

第十五章异常处理

一 异常的介绍

1. 什么是异常
2. 异常由三分部构成
3. 异常的分类

二 异常处理

1. 异常处理的格式
2. 异常处理多分支
3. try 与 else 联用：
4. try 与 finally 组合：

三 主动触发异常

1. raise 主动触发异常
2. assert 断言
3. 自定义异常

本文是Python通用编程系列教程，已全部更新完成，实现的目标是从零基础开始到精通Python编程语言。本教程不是对Python的内容进行泛泛而谈，而是精细化，深入化的讲解，共5个阶段，25章内容。所以，需要有耐心的学习，才能真正有所收获。虽不涉及任何框架的使用，但是会对操作系统和网络通信进行全局的讲解，甚至会对一些开源模块和服务器进行重写。学完之后，你所收获的不仅仅是精通一门Python编程语言，而且具备快速学习其他编程语言的能力，无障碍阅读所有Python源码的能力和对计算机与网络的全面认识。对于零基础的小白来说，是入门计算机领域并精通一门编程语言的绝佳教材。对于有一定Python基础的童鞋，相信这套教程会让你的水平更上一层楼。

一 异常的介绍

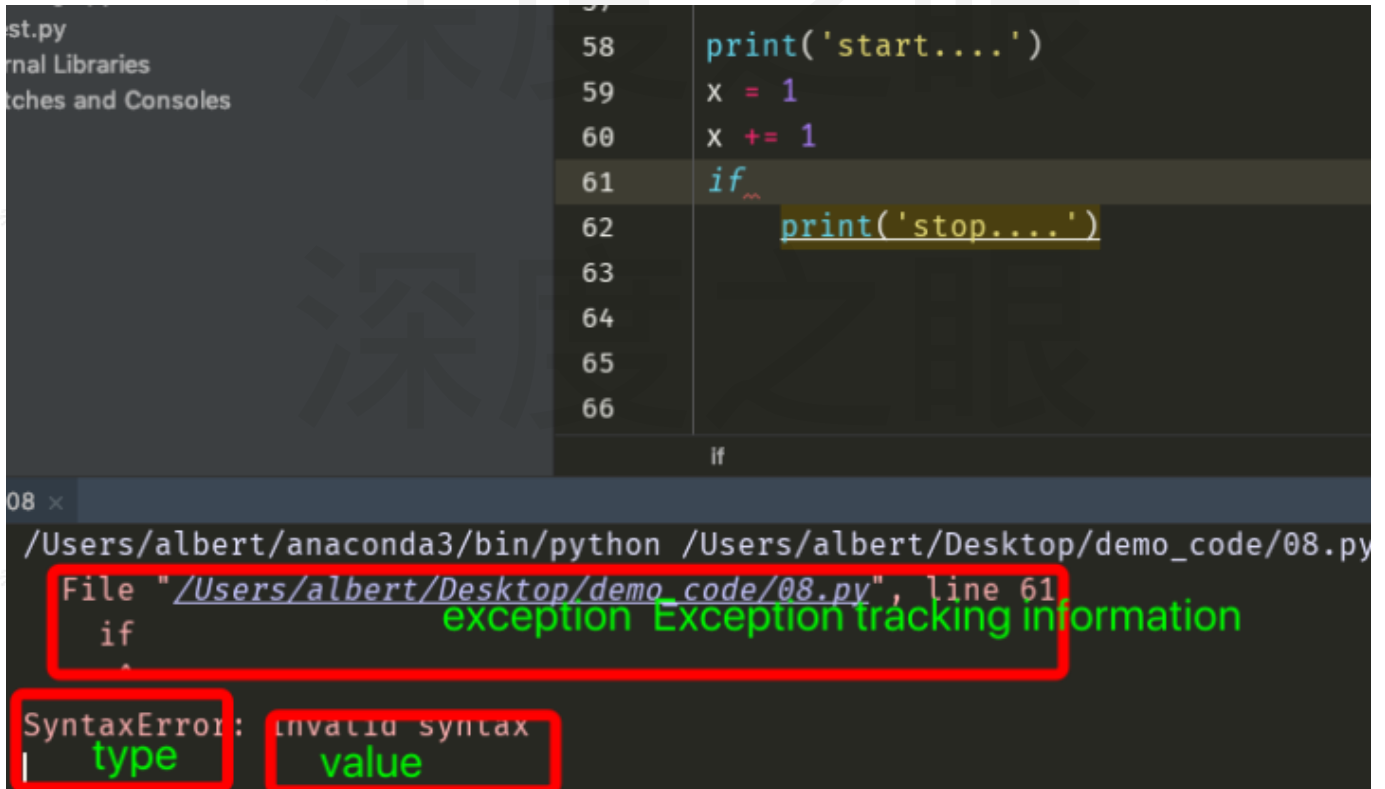
1. 什么是异常

异常处理也就是处理异常，那么我们首先要知道什么是异常？

异常是错误发生的信号，程序一旦出错，如果程序中还没有相应的处理机制，那么该错误就会产生一个异常抛出来，程序的运行也随之终止

2. 异常由三分部构成

1. 异常的追踪信息
2. 异常的类型
3. 异常的值



```
57
58 print('start....')
59 x = 1
60 x += 1
61 if
62     print('stop....')
63
64
65
66
```

File "/Users/albert/Desktop/demo_code/08.py", line 61
if
SyntaxError: invalid syntax
type: SyntaxError
value: invalid syntax

异常的类型就是异常的类，所以可以打印SyntaxError来查看

3. 异常的分类

语法异常

```

emo.py
dhfk.docx
ettings.py
est.py
ernal Libraries
tches and Consoles
56
57
58 print('start....')
59 x = 1
60 x += 1
61 print(x)
62 if
63     print('stop....')
64
65
66
67
08 x
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
File "/Users/albert/Desktop/demo_code/08.py", line 62
    if
    ^
SyntaxError: invalid syntax

```

这类异常是在程序运行之前，Python解释器会通篇监测语法，发现错误，直接抛错，所以你会发现程序根本就没有执行，这种异常我们应该在程序运行前就改正，这也是最基本的素质。

逻辑异常：

索引异常：

```

st.py
ernal Libraries
tches and Consoles
62
63
64 IndexError
65 l=['a','b']
66 l[100]
67
68
69
70
71
08 x
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 66, in <modul
    l[100]
IndexError: list index out of range

```

键值对的键异常：

作者：马小享

```

65
66 # KeyError
67 d = {'a': 1}
68 d['b']
69
70
71
72
73

```

08 x

```

/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 68, in <module>
    d['b']
KeyError: 'b'

```

属性异常：

```

73
74 # AttributeError:
75 class Foo:
76     pass
77
78 Foo.x
79
80 import os
81 os.aaa
82
83
84
85

```

08 x

```

/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 78, in <module>
    Foo.x
AttributeError: type object 'Foo' has no attribute 'x'

```

注意：

1. 程序上面的代码出现异常，程序随之终止，下面的代码不再运行

2. 一切皆对象，os 也是对象

```
b.txt
demo.py
dhfk.docx
ettings.py
est.py
ernal Libraries
atches and Consoles

74 # AttributeError:
75 class Foo:
76     pass
77
78 # Foo.x
79 |
80 import os
81 os.aaa
82
83
84
85
```

```
08 x
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 81, in <module>
    os.aaa
AttributeError: module 'os' has no attribute 'aaa'
```

注意下图的打印信息

```
st.py
nal Libraries
ches and Consoles

115 def func():
116     print(123)
117     import os
118     os.XXXX
119
120 func()
121
122
...
```

```
08 x
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
123
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 120, in <module>
    func()
  File "/Users/albert/Desktop/demo_code/08.py", line 118, in func
    os.XXXX
AttributeError: module 'os' has no attribute 'xxxx'
```

0 不能做被除数异常：

```
85  
86 # ZeroDivisionError  
87 1 / 0  
88  
89
```

```
08 x  
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py  
Traceback (most recent call last):  
  File "/Users/albert/Desktop/demo_code/08.py", line 87, in <module>  
    1 / 0  
ZeroDivisionError: division by zero  
Process finished with exit code 1
```

文件找不到异常：

```
89  
90 # FileNotFoundError  
91 f=open('a.txt','r',encoding='utf-8')  
92  
93  
94  
95
```

```
08 x  
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py  
Traceback (most recent call last):  
  File "/Users/albert/Desktop/demo_code/08.py", line 91, in <module>  
    f=open('a.txt','r',encoding='utf-8')  
FileNotFoundError: [Errno 2] No such file or directory: 'a.txt'
```

关闭的文件不能读写异常：

作者：马一特

深度之眼

作者：马一特

作者：马一特

作者：马一特

```

84
85 # FileNotFoundError
86 f = open('a.txt', 'w', encoding='utf-8')
87
88 # ValueError: I/O operation on closed file
89 f = open('a.txt', 'r', encoding='utf-8')
90 f.close()
91 f.readline()
92
93
94

```

```

08 x
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 91, in <module>
    f.readline()
ValueError: I/O operation on closed file.

```

不是整型格式的字符串转化为整型异常：

```

93
94 # ValueError: invalid literal for int() with base 10: 'aaaa'
95 int('aaaa')
96 int('1.2') # 这个也异常
97

```

```

08 x
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 102, in <module>
    int('aaaa')
ValueError: invalid literal for int() with base 10: 'aaaa'

```

整型不可迭代异常：

作者：马一特

作者：马一特

深度之眼

作者：马一特

作者：马一特

```
99 # TypeError
100 for i in 333:
101     pass
102
103
104
```

```
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 100, in <module>
    for i in 333:
TypeError: 'int' object is not iterable

Process finished with exit code 1
```

变量名未定义异常：

函数也有自己的数据类型，他其实也是变量

```
106
107 # NameError
108 x
109 func()
110
111
112
113
```

```
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 108, in <module>
    x
NameError: name 'x' is not defined
```

常用异常总结：

AttributeError 试图访问一个对象没有的树形，比如foo.x，但是foo没有属性x
IOError 输入/输出异常；基本上是无法打开文件
ImportError 无法引入模块或包；基本上是路径问题或名称错误
IndentationError 语法错误（的子类）；代码没有正确对齐
IndexError 下标索引超出序列边界，比如当x只有三个元素，却试图访问x[5]
KeyError 试图访问字典里不存在的键
KeyboardInterrupt Ctrl+C被按下
NameError 使用一个还未被赋予对象的变量
SyntaxError Python代码非法，代码不能编译（个人认为这是语法错误，写错了）

`TypeError` 传入对象类型与要求的不符合

`UnboundLocalError` 试图访问一个还未被设置的局部变量，基本上是由于另有一个同名的全局变量，导致你以为正在访问它

`ValueError` 传入一个调用者不期望的值，即使值的类型是正确的

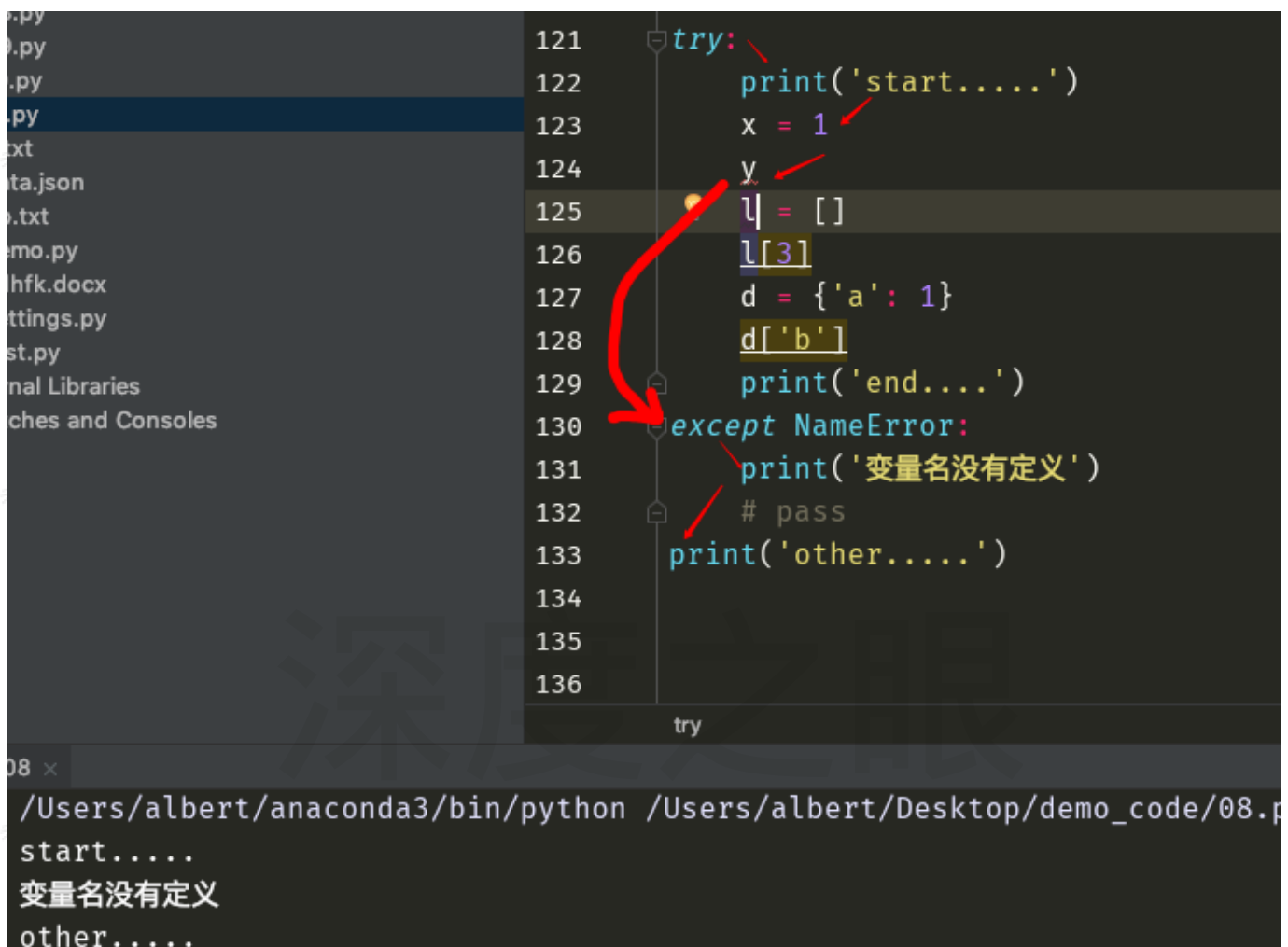
更多异常：

```
ArithmeticError
AssertionError
AttributeError
BaseException
BufferError
BytesWarning
DeprecationWarning
EnvironmentError
EOFError
Exception
FloatingPointError
FutureWarning
GeneratorExit
ImportError
ImportWarning
IndentationError
IndexError
IOError
KeyboardInterrupt
KeyError
LookupError
MemoryError
NameError
NotImplementedError
OSError
OverflowError
PendingDeprecationWarning
ReferenceError
RuntimeError
RuntimeWarning
StandardError
StopIteration
SyntaxError
SyntaxWarning
SystemError
SystemExit
```

```
TabError
TypeError
UnboundLocalError
UnicodeDecodeError
UnicodeEncodeError
UnicodeError
UnicodeTranslateError
UnicodeWarning
UserWarning
ValueError
Warning
ZeroDivisionError
```

二 异常处理

1. 异常处理的格式



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The code editor contains a try-except block. A red arrow points from the 'except' line to the output in the terminal. The output shows the execution of the code, including the error message '变量名没有定义' (Variable name not defined).

```
121 try:
122     print('start.....')
123     x = 1
124     y
125     l = []
126     l[3]
127     d = {'a': 1}
128     d['b']
129     print('end.....')
130 except NameError:
131     print('变量名没有定义')
132     # pass
133     print('other.....')
134
135
136
```

try

```
/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.p
start.....
变量名没有定义
other.....
```

try 加缩进的代码块，这部分代码是尝试捕获异常的代码，except 加可以预测到的异常种类。如果发生异常，并且补捕获到了异常的种类，就会执行 except 下面缩进的代码块。如果没有异常捕捉失败，那么异常还会发生，整个程序就终止了，所以 except 后面的 error 要和你抛出的异常相匹配才可以

```

119
120 # 单分支
121 try:
122     print('start.....')
123     x = 1
124     # y
125     l = []
126     l[3]
127     d = {'a': 1}
128     d['b']
129     print('end.....')
130 except NameError:
131     print('变量名没有定义')
132 # pass
133 print('other.....')
134
135 # 多分支
136 # try:
137 #     print('start.....')
138 except NameError
    
```

08 x

```

/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
start.....
Traceback (most recent call last):
  File "/Users/albert/Desktop/demo_code/08.py", line 126, in <module>
    l[3]
IndexError: list index out of range
    
```

2. 异常处理多分支

深度之眼

作者：马一特

作者：马一特

```

135 # 多分支
136 try:
137     print('start.....')
138     x = 1
139     # y
140     l = []
141     l[3]
142     d = {'a': 1}
143     d['b']
144     print('end.....')
145 except NameError:
146     print('变量名没有定义')
147 except KeyError:
148     print('字典的key不存在')
149 except IndexError:
150     print('索引超出列表的范围')
151
152 print('other.....')
153
try

```

08 x

```

/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.p
start.....
索引超出列表的范围
other.....

```

如果我们的需求是既要记录每一种异常的种类，要指明具体的异常信息，就是用这中方式处理，但是如果我们没有这么精确的需求，也可以使用同一种逻辑来处理。

作者：马一特

深度之眼

作者：马一特

作者：马一特

作者：马一特

```

154 # 多种异常采用同一段逻辑处理
155 try:
156     print('start.....')
157     x = 1
158     # y
159     l = []
160     # l[3]
161     d = {'a': 1}
162     d['b']
163     print('end.....')
164 except (NameError, KeyError, IndexError):
165     print('变量名或者字典的key或者列表的索引有问题')
166
167 print('other.....')

```

08 x /Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
start.....
变量名或者字典的key或者列表的索引有问题
other.....

如果我们不需要考虑异常发生的种类，只要代码平稳过渡，那么我们可以使用万能异常

```

169 # 万能异常
170 try:
171     print('start.....')
172     x = 1
173     # y
174     l = []
175     # l[3]
176     d = {'a': 1}
177     # d['b']
178     import os
179
180     os.aaa
181     print('end.....')
182 except Exception:
183     print('万能异常--->')
184
185 print('other.....')

```

8 x /Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
start.....
万能异常--->
other.....

虽然使用万能异常并不能清除的知道异常的类型，但是我们把异常的值赋值给一个变量名（上面所讲的其他异常也可以赋值），代码如下：

```
187 # 获取异常的值
188 try:
189     print('start.....')
190     x=1
191     y
192     l=[]
193     l[3]
194     d={'a':1}
195     d['b']
196     import os
197     os.aaa
198     print('end.....')
199 except Exception as e:
200     print('万能异常--->',e)
201
202
203 print('other.....')
204
```

try

08 x /Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
start.....
万能异常---> name 'y' is not defined
other.....

既要捕获指定的异常，又要确保程序平稳过渡


```

.py
.py
.py
.py
.py
.py
.py
.py
.py
xt
ta.json
.txt
mo.py
hfk.docx
tings.py
t.py
nal Libraries
ches and Consoles
232 try:
233     print('start.....')
234     # x=1
235     # # y
236     # l=[]
237     # l[3]
238     # d={'a':1}
239     # d['b']
240     # import os
241     # os.aaa
242     print('end....')
243 except NameError as e:
244     print('NameError: ', e)
245
246 except KeyError as e:
247     print('KeyError: ', e)
248
249 except Exception as e:
250     print('万能异常---》', e)
251
252 else:
253     print('在被检测的代码块没有出现任何异常的情况下执行')
254 print('other.....')

```

8 x

```

/Users/albert/anaconda3/bin/python /Users/albert/Desktop/demo_code/08.py
start.....
end....
在被检测的代码块没有出现任何异常的情况下执行
other.....

```

4. try 与 finally 组合：

无论有没有异常发生，finally 子代码都会执行

```

try:
    print('start.....')
    # x=1
    # # y
    # l=[]
    # l[3]
    # d={'a':1}
    # d['b']
    # import os
    # os.aaa
    print('end....')

```



```

except NameError as e:
    print('NameError: ',e)
except KeyError as e:
    print('KeyError: ',e)
except Exception as e:
    print('万能异常---》',e)
else:
    print('在被检测的代码块没有出现任何异常的情况下执行')
finally:
    print('无论有没有异常发生，都会执行')
print('other.....')

```

finally的子代码块中通常放回收系统资源的代码

```

try:
    f = open('a.txt', mode='w', encoding='utf-8')
    f.readline()
    # f.close()
finally:
    f.close()
print('other.....')

```

会报错，不要吃惊，文件打开后，没有关闭，Python内部的资源，会有自己回收机制去回收，但是程序只要抛错，如果没有 finally 后面的代码，文件是无法关闭的，这个系统资源就需要我们手动处理。

三 主动触发异常

1. raise 主动触发异常

上面讲到的是程序发生的我们不想让他发生的异常，我们已经知道怎么处理了，另外还有一种情况，我们也希望能够主动触发异常，来对使用者进行一定的限制。

```

print(TypeError) # TypeError是一个类
obj = TypeError("类型错误") # 这其实就是实例化对象的过程
print(obj)
class People:
    def __init__(self, name):
        if not isinstance(name, str):

```

```

        raise TypeError('用户名 %s 必须是str类型' % name)
    self.name = name
p = People(123)

```

2. assert 断言

试想一个场景，假如程序执行了很多代码到了第300行代码，而第300行代码得出的变量值是否为空，将关系到下面的代码是否运行，没有断言之前我们一般会这么处理

```

print('part1.....')
print('part1.....')
print('part1.....')
print('part1.....')
print('part1.....')
students = ['Albert', 'Harden', 'Curry', 'Wade']
students = []
if len(students) <= 0:
    raise TypeError # 这里也可以不传参数
print('part2.....')
print('part2.....')
print('part2.....')
print('part2.....')
print('part2.....')
print('part2.....')

```

使用断言：

```

print('part1.....')
print('part1.....')
print('part1.....')
print('part1.....')
print('part1.....')
students = ['Albert', 'Harden', 'Curry', 'Wade']
# students = []
# if len(students) <= 0:
#     raise TypeError # 这里也可以不传参数
assert len(students) > 0
print('part2.....')
print('part2.....')
print('part2.....')

```

```
print('part2.....')
print('part2.....')
print('part2.....')
```

3. 自定义异常

异常也就是一个类，那么我们当然可以自定义

```
class RegisterError(BaseException): # 需要继承这个异常的基类BaseException
    def __init__(self, message, username):
        self.message=message
        self.username=username
    def __str__(self): # 自定义对象的打印格式
        return '<%s:%s>' % (self.username, self.message)

print(RegisterError)
obj = RegisterError('注册失败', 'Albert')
print(obj)
# 这行代码就相当于先初始化对象，再打印这个对象
raise RegisterError('注册失败', 'Albert')
```