Python raise from

If this Python Tutorial saves you hours of work, please whitelist it in your ad blocker 😭 and

Donate Now (https://www.pythontutorial.net/donation/)

to help us ❤️ pay for the web hosting fee and CDN to keep the website running.

Summary: in this tutorial, you will learn how to use the Python raise from statement to raise an exception with extra information.

Introduction to the Python raise from statement

The `raise from` statement has the following syntax:

```
raise <ExceptionType> from <cause>
```

Technically, it's equivalent to the following:

```
ex = ExceptionType
ex.__cause__ = cause
raise ex
```

By default, the `__cause__` attribute on exception (https://www.pythontutorial.net/python-oop/python-exceptions/) objects is always initialized to `None` (https://www.pythontutorial.net/advanced-python/python-none/) .

Python raise from statement example

The following `divide()` function divides a number by another and returns the result of the division:

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError as ex:
        raise ValueError('b must not be zero')
```

The `divide()` function has an exception handler that catches the `ZeroDivisionError` exception. Inside the handler, we raise a new `ValueError` exception.

If you pass zero to the second argument of the `divide()` function:

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError as ex:
        raise ValueError('b must not be zero') from ex


divide(10, 0)
```

you'll get the following stack trace:

```
Traceback (most recent call last):
  File "c:/python/app.py", line 3, in divide
    return a / b
ZeroDivisionError: division by zero

During handling of the above exception, another exception occurred:
```

```
Traceback (most recent call last):
  File "c:/python/app.py", line 8, in <module>
    divide(10, 0)
  File "c:/python/app.py", line 5, in divide
    raise ValueError('b must not be zero')
ValueError: b must not be zero
```

The import message is:

```
During handling of the above exception, another exception occurred:
```

It means that while you were handling the `ZeroDivisionError` exception, the `ValueError` exception occurred.

To instruct Python that you want to modify and forward the `ZeroDivisionError` exception, you can use the `raise from` statement like this:

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError as ex:
        raise ValueError('b must not be zero') from ex

divide(10, 0)
```

When you run the code, you'll get the following stack trace:

```
Traceback (most recent call last):
  File "c:/python/app.py", line 3, in divide
    return a / b
ZeroDivisionError: division by zero

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

```
Traceback (most recent call last):
  File "c:/python/app.py", line 8, in <module>
    divide(10, 0)
  File "c:/python/app.py", line 5, in divide
    raise ValueError('b must not be zero') from ex
ValueError: b must not be zero
```

Now, you receive the `ValueError` exception with a cause added to the __cause__ attribute of the exception object.

The following modifies the above code to show the `__cause__` attribute of the `ValueError` exception:

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError as ex:
        raise ValueError('b must not be zero') from ex


try:
    divide(10, 0)
except ValueError as ex:
    print('cause:', ex.__cause__)
    print('exception:', ex)
```

Output:

```
cause: division by zero
exception: b must not be zero
```

Python raise exception from None

If the cause of the exception is not important, you can omit the cause by using the `raise exception from None` statement:

```
raise <ExceptionType> from None
```

For example, you can hide the cause of the `ValueError` exception in the `divide()` function as follows:

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        raise ValueError('b must not be zero') from None


try:
    divide(10, 0)
except ValueError as ex:
    print('cause:', ex.__cause__)
    print('exception:', ex)
```

Output:

```
cause: None
exception: b must not be zero
```

Now, the `__cause__` is `None`. Also, the `divide()` function raises the `ValueError` exception without any additional information.

If you remove the `try` statement in the code that calls the `divide()` function:

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        raise ValueError('b must not be zero') from None



divide(10, 0)
```

you’ll get the following stack trace:

```
Traceback (most recent call last):
  File "c:/python/app.py", line 8, in <module>
    divide(10, 0)
  File "c:/python/app.py", line 5, in divide
    raise ValueError('b must not be zero') from None
ValueError: b must not be zero
```

Summary

- Use the Python `raise from` statement to modify and forward an existing exception.
- Use the `raise exception from None` statment to hide the cause of the exception.