初识异常：异常的产生与分类

目录

什么是异常

- 异常是 Python 解释器在执行程序的过程中，出现错误时的一种管理机制，它会在错误被监测到的位置“引发”
- 引发异常时，代码块会发生中断
- 异常是一种特殊对象

什么是异常

- 除 SystemExit 异常外，当异常没有被完全处理时，都会打印栈回溯信息

```
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>>
```

异常的分类

异常基类：

- BaseException —— 所有内置异常的基类
- Exception —— 所有内置的非系统退出类异常都派生自此类，所有用户自定义异常也应当派生自此类

异常的分类

异常基类：

- ArithmeticError —— 此基类用于派生针对各种算术类错误而引发的内置异常
- BufferError —— 当与缓冲区相关的操作无法执行时此基类将被引发
- LookupError —— 此基类用于派生当映射或序列所使用的键或索引无效时引发的异常

* https://docs.python.org/zh-cn/3.10/library/exceptions.html#base-classes

异常的分类

异常层次结构

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
      +-- ImportError
      |    +-- ModuleNotFoundError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalError
      +-- OSError
```

* https://docs.python.org/zh-cn/3.10/library/exceptions.html#exception-hierarchy

总结

1. 异常是错误自动引发，且无法被 Python 解释器自动处理的特殊对象

2. 异常有层次结构，如果不能明确需要哪些捕获，可以使用异常事件 Exception 捕获全部的异常

课后作业

请通过学习过的列表、字典、数字、字符串等数据类型，以及 if、for、while 等条件控制语句，自行编写程序， 引发至少 5 个异常并查看异常类型。

异常捕获：出现异常时，如何利用程序进行处理?

目录

try–except 代码块

try：

可能产生异常的代码

except 异常：

捕获指定的异常后运行的代码

else 代码块

try：

可能产生异常的代码

except 异常：

捕获指定的异常后运行的代码

else：

try 部分的代码没有抛出异常，执行此部分代码

finally 代码块

try：

可能产生异常的代码

except 异常：

捕获指定的异常后运行的代码

finally：

无论是否抛出异常，该部分代码均会执行

捕获多个异常

- try 语句块可以支持捕获多个异常：

```
try：
    可能产生异常的语句
except 异常 1：
    处理方式一
except 异常 2：
    处理方式二
except …
```

捕获多个异常

- 异常可以嵌套使用：

```
try:
    try:
        try:
            语句
        except:
            pass
except:
    pass
```

总结

1. try–except 语句块可以捕获异常
2. try–except 语句块可以支持 else 和 finally 语句实现复杂的异常处理
3. try–except 可以捕获多个异常，根据软件开发需要，可以使用多个 except 语句或语句嵌套方式实现多个异常的处理

课后作业

在文件操作过程中，使用 open() 函数打开文件时，有可能会出现该文件在磁盘中不存在的异常，请编写 try-except 语句，对文件打开的异常进行异常捕获。

* 当出现异常时，需关闭文件，无异常时正常读取文件。

自定义异常捕获：如何定义业务异常？

目录

自定义异常的用途

- 借用 Python 强大的异常处理功能，编写程序中断逻辑，继承现有的异常，并为现有的异常增加额外功能

自定义异常的方法

- 自定义异常需要继承异常的基类，一般通过继承 Exception 类实现：

```
class MyException(Exception)
```

自定义异常的方法

```
try:
    raise MyException
except MyException as e:
    print(e)
```

with 语句

- 为了简化异常处理逻辑，Python 引入了 with 语句，以打开文件为例：

```
with open("/path/to/file.txt") as f:

    txt = f.readlines()
```

with 语句

- with 语句使用了 __enter__() 和 __exit__() 两个方法实现，自定义 with 语句，可以使用如下写法：

```
class MyClass:
    def __enter__(self):
        with 语句后的对象会被调用，并将结果返回给 as 语句后的对象
    def __exit__(self):
        with 语句块所有代码执行完，执行此部分代码
```

with 语句

- 自定义类后，可以使用 with 语句调用 MyClass 类：

```
with MyClass() as mc:

    mc.xxx()
```

总结

1. 自定义异常必须继承 Exception 类
2. 自定义异常可以利用异常机制让程序中断
3. with 语句可以简化异常处理

课后作业

请编写自定义异常类，如果读取的文件小于 10 个字符，抛出文件过小的自定义异常，如果读取的文件大于等于 10 个字符，则正常输出该文件内容。

避坑指南：编写捕获异常程序时经常出现的问题

目录

捕获位置设置不当

- 只要在猎物出现的沿途设置陷阱，一定可以捕获猎物
- try–except 语句块设置过晚，程序已经报错退出
- 受到 if、while 等分支语句的影响，没有运行 try–except 语句块

设置范围不当

- 捕获异常的范围过大，导致抛出异常时，无法对定位问题产生有效帮助
- 抛出异常使用 Exception 基类，异常产生的原因不够明确

捕获处理设置不当

- 捕获异常后，处理异常应当明确告知用户异常原因
- 能够由程序自动处理的异常，应当处理妥善，如：关闭文件，以防止文件数据丢失

嵌套 try–except 语法错误

- try–except 语句块应成对出现
- 可以使用 IDE 和缩进来避免嵌套使用该语句块导致的语法错误

总结

1. 位置、范围、处理方式使用不当，是编写异常处理逻辑时的常见错误

2. 异常处理语句较长，一般可通过 VSCode 等 IDE 自动检查语法，避免出现语法错误

课后作业

程序员小派在调试程序时，发现程序偶尔会抛出自定义异常，但是根据异常提示无法定位异常产生的原因和产生异常的具体代码行数。作为小派的搭档，请你对该问题提出有效的代码优化方案。

THANKS