[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON2/DAY03/CASE/01/index.html" \l "page_top_case)

# NSD Python2 DAY03

1. [案例1：编写游戏人物](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON2/DAY03/CASE/01/index.html" \l "case1)
2. [案例3：出版商程序](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON2/DAY03/CASE/01/index.html" \l "case2)
3. [案例4：分析apache访问日志](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON2/DAY03/CASE/01/index.html" \l "case3)

## 1 案例1：编写游戏人物

### 1.1 问题

创建程序，要求如下：

1. 创建游戏角色类
2. 游戏人物角色拥有名字、武器等属性
3. 游戏人物具有攻击和行走的方法
4. 武器通过武器类实现

### 1.2 方案

由于游戏角色和武器都由类实现，而这个两个类又完全不同，所以可以通过组合来实现。将武器类的一个实例作为游戏人物的一个属性。

### 1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

1. [root@localhost day07]# vim game\_role.py
2. class Weapon:
3. def \_\_init\_\_(self, wname, strength):
4. self.wname = wname
5. self.strength = strength
6. class Warrior:
7. def \_\_init\_\_(self, name, weapon):
8. self.name = name
9. self.weapon = weapon
10. def speak(self, words):
11. print("I'm %s, %s" % (self.name, words))
12. def show\_me(self):
13. print("我是%s, 我是一个战士。我用的武器是%s" % (self.name, self.weapon.wname))
14. if \_\_name\_\_ == '\_\_main\_\_':
15. blade = Weapon('青龙偃月刀', 100)
16. print(blade.wname, blade.strength)
17. gy = Warrior('关羽', blade)
18. gy.show\_me()
19. cz = Weapon('禅杖', 100)
20. lzs = Warrior('鲁智深', cz)
21. lzs.show\_me()

步骤二：测试脚本执行

1. [root@localhost day07]# python3 game\_role.py
2. 青龙偃月刀 100
3. 我是关羽, 我是一个战士。我用的武器是青龙偃月刀
4. 我是鲁智深, 我是一个战士。我用的武器是禅杖

## 2 案例3：出版商程序

### 2.1 问题

创建books.py文件，实现以下目标：

1. 为出版商编写一个Book类
2. Book类有书名、作者、页数等属性
3. 打印实例时，输出书名
4. 调用实例时，显示该书由哪个作者编写

### 2.2 方案

创建一个类，类中创建3种魔法方法：

1.\_\_init\_\_方法：\_\_init\_\_方法用于初始化实例属性，创建对象后会自动调用\_\_init\_\_方法，属于构造器方法，此处初始化了书名及作者两个属性

2.\_\_str\_\_方法：创建对象后，打印实例对象pybook，返回书名，打印出书名

3.\_\_call\_\_方法：创建对象后，可以像调用函数一样调用该方法，模拟函数的行为，打印出书名及作者

### 2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

1. [root@localhost day07] # vim books.py
2. #!/usr/bin/env python3
3. class Book:
4. def \_\_init\_\_(self, title, author):
5. self.title = title
6. self.author = author
7. def \_\_str\_\_(self):
8. return '<Book: %s>' % self.title
9. def \_\_call\_\_(self):
10. print('《%s》 is written by %s.' % (self.title, self.author))
11. if \_\_name\_\_ == '\_\_main\_\_':
12. pybook = Book('Core Python', 'Weysley')
13. print(pybook) # 调用\_\_str\_\_
14. pybook() # 调用\_\_call\_\_

步骤二：测试脚本执行

1. [root@localhost day07]# python3 books.py
2. <Book: Core Python>
3. 《Core Python》 is written by Weysley.

## 3 案例4：分析apache访问日志

### 3.1 问题

编写count\_patt.py脚本，实现一个apche日志分析脚本：

1. 统计每个客户端访问apache服务器的次数
2. 将统计信息通过字典的方式显示出来
3. 分别统计客户端是Firefox和MSIE的访问次数
4. 分别使用函数式编程和面向对象编程的方式实现

### 3.2 方案

collections是python内建的一个集合模块，模块中提供了许多有用的集合类,其中counter类 是一个简单的计数器，以字典的键值对形式储存，其中搜索的元素作为键，出现的次数作为值

实现过程：

1.实例化一个计数器

2.实例化正则表达式

3.将文件以对象形式打开

4.通过正则表达式查找文件每一行

5.如果找到结果

6.将结果添加到计数器，通过update方法更新原有数据

7.返回计数器

8.将文件地址和正则表达式作为实参传递给函数

### 3.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

1. [root@localhost day08]# vim count\_patt.py
2. #!/usr/bin/env python3
3. import re
4. import collections
5. #fname 文件地址 patt 正则表达式
6. def count\_patt(fname,patt):
8. counter = collections.Counter()
10. cpatt = re.compile(patt)
11. with open(fname) as fobj:
12. for line in fobj:
14. m = cpatt.search(line)
16. if m:
18. counter.update([m.group()])
20. return counter
21. if \_\_name\_\_ == "\_\_main\_\_":
22. fname = "access\_log.txt"
23. ip\_patt = "^(\d+\.){3}\d+"
24. a = count\_patt(fname,ip\_patt)
25. print(a)
26. br\_patt = "Firefox|MSIE|Chrome"
27. b = count\_patt(fname,br\_patt)
28. print(b)

实现此案例还可通过面向对象方式实现：

实现过程：

1.创建类CountPatt

2.定义构造方法 创建正则对象

3.定义类方法

4.创建计数器对象

5.打开文本文件

6.通过正则表达式查找文件每一行

7.如果找到结果

8.将结果添加到计数器，通过update方法更新原有数据

9.返回计数器

10.将文件地址和正则表达式作为实参传递给函数

1. [root@localhost day08]# vim count\_patt2.py
2. #!/usr/bin/env python3
3. import re
4. import collections
5. import re
6. import collections
7. class CountPatt:
9. def \_\_init\_\_(self,patt):
10. self.cpatt = re.compile(patt)
12. def count\_patt(self,fname):
14. counter = collections.Counter()
16. with open(fname) as fobj:
18. for line in fobj:
20. m = self.cpatt.search(line)
22. if m:
24. counter.update([m.group()])
26. return counter
27. if \_\_name\_\_ == "\_\_main\_\_":
28. fname = "access\_log.txt"
29. ip\_patt = "^(\d+\.){3}\d+"
30. br\_patt = "Firefox|MSIE|Chrome"
31. ip = CountPatt(ip\_patt)
32. print(ip.count\_patt(fname))
33. br = CountPatt(br\_patt)
34. print(br.count\_patt(fname))

步骤二：测试脚本执行

1. [root@localhost day08]# python3 count\_patt.py
2. Counter({‘172.40.0.54’：391，‘172.40.50.116’：244，‘201.1.1.254’：173，‘127.0.0.1’：121，‘201.1.2.254’：119})
3. Counter({‘Firefox’：870，‘MSIE’：391，‘Chrome’：24})
4. [root@localhost day08]# python3 count\_patt2.py
5. Counter({‘172.40.0.54’：391，‘172.40.50.116’：244，‘201.1.1.254’：173，‘127.0.0.1’：121，‘201.1.2.254’：119})
6. Counter({‘Firefox’：870，‘MSIE’：391，‘Chrome’：24})