Exceptions

May 7, 2021

1 Exceptions

Exceptions contain a payload `args` that must be a single string containing the error message. It can be retrieve by using the readonly named attribute `args` or transforming the exception into `str`.

[2]:
```
def access_third(sequence):
    return sequence[3]

sequence = [1, 2]
access_third(sequence)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-2-ce11936d1c63> in <module>
      3
      4 sequence = [1, 2]
----> 5 access_third(sequence)

<ipython-input-2-ce11936d1c63> in access_third(sequence)
      1 def access_third(sequence):
----> 2     return sequence[3]
      3
      4 sequence = [1, 2]
      5 access_third(sequence)

IndexError: list index out of range
```

[6]:
```
try:
    access_third(sequence)
except LookupError as e:
    print(e)
    print(e.args)
    print(e.args[0])
```

```
list index out of range
('list index out of range',)
list index out of range
```

1.0.1 Named attributes

Some exceptions can contain named attributes that contain further information regarding the cause of the exception.

[8]:
```
try:
    b'\x81'.decode('utf-8')
except UnicodeError as e:
    print(e)
    print("encoding:", e.encoding)
    print("reason:", e.reason)
    print("object:", e.object)
    print("start:", e.start)
    print("end:", e.end)
```

```
'utf-8' codec can't decode byte 0x81 in position 0: invalid start byte
encoding: utf-8
reason: invalid start byte
object: b'\x81'
start: 0
end: 1
```

[1]:
```
class MySimpleException(Exception):
    pass

class MyComplexException(Exception):
    def __init__(self, text, other_information):
        super().__init__(text)
        self._info = other_information

    @property
    def info(self):
        return self._info

    def __str__(self):
        return '{} because {}'.format(self.args[0], self._info)

    def __repr__(self):
        return 'MyComplexException(test={!r}, other_information={!r})'.
 ↪format(self.args[0], self._info)
```

[2]:
```
try:
    raise MySimpleException("I'm a humble simple exception")
except MySimpleException as e:
    print(e)
```

```
I'm a humble simple exception
```

[3]:
```
try:
    raise MyComplexException("I'm a more complex exception", 'With even more␣
 ↪data to be provided, and accessible')
except MyComplexException as e:
    print(e)
    print(e.args)
    print(e.info)
```

```
I'm a more complex exception because With even more data to be provided, and
accessible
("I'm a more complex exception",)
With even more data to be provided, and accessible
```

1.1 Chaining Exceptions

Chaining exceptions can happen in two way `implicit` and `explicit`.

1.1.1 Implicit chaining

This way of chaining exceptions happens when handling an exception another exception occurs. The way Python associates this two exceptions is by assigning the original exception to the `__context__` attribute of the new exception.

1.1.2 Explicit chaining

This way of chaining exceptions happens when the handling of an exception occurs inside the handling of another exception. This is also used when translating an exception from one type into another. The way Python associates this two exceptions is by assigning the original exception to the `__cause__` attribute of the new exception.

[4]:
```
try:
    raise MySimpleException("I'm a humble simple exception")
except MySimpleException as e:
    raise IOException("WOLOLO")
```

```
---------------------------------------------------------------------------
MySimpleException                         Traceback (most recent call last)
<ipython-input-4-fb848da3632a> in <module>
      1 try:
----> 2     raise MySimpleException("I'm a humble simple exception")
      3 except MySimpleException as e:

MySimpleException: I'm a humble simple exception

During handling of the above exception, another exception occurred:

NameError                                 Traceback (most recent call last)
<ipython-input-4-fb848da3632a> in <module>
```

```
      2     raise MySimpleException("I'm a humble simple exception")
      3 except MySimpleException as e:
----> 4     raise IOException("WOLOLO")

NameError: name 'IOException' is not defined
```

[7]:
```python
try:
    raise MySimpleException("I'm a humble simple exception")
except MySimpleException as e:
    try:
        raise Exception("WOLOLO")
    except Exception as f:
        print(e)
        print(f)
        print(f.__context__ is e)
```

```
I'm a humble simple exception
WOLOLO
True
```

[11]:
```python
try:
    raise MySimpleException("I'm a humble simple exception")
except MySimpleException as e:
    raise Exception("WOLOLO") from e
```

```
---------------------------------------------------------------------------
MySimpleException                         Traceback (most recent call last)
<ipython-input-11-458406941774> in <module>
      1 try:
----> 2     raise MySimpleException("I'm a humble simple exception")
      3 except MySimpleException as e:

MySimpleException: I'm a humble simple exception

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

Exception                                 Traceback (most recent call last)
<ipython-input-11-458406941774> in <module>
      2     raise MySimpleException("I'm a humble simple exception")
      3 except MySimpleException as e:
----> 4     raise Exception("WOLOLO") from e

Exception: WOLOLO
```

[12]:
```python
try:
    raise MySimpleException("I'm a humble simple exception")
```

```python
except MySimpleException as e:
    try:
        raise Exception("WOLOLO") from e
    except Exception as f:
        print(e)
        print(f)
        print(f.__cause__ is e)
```

```
I'm a humble simple exception
WOLOLO
True
```