Python - Encadeamento de Exceções

O encadeamento de exceções é uma técnica de tratamento de exceções, lançando novamente uma exceção capturada após envolvê-la em uma nova exceção. A exceção original é salva como uma propriedade (como causa) da nova exceção.

Durante o tratamento de uma exceção 'A', é possível que ocorra outra exceção 'B'. É útil conhecer ambas as exceções para depurar o problema. Às vezes é útil para um manipulador de exceção recriar deliberadamente uma exceção, seja para fornecer informações extras ou para traduzir uma exceção para outro tipo.

No Python 3.x, é possível implementar o encadeamento de exceções. Se houver alguma exceção não tratada dentro de uma seção except, ela terá a exceção sendo tratada anexada a ela e incluída na mensagem de erro.

Exemplo

No trecho de código a seguir, tentar abrir um arquivo inexistente gera FileNotFoundError. É detectado pelo bloco except. Durante o tratamento, outra exceção é gerada.

```
try:
   open("nofile.txt")
except OSError:
   raise RuntimeError("unable to handle error")
```

Ele produzirá a seguinte **saída** -

```
Traceback (most recent call last):
  File "/home/cg/root/64afcad39c651/main.py", line 2, in <module>
open("nofile.txt")
FileNotFoundError: [Errno 2] No such file or directory: 'nofile.txt'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/cg/root/64afcad39c651/main.py", line 4, in <module>
    raise RuntimeError("unable to handle error")
RuntimeError: unable to handle error
```

elevação . . de

Se você usar uma cláusula from opcional na instrução raise, isso indica que uma exceção é uma consequência direta de outra. Isso pode ser útil ao transformar exceções. O token após a palavra-chave from deve ser o objeto de exceção.

```
try:
   open("nofile.txt")
except OSError as exc:
   raise RuntimeError from exc
```

Ele produzirá a seguinte **saída** -

```
Traceback (most recent call last):
  File "/home/cg/root/64afcad39c651/main.py", line 2, in <module>
    open("nofile.txt")
FileNotFoundError: [Errno 2] No such file or directory: 'nofile.txt'

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/home/cg/root/64afcad39c651/main.py", line 4, in <module>
    raise RuntimeError from exc
RuntimeError
```

elevação . . de Nenhum

Se usarmos None na cláusula from em vez do objeto de exceção, o encadeamento automático de exceções encontrado no exemplo anterior será desabilitado.

```
try:
   open("nofile.txt")
except OSError as exc:
   raise RuntimeError from None
```

Ele produzirá a seguinte **saída** -

```
Traceback (most recent call last):
 File "C:\Python311\hello.py", line 4, in <module>
  raise RuntimeError from None
RuntimeError
```

__context__ e __causa__

Levantar uma exceção no bloco except adicionará automaticamente a exceção capturada ao atributo __context__ da nova exceção. Da mesma forma, você também pode adicionar __cause__ a qualquer exceção usando a expressão **raise ... from** sintaxe.

```
try:
   try:
      raise ValueError("ValueError")
   except ValueError as e1:
      raise TypeError("TypeError") from e1
except TypeError as e2:
   print("The exception was", repr(e2))
   print("Its __context__ was", repr(e2.__context__))
   print("Its __cause__ was", repr(e2.__cause__))
```

Ele produzirá a seguinte **saída** -

```
The exception was TypeError('TypeError')
Its __context__ was ValueError('ValueError')
Its __cause__ was ValueError('ValueError')
```