Debugging
Find and fix bugs

Ting ZHANG

Tools & Strategies

- Raise exceptions (√)
- Get the traceback string
- Assertion
- Logging
- Debugger(…)

The traceback info

```
def spam():
    bacon()


def bacon():
    raise Exception("This is the error message.")


spam()
```

The traceback info

```python
def spam():
    bacon()


def bacon():
    raise Exception("This is the error message.")


spam()
```

```
Traceback (most recent call last):
  File "C:/Users/Administrator/PycharmProjects/pythonProject/lab07.py", line 9, in <module>
    spam()
  File "C:/Users/Administrator/PycharmProjects/pythonProject/lab07.py", line 2, in spam
    bacon()
  File "C:/Users/Administrator/PycharmProjects/pythonProject/lab07.py", line 6, in bacon
    raise Exception("This is the error message.")
Exception: This is the error message.
```

The traceback info helps us to find where is the bug.

The traceback info

```
try:
    raise Exception("This is the error message.")
except:
    print('This exception has been handled elegantly! ')
```

```
This exception has been handled elegantly!
```

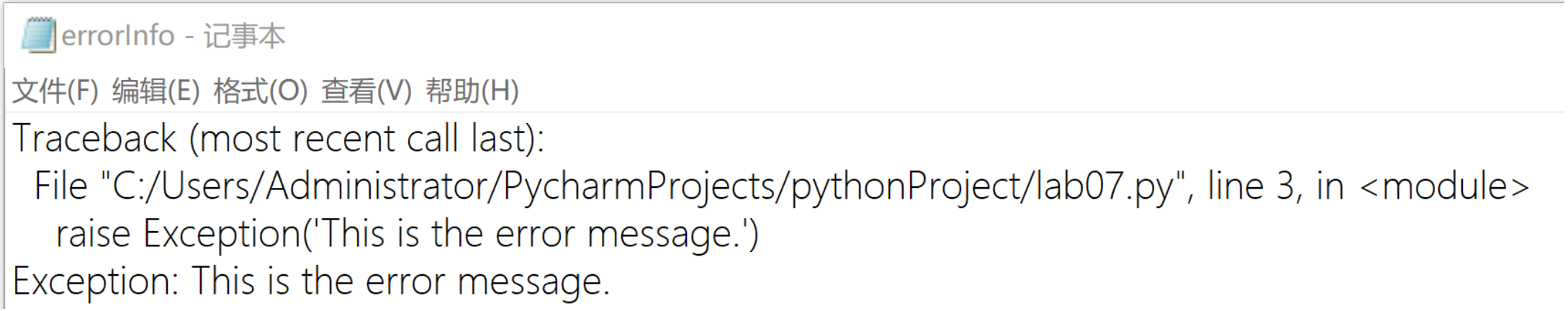
Exception handled elegantly but NO traceback info

Exception handled elegantly & The traceback info

```
import traceback
try:
    raise Exception('This is the error message.')
except:
    errorFile = open('errorInfo.txt', 'w')
    errorFile.write(traceback.format_exc())
    errorFile.close()
    print('This exception has been handled elegantly! ')
    print('The traceback info was written to errorInfo.txt.')
```

Exception handled elegantly & The traceback info

```
C:\Users\Administrator\PycharmProjects\pythonProject
This exception has been handled elegantly!
The traceback info was written to errorInfo.txt.
```


errorInfo - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

Traceback (most recent call last):
  File "C:/Users/Administrator/PycharmProjects/pythonProject/lab07.py", line 3, in <module>
    raise Exception('This is the error message.')
Exception: This is the error message.

Assertion

- It is typically used to check for conditions that cannot happen
- Assert statements cause termination

  an immediate stop → Fail Fast
- For errors from programmer not user

Syntax

assert condition, the output when condition is false

Assertion

```
ages = [28, 32, 15, 78, 37, 68, 96, 18, 72]
ages.sort()
assert ages[0] <= ages[-1], 'The first age should be <= tha last one'
```

Assertion

```
ages = [28, 32, 15, 78, 37, 68, 96, 18, 72]
ages.sort()
ages[0] = 100
assert ages[0] <= ages[-1], 'The first age should be <= tha last one'
```

```
Traceback (most recent call last):
  File "C:/Users/Administrator/PycharmProjects/pythonProject/lab07.py", line 15, in <module>
    assert ages[0] <= ages[-1], 'The first age should be <= tha last one'
AssertionError: The first age should be <= tha last one
```

Logging

- Use print() to show the value of some variable
  show messages to user

- Logging module displays the logging info on the screen/file

  show info to programmer

```python
import logging
# logging.disable(logging.DEBUG)
logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s - %(levelname)s - %(message)s')
logging.debug('Start of program')


def factorial(n):
    logging.debug('Start of factorial(%s)' % n)
    total = 1
    for i in range(1, n+1):
        total *= i
        logging.debug(f'i is {i}, total is {total}')
    logging.debug('End of factorial(%s)' % n)
    return total


print(factorial(5))
logging.debug('End of program')
```

```
2021-10-21 21:05:50,227 - DEBUG - Start of program
2021-10-21 21:05:50,227 - DEBUG - Start of factorial(5)
2021-10-21 21:05:50,227 - DEBUG - i is 1, total is 1
2021-10-21 21:05:50,228 - DEBUG - i is 2, total is 2
2021-10-21 21:05:50,228 - DEBUG - i is 3, total is 6
2021-10-21 21:05:50,228 - DEBUG - i is 4, total is 24
2021-10-21 21:05:50,228 - DEBUG - i is 5, total is 120
2021-10-21 21:05:50,228 - DEBUG - End of factorial(5)
2021-10-21 21:05:50,228 - DEBUG - End of program
120
```

- logging.disable(logging.DEBUG)

  disable all the logging messages of this level and below