[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON2/DAY01/CASE/01/index.html" \l "page_top_case)

# NSD Python2 DAY01

1. [案例1：简化除法判断](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON2/DAY01/CASE/01/index.html" \l "case1)
2. [案例2：自定义异常](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON2/DAY01/CASE/01/index.html" \l "case3)
3. [案例3：操作文件系统](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON2/DAY01/CASE/01/index.html" \l "case5)
4. [案例4：记账程序](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/PYTHON2/DAY01/CASE/01/index.html" \l "case7)

## 1 案例1：简化除法判断

### 1.1 问题

创建mydiv.py脚本，要求如下：

1. 提示用户输入一个数字作为除数
2. 如果用户按下Ctrl+C或Ctrl+D则退出程序
3. 如果用户输入非数字字符，提示用户应该输入数字
4. 如果用户输入0，提示用户0不能作为除数

### 1.2 方案

首先，执行try子句（在关键字try和关键字except之间的语句），输入数字，让这个数字被100整除，

1.如果没有异常发生，忽略except子句，try子句执行后，执行else子句和finally子句，最后执行 try 语句之后的代码结束整个程序。

2.如果在执行try子句的过程中发生了异常，异常的类型和 except 之后的名称相符，那么对应的except子句将被执行。然后执行finally子句，最后执行 try 语句之后的代码结束整个程序。

需要注意的是：允许用户中断这个程序（使用Ctrl+C或Ctrl+D方法）。用户中断的信息会引发KeyboardInterrupt 和EOFError 这两种异常。

一个 try 语句可能包含多个except子句，分别来处理不同的特定的异常。最多只有一个分支会被执行。

处理程序将只针对对应的try子句中的异常进行处理，而不是其他的 try 的处理程序中的异常。

try except 语句只有一个可选的else子句，使用这个子句，必须放在所有的except子句之后。这个子句将在try子句没有发生任何异常的时候执行。

finally子句是无论异常是否发生，是否捕捉都会执行的一段代码，使用finally可以保证文件总是能正常的关闭

### 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 day05]# vim mydiv.py
2. #!/usr/bin/env python3
3. try:
4. num = int(input("number: "))
5. result = 100 / num
6. except ValueError:
7. print('请输入数字')
8. except ZeroDivisionError:
9. print('不允许使用0')
10. except (KeyboardInterrupt, EOFError):
11. print('\nBye-bye')
12. else:
13. print(result) # 不发生异常才执行的语句
14. finally:
15. print('Done') # 不管异常是否发生都要执行的语句
16. print('end of program')
17. #不是必须把所有的语句写全，常用的有try-except和try-finally组合

步骤二：测试脚本执行

1. [root@localhost day05]# python3 mydiv.py
2. number: 0
3. 不允许使用0
4. Done
5. end of program
6. [root@localhost day05]# python3 mydiv.py
7. number: nighao
8. 请输入数字
9. Done
10. end of program
11. [root@localhost day05]# python3 mydiv.py
12. number: 3
13. 33.333333333333336
14. Done
15. end of program
16. [root@localhost day05]# python3 mydiv.py
17. number: 55^C
18. Bye-bye
19. Done
20. end of program
21. [root@localhost day05]# python3 mydiv.py
22. number:
23. Bye-bye
24. Done
25. end of program

## 2

## 3 案例2：自定义异常

### 3.1 问题

创建myerror.py脚本，要求如下：

1. 编写第一个函数，函数接收姓名和年龄，如果年龄不在1到120之间，产生ValueError异常
2. 编写第二个函数，函数接收姓名和年龄，如果年龄不在1到120之间，产生断言异常

### 3.2 方案

两个函数，分别有引发异常及断言异常的功能：

1.当set\_age()函数调用名字与年龄两个实参时，如果年龄在0-120范围内，打印“bob is 25 years old”，如果年龄在0-120范围外，利用raise 语句抛出一个指定的异常

2.当set\_age2()函数调用名字与年龄两个实参时，如果年龄在0-120范围内，表达式为true，打印“bob is 20 years old”，如果年龄在0-120范围外，表达式为Flase，利用assert 断言语句抛出一个指定的异常

### 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 day05]# vim myerror.py
2. #!/usr/bin/env python3
3. def set\_age(name, age):
4. if not 0 < age < 120:
5. raise ValueError("age out of range.")
6. print("%s is %s years old" % (name, age))
7. def set\_age2(name, age):
8. assert 0 < age < 120, 'age out of range.'
9. print("%s is %s years old" % (name, age))
10. if \_\_name\_\_ == '\_\_main\_\_':
11. set\_age('bob', 25)
12. set\_age2('bob', 20)

步骤二：测试脚本执行

1. [root@localhost day05]# python3 myerror.py
2. bob is 25 years old
3. bob is 20 years old
4. [root@localhost day05]# python3 myerror.py
5. Traceback (most recent call last):
6. File "myerror.py", line 11, in <module>
7. set\_age('bob', 125)
8. File "myerror.py", line 3, in set\_age
9. raise ValueError("age out of range.")
10. ValueError: age out of range.
11. [root@localhost day05]# python3 myerror.py
12. bob is 25 years old
13. Traceback (most recent call last):
14. File "myerror.py", line 12, in <module>
15. set\_age2('bob', 120)
16. File "myerror.py", line 7, in set\_age2
17. assert 0 < age < 100, 'age out of range.'
18. AssertionError: age out of range.

## 4

## 5 案例3：操作文件系统

### 5.1 问题

创建os\_module.py脚本，熟悉os模块操作,要求如下：

1. 切换到/tmp目录
2. 创建example目录
3. 切换到/tmp/example目录
4. 创建test文件，并写入字符串foo bar
5. 列出/tmp/exaple目录内容
6. 打印test文件内容
7. 反向操作，把test文件以及example目录删除

### 5.2 方案

用os方法查看用户当前所在位置，切换到指定目录，创建example目录，切换到创建目录下，以读写方式打开并创建一个新文件，将指定内容写入新文件中，列出目录下有指定目录下有哪些文件，指定从开始位置读取指定文件字符串并打印出来，关闭打开文件，并删除文件，删除目录。

注意：读取打印文件内容时，要将字节转化为字符串读取出来。

### 5.3 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 day05]# vim os\_module.py
2. #!/usr/bin/env python3
3. import os
4. #1)    切换到/tmp目录
5. os.getcwd()        #'/root/python代码/os'
6. os.chdir("/tmp")
7. os.getcwd()        #'/tmp'
8. #2)    创建example目录
9. os.mkdir("example")
10. #3)    切换到/tmp/example目录
11. os.chdir("/tmp/example")
12. os.getcwd()        #'/tmp/example'
13. #4)    创建test文件，并写入字符串foo bar
14. f=os.open("test.txt",os.O\_RDWR|os.O\_CREAT)        #以读写方式打开/创建并打开一个新文件
15. os.write(f,b"foo bar nihao")
16. #5)    列出/tmp/exaple目录内容
17. os.listdir("/tmp/example")     #['test.txt']
18. #6)    打印test文件内容
19. os.lseek(f,0,0)        #指定从开始位置读取字符串
20. str=os.read(f,100)
21. str = bytes.decode(str)
22. print("读取的字符是：",str)
23. os.close(f)
24. #7)    反向操作，把test文件以及example目录删除
25. os.remove("/tmp/example/test.txt")
26. os.removedirs("/tmp/example")

步骤二：测试脚本执行

1. [root@localhost day05]# python3 os\_module.py
2. 读取的字符是： foo bar

## 6

## 7 案例4：记账程序

### 7.1 问题

创建account.py脚本，要求如下：

1. 假设在记账时，有一万元钱
2. 无论是开销还是收入都要进行记账
3. 记账内容包括时间、金额和说明等
4. 记账数据要求永久存储

### 7.2 方案

创建4个函数，分别实现记录开销、记录收入、查询收支、判断函数调用的四个方法，导入时间模块获取时间，导入os模块判断文件是否存在，导入pickle模块用来python特有类型与数据类型转换：

1.调用show\_menu()函数后，先判断记录余额文件是否存在，如果不存在创建文件并写入余额，如果存在，利用while循环在交互端输出提示，请用户input0/1/2/3任意数值，如果输入的值不是0/1/2/3，打印输入值无效请重新输入并重新开始循环，如果输入的值是3，停止整个循环，如果输入的值是0/1/2通过字典键值对关联关系，调用相对应函数

2.如果输入的值是0，字典cmds中0键对应的值是spend\_money，调用spend\_money ()记录开销函数，让此函数实现获取当前系统日期、输入开销金额、输入开销备注信息、以二进制读方式打开记录余额文件计算本次开销后余额，以写方式打开记录余额文件将计算后开销余额写入文件，以追加方式打开记账文件，将日期、开销、备注、余额写入追加入记账文件最后

3.如果输入的值是1，字典cmds中0键对应的值是save\_money，调用save\_money ()记录收入函数，让此函数实现获取当前系统日期、输入收入金额、输入收入备注信息、以二进制读方式打开记录余额文件计算本次收入后余额，以写方式打开记录余额文件将计算后收入余额写入文件，以追加方式打开记账文件，将日期、开销、备注、余额写入追加入记账文件最后

4.如果输入的值是2，调用查询收支函数query ()，以二进制读方式打开记账文件，利用for循环遍历文件中数据，打印出来，打开记录余额文件读取余额并打印。

需要注意的是：为确保代码可以正常执行，while循环利用try except语句处理异常，优先匹配特殊异常，让用户按下Ctrl+C或Ctrl+D可以退出程序，遇到索引错误可以结束当次循环，重新开始选择选项。

将记录余额文件以及记账文件作为参数传入函数中

### 7.3 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 day05]# vim account.py
2. #!/usr/bin/env python3
3. # 日期　　开销　　收入　　余额　　备注
4. import time
5. import os
6. import pickle as p
7. def spend\_money(record, wallet):
8. date = time.strftime('%Y-%m-%d')
9. amount = int(input('金额: '))
10. comment = input('备注: ')
11. with open(wallet, 'rb') as fobj:
12. #load从数据文件中读取数据，并转换为Python的数据结构
13. balance = p.load(fobj) – amount
14. with open(wallet, 'wb') as fobj:
15. #dump将数据通过特殊形式转换为只有python语言认识的字符串，并写入文件
16. p.dump(balance, fobj)
17. with open(record, 'a') as fobj:
18. fobj.write(
19. "%-15s%-8s%-8s%-10s%-20s\n" %
20. (date, amount, 'n/a', balance, comment)
21. )
22. def save\_money(record, wallet):
23. date = time.strftime('%Y-%m-%d')
24. amount = int(input('金额: '))
25. comment = input('备注: ')
26. with open(wallet, 'rb') as fobj:
27. balance = p.load(fobj) + amount
28. with open(wallet, 'wb') as fobj:
29. p.dump(balance, fobj)
30. with open(record, 'a') as fobj:
31. fobj.write(
32. "%-15s%-8s%-8s%-10s%-20s\n" %
33. (date, 'n/a', amount, balance, comment)
34. )
35. def query(record, wallet):
36. with open(record) as fobj:
37. for line in fobj:
38. print(line, end='')
39. with open(wallet, 'rb') as fobj:
40. #load从数据文件中读取数据，并转换为Python的数据结构
41. balance = p.load(fobj)
42. print('当前余额: %s' % balance)
43. def show\_menu():
44. prompt = """(0) 记录开销
45. (1) 记录收入
46. (2) 查询收支记录
47. (3) 退出
48. 请选择(0/1/2/3): """
49. cmds = {'0': spend\_money, '1': save\_money, '2': query}
50. record = 'record.txt' # 记帐
51. wallet = 'wallet.data' # 记录余额
52. if not os.path.exists(wallet):        #判断文件是否存在
53. with open(wallet, 'wb') as fobj:
54. p.dump(10000, fobj)
55. while True:
56. try:
57. choice = input(prompt).strip()[0]
58. except IndexError:
59. continue
60. except (KeyboardInterrupt, EOFError):
61. print('\nBye-bye')
62. choice = '3'
63. if choice not in '0123':
64. print('无效输入，请重试')
65. continue
66. if choice == '3':
67. break
68. cmds[choice](record, wallet)
69. if \_\_name\_\_ == '\_\_main\_\_':
70. show\_menu()

步骤二：测试脚本执行

1. [root@localhost day05]# python3 account.py
2. (0) 记录开销
3. (1) 记录收入
4. (2) 查询收支记录
5. (3) 退出
6. 请选择(0/1/2/3): 0
7. 金额: 2000
8. 备注: huafei
9. (0) 记录开销
10. (1) 记录收入
11. (2) 查询收支记录
12. (3) 退出
13. 请选择(0/1/2/3): 1
14. 金额: 1000
15. 备注: shouru
16. (0) 记录开销
17. (1) 记录收入
18. (2) 查询收支记录
19. (3) 退出
20. 请选择(0/1/2/3): 2
21. 2018-04-25 2000 n/a 28890 huafei
22. 2018-04-25 n/a 1000 29890 shouru
23. 当前余额: 29890
24. (0) 记录开销
25. (1) 记录收入
26. (2) 查询收支记录
27. (3) 退出
28. 请选择(0/1/2/3): 3
29. [root@localhost day05]# python3 account.py
30. (0) 记录开销
31. (1) 记录收入
32. (2) 查询收支记录
33. (3) 退出
34. 请选择(0/1/2/3): ^C
35. Bye-bye
36. [root@localhost day05]# python3 account.py
37. (0) 记录开销
38. (1) 记录收入
39. (2) 查询收支记录
40. (3) 退出
41. 请选择(0/1/2/3):
42. Bye-bye