[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY04/CASE/01/index.html" \l "page_top_case)

# NSD Python1 DAY04

1. [案例1：创建文件](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY04/CASE/01/index.html" \l "case1)
2. [案例2：创建用户](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY04/CASE/01/index.html" \l "case2)
3. [案例3：格式化输出](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON1/DAY04/CASE/01/index.html" \l "case3)

## 1 案例1：创建文件

### 1.1 问题

编写mktxtfile.py脚本，实现以下目标：

1. 编写一个程序，要求用户输入文件名
2. 如果文件已存在，要求用户重新输入
3. 提示用户输入数据，每行数据先写到列表中
4. 将列表数据写入到用户输入的文件名中

### 1.2 方案

用三个函数分别实现文件名获取、文件内容获取、将获取到的文件内容写入get\_fname()函数获取的文件中 这三个方法，最终调用三个函数，完成文件创建：

1.获取文件名函数get\_fname()：利用while语句循环判断文件名是否存在，input文件名，如果不存在，循环停止，返回用户输入的文件名，如果存在，提示已存在，重新进入循环，直至文件名不存在为止，返回文件名用户输入的文件名

2.文件内容获取函数get\_contents()：创建空列表存储获取到的数据，利用while语句让用户循环输入数据，如果输入的数据是end，循环停止，返回列表中内容，如果输入的数据不是end，将输入的数据追加到列表结尾，返回列表中内容

3.wfile()函数：用with语句将获取到的文件以写方式打开，这样打开代码块结束后文件会自动关闭，将get\_contents()函数返回内容写入到已打开文件中

4.最终当用户cat文件名时，可以看到写入结果

### 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 day04]# vim mktxtfile.py
2. #!/usr/bin/env python3
3. import os
4. def get\_fname():
5. while True:
6. filename = input('请输入文件名：')
7. if not os.path.exists(filename):
8. break
9. print('%s 已存在，请重试。' % filename)
10. return filename
11. def get\_contents():
12. contents = []
13. print('请输入内容，结束请输入end。')
14. while True:
15. line = input('> ')
16. if line == 'end':
17. break
18. contents.append(line)
19. return contents
20. def wfile(fname, contents):
21. with open(fname, 'w') as fobj:
22. fobj.writelines(contents)
23. if \_\_name\_\_ == '\_\_main\_\_':
24. fname = get\_fname()
25. contents = get\_contents()
26. contents = ['%s\n' % line for line in contents]
27. wfile(fname, contents)

步骤二：测试脚本执行

1. [root@localhost day04]# ls
2. adduser.py format\_str2.py list\_method.py mylist.py string\_op.py
3. checkid.py format\_str.py mkseq.py randpass2.py
4. fmtoutput.py get\_val.py mktxtfile.py seq\_func.py
5. [root@localhost day04]# python3 mktxtfile.py
6. 请输入文件名：passwd
7. 请输入内容，结束请输入end。
8. > nihao,welcom
9. > woshi
10. > end
11. [root@localhost day04]# python3 mktxtfile.py
12. 请输入文件名：mkseq.py
13. mkseq.py 已存在，请重试。
14. 请输入文件名：randpass.py
15. 请输入内容，结束请输入end。
16. > myname
17. > end
18. [root@localhost day04]# cat passwd
19. nihao,welcom
20. woshi
21. [root@localhost day04]# cat randpass.py
22. myname
23. [root@localhost day04]# ls
24. adduser.py format\_str2.py list\_method.py mylist.py randpass.py
25. checkid.py format\_str.py mkseq.py passwd seq\_func.py
26. fmtoutput.py get\_val.py mktxtfile.py randpass2.py string\_op.py

## 2 案例2：创建用户

### 2.1 问题

创建adduser.py文件，实现以下目标：

1. 编写一个程序，实现创建用户的功能
2. 提示用户输入用户名
3. 随机生成8位密码
4. 创建用户并设置密码
5. 将用户相关信息写入指定文件

### 2.2 方案

创建add\_user()函数，让函数具有创建用户、创建密码、将用户密码写入到指定文件三种方法，因此为函数设置3个参数，分别是用户名、密码及用户名密码存放文件，最终通过函数调用上传实参的方式，完成用户创建

1.利用subprocess.call函数运行用户创建命令

2.subprocess.call函数运行密码设置命令

3.用with语句将指定的文件以追加模式打开，这样打开代码块结束后文件会自动关闭，将用户密码用指定格式写入指定文件

4.调用add\_user()函数时上传的用户名实参，是利用sys.argv[]参数，在命令行调用的时候由系统传递给程序，这个变量其实是一个List列表，用于保存命令行上的参数，argv[0] 一般是“被调用的脚本文件名或全路径”，argv[1]和以后就是传入的系统命令参数

### 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

将randpass文件的代码以模块形式导入以下代码中,直接调用gen\_pass()函数获取返回值（即获取随机生成的密码）：

1. [root@localhost day04] # vim adduser.py
2. #!/usr/bin/env python3
3. import sys
4. import subprocess
5. from randpass import gen\_pass
6. def add\_user(username, password, fname):
7. info = """user information:
8. username: %s
9. password: %s
10. """
11. subprocess.run('useradd %s' % username, shell=True)
12. subprocess.run(
13. 'echo %s | passwd --stdin %s' % (password, username),
14. shell=True
15. )
16. with open(fname, 'a') as fobj:
17. fobj.write(info % (username, password))
18. if \_\_name\_\_ == '\_\_main\_\_':
19. username = sys.argv[1]
20. password = gen\_pass()
21. fname = '/tmp/users.txt'
22. add\_user(username, password, fname)

步骤二：测试脚本执行

1. [root@localhost day04]# python3 adduser.py b c d
2. 更改用户 b 的密码 。
3. passwd：所有的身份验证令牌已经成功更新。
4. [root@localhost day04]# python3 adduser.py a c d
5. useradd：用户“a”已存在
6. 更改用户 a 的密码 。
7. passwd：所有的身份验证令牌已经成功更新。
8. [root@localhost day04]# cat /tmp/users.txt
9. user information:
10. username: a
11. password: hD31SmTS
12. user information:
13. username: b
14. password: DztS7ycn
15. user information:
16. username: a
17. password: f2iH0Znt

## 3 案例3：格式化输出

### 3.1 问题

创建fmtoutput.py脚本，要求如下：

1. 提示用户输入（多行）数据
2. 假定屏幕的宽度为50，用户输入的多行数据如图-1所示（文本内容居中）：



图-1

### 3.2 方案

利用for循环方式遍历获取到的用户输入数据列表，将用户输入的每一条数据依次遍历出来

通过format()方法，把遍历得到的字符串当作一个模版，通过传入的参数进行格式化。这个用来格式化的模版使用大括号({,})作为特殊字符，其中^代表居中对齐、48代表宽度。

### 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

将mktxtfile文件的代码以模块形式导入以下代码中,直接调用get\_contents ()函数获取返回值（即获取用户输入数据列表）：

1. [root@localhost day04]# vim fmtoutput.py
2. #!/usr/bin/env python3
3. from mktxtfile import get\_contents
4. width = 48
5. contents = get\_contents()
6. print('+%s+' % ('\*' \* 48))
7. for line in contents:
8. print('+{:^48}+'.format(line))
9. print('+%s+' % ('\*' \* 48))

步骤二：测试脚本执行

1. [root@localhost day04]# python3 fmtoutput.py
2. 请输入内容，结束请输入end。
3. > nihao
4. > my name zhangzhigang
5. > bye
6. > end
7. +\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*+
8. + nihao +
9. + my name zhangzhigang +
10. + bye +
11. +\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*+