raw_input("plz input c:")

if a>b:
    if a>c:
        print('max is %s' % a)
    else:
        print('max is %s' % c)
else:
    if b>c:
        print('max is %s' % b)
    else:
        print('max is %s' % c)
```

```
#!/usr/bin/env python
a = raw_input("plz input a:")
b = raw_input("plz input b:")
c = raw_input("plz input c:")
num_list = [a,b,c]
max = 0
for i in num_list:
    if i > max:
        max = i
print('max={0}'.format(max))
```

3.

```
#!/usr/bin/env python
a=3
b=9
c=0
if a>b :
    n1=a
    n2=b
else:
    n1=b
    n2=a

if n1>c:
    max=n1
    if n2>c:
```

```
            be=n2
            min=c
        else:
            be=c
            min=n2
    else:
        max=c
        be=n1
        min=n2

    print('{0}>{1}>{2}'.format(max,be,min))
    c=0
    if a>b :
        n1=a
        n2=b
    else:
        n1=b
        n2=a

    if n1>c:
        max=n1
        if n2>c:
            be=n2
            min=c
        else:
            be=c
            min=n2
    else:
        max=c
        be=n1
        min=n2

    print('{0}>{1}>{2}'.format(max,be,min))
```

for

语法格式

```
for <target> in <object>
    <statements1>
else:
    <statements>
```

课堂练习

1. 9*9乘法表

```
import sys
for i in range(1,10):
    for j in range(1,i+1):
        sys.stdout.write('{0}*{1}={2} '.format(i,j,i*j))
    print('')

1*1=1
2*1=2 2*2=4
3*1=3 3*2=6 3*3=9
4*1=4 4*2=8 4*3=12 4*4=16
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25
6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

1. 遍历字典中的键和值

```
#!/usr/bin/env python
D = {'username':'superman','age':22,'mail':'superman@hotmail.com'}
for key in D:
    print('{0}===>{1}'.format(key,D[key]))

执行结果
username===>superman
mail===>superman@hotmail.com
age===>22
```

使用字典的items()方法来实现

```
In [321]: for key,value in dict.items():
   .....:     print("dict[{0}]={1}".format(key,value))
   .....:
dict[a]=1
dict[b]=2
```

1. 逐行读取文本文件

```
#!/usr/bin/env python
for line in open('test.txt').readlines():
    print line,
for line in open('test.txt'):
    print line,
```

- 文件readlines方法会一次把文件载入到行字符串的列表
- 第二种方法时使用文件迭代器来自动在每次循环迭代的时候读入一行

while

语法格式

```
while <test>:
    <statements1>
else:
    <statements2>
```

课堂练习

1. 连乘算法 while

```
#!/usr/bin/env python
import sys
import string
n_str=sys.argv[1]
n=string.atoi(n_str)
s=1
i=1
while i<=n:
        s=s*i
        i=i+1
print s
```

1. 要求根据userlist创建用户，要求指定用户名，用户id，用户的附加组及变更用户u密码，若对应用户的附加组不存在，则将附加组创建出来后再根据要求添加用户。
   userlist文件的格式如下：

carol 777 tom uplooking

natasha 778 tom uplooking

r1 779 tom uplooking

```
#!/usr/bin/env python
from subprocess import *
from cStringIO import StringIO
file_string=open('/root/userlist','r').read()
file_line=StringIO(file_string)
for line in file_line:
    line_list=line.strip().split(' ')
    call(['groupadd',line_list[2]])
    call(['useradd','-u',line_list[1],'-G',line_list[2],line_list[0]])
    p1=Popen(['echo',line_list[3]],stdout=PIPE)
    p2=Popen(['passwd','--stdin',line_list[0]],stdin=p1.stdout,stdout=PIPE)
    call(['id',line_list[0]])
```

1. 请问你是否喜欢shell脚本？ 如果你回答yes，则程序退出，否则永远会问你是否喜欢shell脚本？

```
#!/usr/bin/env python
an=raw_input('Do you like python?(plz input yes) : ')
while an != "yes":
    if an == "yes":
        print('I like python too!')
    if an != "yes" :
        an=raw_input('Do you like python?(plz input yes) : ')
```

break\continue\pass和循环else

- break　　跳出最近所在的循环（跳过整个循环语句）
    - continue 跳到最近所在循环的开头处（来到循环的首行）
        - pass什么事情也不做，只是空占位语句
    - 循环else块只有当循环正常离开时才会执行（也就是说没有碰到break语句）

break和continue可以出现在while或for循环主体的任何地方，但通常会进一步嵌套if语句中。

加入break和continue语句后，它们的一般格式为：

```
while <test1>:
    <statements1>
    if <test2>: break
    if <test3>: continue
else:
    <statements2>
```

pass的使用场景：

```
while true: pass

def func1():
    pass
def func2():
    pass
```

循环计数器：range

- range函数是通用的工具，range会产生从0算起的数据列表，但其中不包括该参数的值。
- 如果传进两个参数，第一个将视为下边界，第三个选用参数也一提供步进值。
- 对每个连续整数加上步进值从而得到结果，默认为1

```
In [41]: range(5)
Out[41]: [0, 1, 2, 3, 4]

In [42]: range(1,5)
Out[42]: [1, 2, 3, 4]

In [43]: range(1,5,2)
Out[43]: [1, 3]

In [44]: range(5,1,-1)
Out[44]: [5, 4, 3, 2]

In [45]: range(5,1,-2)
Out[45]: [5, 3]

In [50]: for i in x: print('hello {0}'.format(i))
hello h
hello e
hello l
hello l
hello o
```

课堂练习

1. 将列表`L=[1,2,3,4]`中的每个值都加`2`

```
In [60]: L=[1, 2, 3, 4]

In [61]: for i in range(len(L)):
   ....:     L[i]+=2
   ....:

In [62]: L
Out[62]: [3, 4, 5, 6]
```

课堂练习

1. 写一个for循环，打印字符串`S='batman'`中的每个字符的`ASCII码`。使用内置函数`ord(character)`把每个字符转换成ASCII整数。
2. 接着，修改循环来计算字符串中每个字符的ASCII码的总和
3. 最后，再次修改代码，来返回一个新的列表，其中包含了字符串中每个字符的ASCII码。

```
In [8]: S='batman'

In [9]: for i in S: print ord(i)
98
97
116
109
97
110

In [11]: sum = 0

In [12]: for i in S:
   ....:     sum = sum + ord(i)
   ....:

In [13]: print sum
627

In [15]: list=[]

In [16]: for i in S:list.append(ord(i))

In [17]: list
Out[17]: [98, 97, 116, 109, 97, 110]
```

1. 写一个for循环，对数字0到49，分别问好 `hello 0` 一直到 `hello 49`

```
for i in range(50) :
    print('hello %d' % i)
```

1. 写一个for循环排序后（递增）顺序打印字典dic={'a':9,'c':8,'b':2}的项目。提示：使用字典keys和列表sort方法，或者sorted内置函数

```
>>> dic={'a':9,'c':8,'b':2}
>>> for i in sorted(dic):
...     print i,dic[i]
...
a 9
b 2
c 8
```

函数

函数的定义和调用

做代码复用

定义：

```
def say_hello():
    print('hello world')
```

调用：

```
say_hello()
```

> 课堂练习：写一个函数输出hello，并调用该函数

```
[root@workstation0 ~]# python h1.py
hello
[root@workstation0 ~]# cat h1.py
def say_hello():
    print('hello')
say_hello()
```

参数

可以通过参数向函数传递数据。

> 课堂练习：写一个函数，比较两个数字大小，谁大输出谁，相等就输出相等。

```
def print_max(a, b):
    if a > b:
        print('{0} is maximum'.format(a))
    elif a == b:
        print('{0} is equal to {1}'.format(a, b))
    else:
        print('{0} is maximum'.format(b))
print_max(3, 4)
```

```
print_max(3, '4')
print_max('3', 4)
```

```
[root@workstation0 ~]# python h2.py
4 is maximum
4 is maximum
3 is masimum
[root@workstation0 ~]# cat h2.py
def print_max(a,b):
    if a>b:
        print('{0} is masimum'.format(a))
    elif a==b:
        print('{0} is equal to {1}'.format(a,b))
    else:
        print('{0} is maximum'.format(b))
print_max(3,4)
print_max(3,'4')
print_max('3',4)
```

局部变量

- 在函数体内定义的变量称为局部变量。
- 局部变量在函数执行时产生，在函数结束时消亡。函数外无法看到局部变量。
- 参数可视为一种局部变量。
- 对局部变量的修改不影响全局变量。

> 课堂练习：设置全局变量`x=50`，函数`func1()`中设定局部变量`x=2`，并打印x变量的值；函数`func2(X)`打印x变量的值；调用函数`func1()`和`func2(3)`以及直接输出`x`，查看并分析结果。

```
[root@workstation0 ~]# cat h3.sh
x=50
def func1():
    x=2
    print('local x is {0}'.format(x))
def func2(x):
    print('local x is {0}'.format(x))
func1()
func2(3)
print(x)
[root@workstation0 ~]# python h3.sh
local x is 2
local x is 3
50
```

全局变量

- 在模块内定义的变量叫全局变量。
- 全局变量在全局可见，函数体内可以引用全局变量。
- 函数可以用global关键字声明某变量是全局的。

> 课堂练习： 全局变量x=50，在函数func()中的局部变量x=2，查看程序执行结果。

```
[root@workstation0 ~]# cat h4.sh
x = 50
def func():
    global x
    print('x is {0}'.format(x))
    x = 2
    print('Changed x to {0}'.format(x))
print('x is {0} before function.'.format(x))
func()
print('x is {0} after function.'.format(x))
[root@workstation0 ~]# python h4.sh
x is 50 before function.
x is 50
Changed x to 2
x is 2 after function.
```

局部变量覆盖了全局变量

作用域

- 变量可见的范围叫做变量的作用域。
- 局部变量的作用域在函数内，全局变量的作用域在模块内（关于模块后文会讲）。
- 作用域的基本原则是，内层作用域可访问外层作用域的变量，反之外层不可访问内层变量。
- 如果两个作用域内有同一个名字，那么内层的起作用。

Unbound异常

- python内的“local中有变量定义”，是以local中存在对变量构建的语法（例如赋值）为标准的。
- 这个构建语法所在行可能尚未执行（或永远无法执行到），但是这并不影响“local中有变量定义”这个事实。

> 当local中定义了变量，但是变量尚未完成构建时，对变量做引用，会发生什么？

```
[root@workstation0 ~]# cat h5.sh
my_variable = 1

def func():
    print(my_variable)
    my_variable = 2

func()
[root@workstation0 ~]# python h5.sh
Traceback (most recent call last):
  File "h5.sh", line 7, in <module>
    func()
  File "h5.sh", line 4, in func
    print(my_variable)
UnboundLocalError: local variable 'my_variable' referenced before assignment
```

参数默认值

- 函数定义时，可以给参数一个值。当调用时，该参数并未给出的情况下，使用默认值。
- 提示：从安全性角度来说，不要使用可变对象作为参数默认值。

> 课堂练习：写一个函数，重复说话，默认只说1遍。`say(message, times=1)`其中message是要说的话，times是说的次数

```
def say(message, times=1):
    print(message * times)

say('Hello')
say('World', 5)
执行结果：
Hello
WorldWorldWorldWorldWorld
```

遇到希望使用可变对象作为参数默认值的情况，需要使用None作为默认值。

在函数中判断是否为None，是的话再对参数做默认值赋值处理。

以此模式来避免上述问题。

返回值

- 将函数内数据传递回去。

> 课堂练习：写一个函数实现两个数字比大小，返回最大值，否则返回一样大。

```
[root@workstation0 ~]# cat h7.py
def maximum(x, y):
    if x > y:
        return x
    elif x == y:
        return 'The numbers are equal'
    else:
        return y

print(maximum(2, 3))
print(maximum(2, 2))
[root@workstation0 ~]# python h7.py
3
The numbers are equal
```

多值返回

- Python允许一次返回多个值。
- 多值返回的本质是使用tuple中转（后面详细讲解）。

> 课堂练习： 函数`f()`返回多个值`1,2,3,4,5`，观察`print (f())`和`a,b,c,d,e = f();print(c)`的结果

```
[root@workstation0 ~]# cat h8.py
def f():
    return 1,2,3,4,5
print (f())

a,b,c,d,e = f()
print(c)
[root@workstation0 ~]# python h8.py
(1, 2, 3, 4, 5)
3
```

Doc

- Python允许为每个函数增加一个文档字符串，这个文档字符串可以被代码访问。
- 具体方式是，在函数的第一个逻辑行定义一个字符串（不执行任何操作）。
- 访问方式是function.**doc**。

> 课堂练习： 函数`print_max(x, y)`求两个数的最大值，为该函数写一个文档字符串，并访问该文档字符串。

```
[root@workstation0 ~]# cat h9.py
def print_max(x, y):
    '''Prints the maximum of two numbers. \nThe two values must be integers.'''
    x = int(x)
    y = int(y)
    if x > y:
        print('{} is maximum'.format(x))
    else:
        print('{} is maximum'.format(y))

print_max(3, 5)
print(print_max.__doc__)
[root@workstation0 ~]# python h9.py
5 is maximum
Prints the maximum of two numbers.
The two values must be integers.
```

Lambda

- 一种快速定义简单函数的方法。
- 函数名=lambada 变量名: return

函数格式：

- lambda_form ::= "lambda" [parameter_list]: expression
- old_lambda_form ::= "lambda" [parameter_list]: old_expression

> 课堂练习：写一个函数实现数字加1的结果，函数名为`inc(n)`，返回`n+1`。

```
[root@workstation0 ~]# cat h10.py
def inc(n):
    return n+1
print(inc(10))

# 函数名=lambada 变量名: return
inc = lambda n: n+1
print(inc(10))
[root@workstation0 ~]# python h10.py
11
11
```

练习

定义fib函数。

输入n，fib的输出为数列第n项的值。

数列某项等于前两项之和。

```
def fib(n):
    if n <= 1: return 1
    return fib(n-1) + fib(n-2)

fib(10)
```

输出为数列第n项以及之前所有项的值

```
In [21]: list=[]

In [22]: for i in range(11):
   ....:     list.append(fib(i))
   ....:

In [24]: print list
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

In [25]: list1=[fib(i) for i in range(11)]

In [26]: list1
Out[26]: [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```