

干货分享 | Python从入门到编写POC之读写文件

i春秋 2月6日



自发布了Python介绍及安装使用的相关文章后，深受大家喜爱，遂将Python从入门到编写POC系列教程的其他章节与大家分享，旨在快速提升安全技能！



学习中有任何疑问的小伙伴欢迎进群咨询，群里会有大佬耐心解答，同时i春秋不定期会举办直播讲座，帮助大家了解安全行业动态趋势，解锁更多实用技能！

快速进群入口





今天与大家分享的是Python从入门到编写POC系列文章之读写文件，希望对大家学习有所帮助。

注：i春秋公众号旨在为大家提供更多的学习方法与技能技巧，文章仅供学习参考。

读写文件

读取键盘输入【raw_input()或者input()】

```
1 >>> demo = raw_input('INPUT:'); \ # \是在CMD中的换行符
2 ... print "content is" , demo
3 INPUT:HELL0 MOMO
4 content is HELL0 MOMO
```

```
>>> demo = raw_input('INPUT:'); \
... print "content is" , demo
INPUT:HELL0 MOMO
content is HELL0 MOMO
```

打开或关闭文件，这里要用Python的内置函数open()，然后创建一个file对象。

Python打开文件的模式：

- r 以只读模式打开文件。
- w 以只写模式打开文件，且先把文件内容清空（truncate the file first）wb 以二进制格式打开一个文件只用于写入。如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。
- a 以添加模式打开文件，写文件的时候总是写到文件末尾，用seek也无用。打开的文件也是不能读的。
- r+ 以读写方式打开文件，文件可读可写，可写到文件的任何位置。
- w+ 和r+不同的是，它会truncate the file first。
- a+ 和r+不同的是，它只能写到文件末尾。

一个文件被打开后，有一个file对象，可以得到有关该文件的各种信息，以下是一些使用方法：

- file.closed 返回true如果文件已被关闭，否则返回false。
- file.mode 返回被打开文件的访问模式。
- file.name 返回文件的名称。
- file.softspace 如果用print输出后，必须跟一个空格符，则返回false。否则返回true。

举个例子

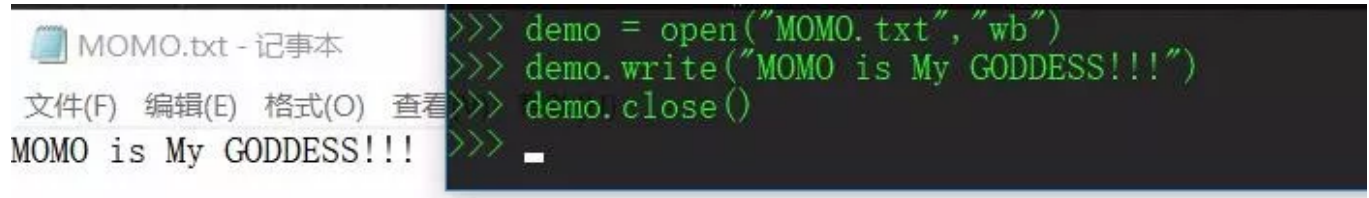
```
1 >>> demo = open("MOMO.txt","wb")
2 >>> print u'是否关闭: ', demo.closed
3 是否关闭: False
4 >>> print u'访问模式:',demo.mode
5 访问模式: wb
6 >>> print u'文件名称: ',demo.name
7 文件名称: MOMO.txt
8 >>> print u'末尾是否加空格: ',demo.softspace
9 末尾是否加空格: 0
```

 MOMO.txt

```
>>> demo = open("MOMO.txt","wb")
>>> print u'是否关闭: ', demo.closed
是否关闭: False
>>> print u'访问模式:',demo.mode
访问模式: wb
>>> print u'文件名称: ',demo.name
文件名称: MOMO.txt
>>> print u'末尾是否加空格: ',demo.softspace
末尾是否加空格: 0
```

在txt中写入东西

```
1 >>> demo = open("MOMO.txt","wb")
2 >>> demo.write("MOMO is My GODDESS!!!")
3 >>> demo.close()
```



读txt中的内容

```
1 >>> demo1 = open("MOMO.txt", "r")
2 >>> demo1.readlines()
3 ['MOMO is My GODDESS!!!']
```

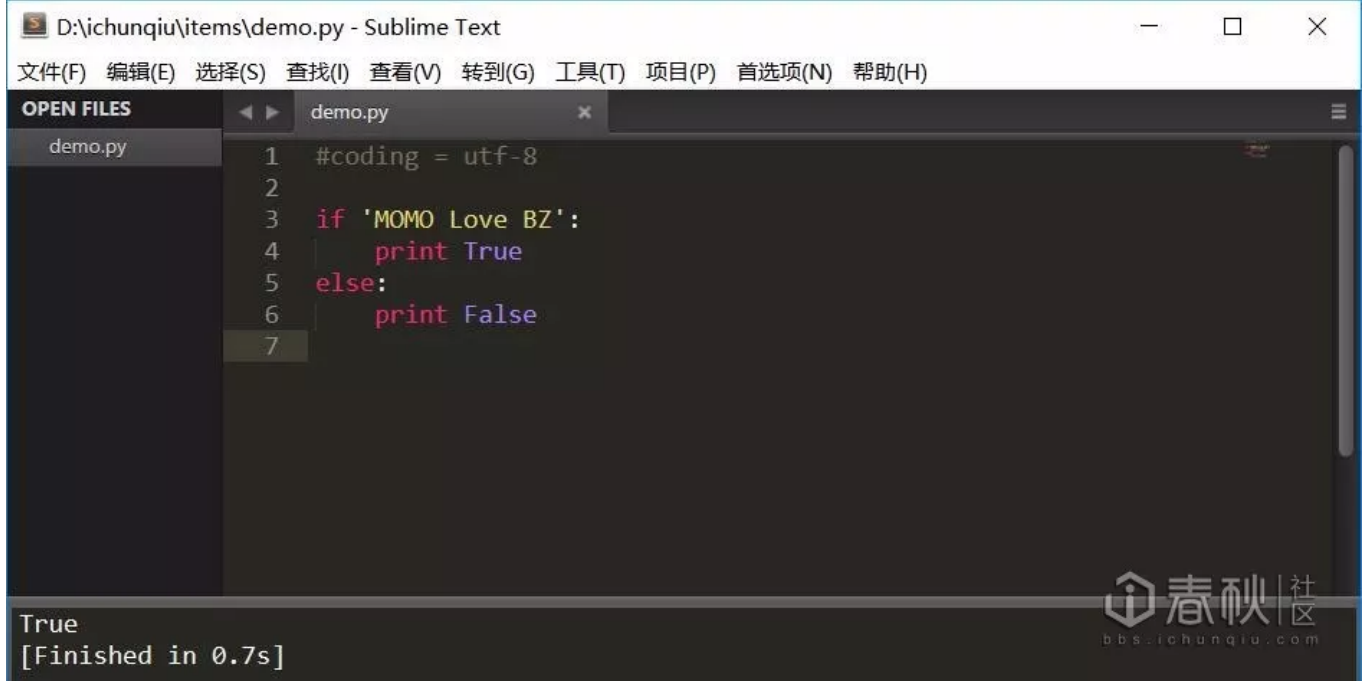
```
>>> demo1 = open("MOMO.txt", "r")
>>> demo1.readlines()
['MOMO is My GODDESS!!!']
```

判断

我们开始用编辑器，这里小编用的编辑器是Sublime Text3。

咱们讲的是Python判断，先来看一个例子：

```
1 #coding = utf-8
2
3 if 'MOMO' Love BZ':
4     print True
5 else:
6     print False
```



The screenshot shows a Sublime Text editor window titled "D:\ichunqiu\items\demo.py - Sublime Text". The menu bar includes "文件(F)", "编辑(E)", "选择(S)", "查找(I)", "查看(V)", "转到(G)", "工具(T)", "项目(P)", "首选项(N)", and "帮助(H)". The "OPEN FILES" sidebar on the left shows "demo.py". The main editor area displays the following Python code:

```
1 #coding = utf-8
2
3 if 'MOMO Love BZ':
4     print True
5 else:
6     print False
7
```

The output at the bottom of the window is:

```
True
[Finished in 0.7s]
```

A watermark for "春秋社区" (Chunqiu Community) is visible in the bottom right corner of the editor area.

当判断条件成立时（非零），则执行后面的语句，所以就返回了True。

if 语句的判断条件可以用>（大于）、<（小于）、==（等于）、>=（大于等于）、<=（小于等于）来表示其关系。

循环

While循环

格式为：

```
1 while 判断条件:
2     执行语句.....
```

举个例子：

```
1 #coding=utf-8
2
3 port = 1100
4
5 while port < 1109:
6     print "The PORT is:"+str(port)
7     port = port + 1
```

```
demo.py
1  #coding=utf-8
2
3  port = 1100
4
5  while port < 1109:
6      print "The PORT is:" + str(port)
7      port = port + 1
8
The PORT is:1100
The PORT is:1101
The PORT is:1102
The PORT is:1103
The PORT is:1104
The PORT is:1105
```

PS：如果没有port=port+1，该循环将成为无限循环，通过Ctrl+C终止，当然了，while循环也有else语句，方法跟 if 语句差不多，这里就不演示了还有break和continue语句。

在循环过程中，break语句可以提前退出循环。

通过continue语句，跳过当前的这次循环，直接开始下一次循环。

不过根据经验来说，While循环出现的次数没有For循环多。

For循环

格式为：

```
1  for iterating_var in sequence:
2      statements(s)
```

For循环的好处呢，比如你要算1+2+3+4+5手写还有解决。

当然也可以用Python，这里用到的就是For循环。

```
1  #coding = utf-8
2
3  sum = 0
4
```

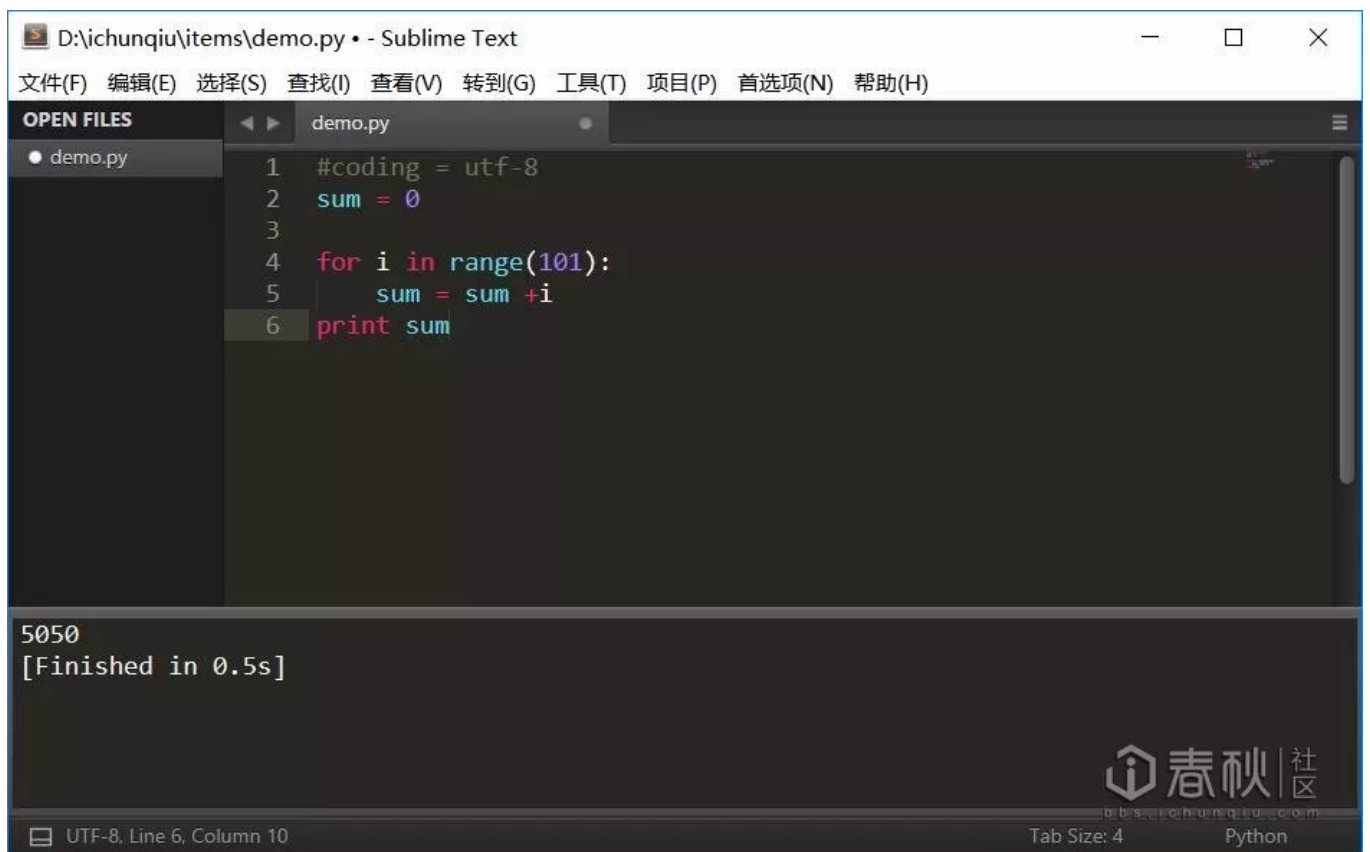
```
5 for i in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
6     sum = sum + i
7
8 print sum
```

那如果是 $1+2+3+4+\dots+100$ 呢？

这里咱们还用For循环，不过加了range()函数，可以生成一个整数数列：

range(101)生成1-100的数

```
1 #coding = utf-8
2
3 sum = 0
4
5 for i in range(101):
6     sum = sum +i
7 print sum
```



The screenshot shows a Sublime Text editor window titled "D:\ichunqiu\items\demo.py • - Sublime Text". The menu bar includes "文件(F)", "编辑(E)", "选择(S)", "查找(I)", "查看(V)", "转到(G)", "工具(T)", "项目(P)", "首选项(N)", and "帮助(H)". The "OPEN FILES" panel on the left shows "demo.py". The main editor area displays the following Python code:

```
1 #coding = utf-8
2 sum = 0
3
4 for i in range(101):
5     sum = sum +i
6 print sum
```

The output console at the bottom shows "5050" and "[Finished in 0.5s]". The status bar at the bottom indicates "UTF-8, Line 6, Column 10", "Tab Size: 4", and "Python". A watermark for "春秋社区" (Chunqiu Community) is visible in the bottom right corner.

理解函数

在初中时代，数学课本是可以这么定义函数的： $y=8x+6$

在高中时代，就可以这么定义函数了： $f(x)=8x+8$

其中， x 是什么呢？是变量，你想让它是啥数就是啥数。

当 $x=6$ 时， $f(6)=8*6+8=56$

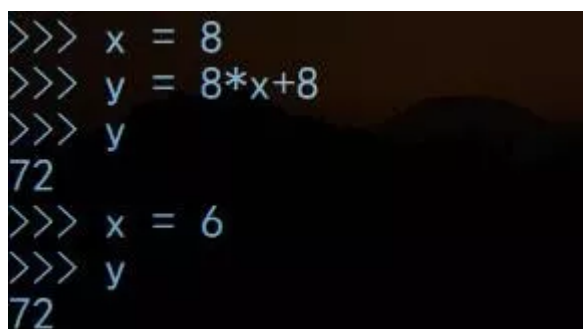
But,这并不是全部，其实在函数中，并没有规定函数是一个数，它可是任何东西。

变量的本质，你可以把他当作一个占位符，建立一个简单的函数，咱们现在Python的解释器操作：

```
1 >>> x = 8
2 >>> y=8*x+8
3 >>> y
4 72
```

咱们在继续

```
1 >>> x = 6
2 >>> y
3 72
```



```
>>> x = 8
>>> y = 8*x+8
>>> y
72
>>> x = 6
>>> y
72
```

当 x 等于6时了， $y=8*x+8$ 为啥还是72呢？

$x=8$ 时含义将8这个对象贴上了变量 x 的标签，经过计算得到了72，然后变量 y 就引用了72；当 x 值变了时，但是 y 引用得对象还没有变，所以就还是72。

这里还“不正规”，咱们接下来就学习定义一个“正规”函数。

函数

代码块以 `def` 关键词开头，`def` 这个简写来自 `define`，后接函数标识符名称和圆括号 `()`。

定义函数内容以冒号起始，然后必须缩进（四个空格或者一个tab键），定义函数格式为：

```
1 def 函数名(参数):  
2     函数体
```

举个例子：

```
1 >>> def MOMO():  
2 ...     print 'HELLØ MOMO!!'  
3 ...  
4 >>> MOMO() #调用MOMO这个函数  
5 HELLØ MOMO!!
```



在定义函数的时候，参数可以等你被赋值，也可以定义一个默认值。

```
1 #coding = utf-8  
2  
3 def MOMO(a,b='BaZong'):  
4     print "a:",a  
5     print "b:",b  
6  
7 MOMO('MOMO')  
8 print "-----"  
9 MOMO(a='MOMO')
```

```
OPEN FILES  demo.py x
demo.py
1 #coding = utf-8
2
3 def MOMO(a,b='BaZong'):
4     print "a:",a
5     print "b:",b
6
7 MOMO('MOMO')
8 print "-----"
9 MOMO(a='MOMO')
```

```
a: MOMO
b: BaZong
-----
a: MOMO
b: BaZong
[Finished in 0.2s]
```

春秋社区
bbs.ichunqiu.com

如果我不给变量b定义一个值，那么就会用BaZong这个值，那如果它传了一个新的值，那就会用新的那个值。

```
D:\ichunqiu\items\demo.py - Sublime Text
文件(F) 编辑(E) 选择(S) 查找(I) 查看(V) 转到(G) 工具(T) 项目(P) 首选项(N) 帮助(H)
OPEN FILES  demo.py x
demo.py
1 #coding = utf-8
2
3 def MOMO(a,b='BaZong'):
4     print "a:",a
5     print "b:",b
6
7 MOMO('MOMO')
8 print "-----"
9 MOMO('MOMO','Afu')
```

```
a: MOMO
b: BaZong
-----
a: MOMO
b: Afu
[Finished in 0.1s]
```

春秋社区
bbs.ichunqiu.com

ASCII, Line 9, Column 18; Build finished Tab Size: 4 Python

返回值

是不是感觉跟你见到的高大上的Python有一点差别，也许你们会注意到上面都是用print输出结果，用个求绝对值的栗子来引用返回值return。

```
1 #coding = utf-8
2
3 def my_abs(x):
4     if x >= 0:
5         return x
6     else:
7         return -x
8 if __name__ == '__main__':
9     number = 666
10    print number
```

仔细看my_abs()函数，有return x和return -x，意思就是将x/-x的值返回，返回给谁呢？

一般情况，要将返回的值传给一个变量，然后通过变量打印出来，如果没有，那就没有回显了。

return还有个作用，咱们在来通过一个例子看，我这里导入了base64模块。

```
1 #coding = utf-8
2
3 import base64
4
5 def demo():
6     str = "TU9NTyBpcyBhIGJlYXV0aWZ1bCBnaXJs"
7     result = base64.b64decode(str)
8     print result
9     return
10    print "KISS MOMO"
11
12 demo()
```

```
OPEN FILES
demo.py
1 #coding = utf-8
2
3 import base64
4
5 def demo():
6     str = "TU9NTyBpcyBhIGJlYXV0awZ1bCBnaXJs"
7     result = base64.b64decode(str)
8     print result
9     return
10    print "KISS MOMO"
11
12 demo()
13
14
```

MOMO is a beautiful girl
[Finished in 0.1s]

春秋社区
bbs.chunqiu.com

两个print语句，只执行了return上面的那个，这是为什么呢？

第一个print语句遇到return，return告诉他自己要回家，即为返回，终止了第一个print的路程；然后就没有执行第二个print语句，你可以把它当成循环中的break。


全局变量and局部变量

通过例子来说明：

```
1 #coding = utf-8
2
3 a = 'momo'
4 def demo():
5     a = 'BaZong'
6     print "this a is",a
7 demo()
8 print "-----"
9 print "this a is",a
```

```
OPEN FILES demo.py x
demo.py
1 #coding = utf-8
2
3 a = 'momo'
4 def demo():
5     a = 'BaZong'
6     print "this a is",a
7 demo()
8 print "-----"
9 print "this a is",a

this a is BaZong
-----
this a is momo
[Finished in 0.1s]
```




前一个print输出的函数内部的变量，后一个是外部的，像这样的，只在函数体内（某个范围内）起作用的就叫局部变量。

全局变量

```
1 #coding = utf-8
2
3 a = 'momo'
4 def demo():
5     global a
6     a = 'BaZong'
7     print "this a is",a
8 demo()
9 print "-----"
10 print "this a is",a
```

```
OPEN FILES demo.py x
demo.py
1 #coding = utf-8
2
3 a = 'momo'
4 def demo():
5     global a
6     a = 'BaZong'
7     print "this a is",a
8 demo()
9 print "-----"
10 print "this a is",a

this a is BaZong
-----
this a is BaZong
[Finished in 0.1s]
```



这两个例子区别就是多了个global a，意思就是让a是全局变量，也就是说，外面跟里面的a都一样。

学任何一门程序语言的窍门都是要多练，多琢磨。

文章素材来源于i春秋社区

以上是今天分享的内容，大家看懂了吗？后面我们将持续分享 Python 系列相关文章，请大家及时关注。

上 期 分 享

涨姿势



猜 你 喜 欢