Python内置异常脑图
ver. 3.12
制作：黄健宏

BaseException
所有内置异常的基类。
它不应该被用户自定义类直接继承。

注意事项

- 在带有提及异常SomeException的try...except语句中，SomeException及其子类会被处理，但SomeException所继承的父类则不会。

```
>>> class SomeException(Exception):
...     pass
...
>>> class UnderException(SomeException):
...     pass
...
```

```
# 派生的子类异常也会被捕捉
>>> try:
...     raise UnderException
... except SomeException:
...     print("SomeException founded!")
...
SomeException founded!
```

```
# 但父类异常不会被捕捉
>>> try:
...     raise Exception
... except SomeException:
...     print("SomeException founded!")
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
Exception
```

- 通过子类化创建的两个不相关异常类永远是不等效的，既使它们具有相同的名称。
- 内置异常可通过解释器或内置函数生成。
- 除非另有说明，它们都会有一个提示导致错误详细原因的“关联值”。这可以时一个字符串或多个信息项组成的元组。关联值通常会参数被传递给异常类的构造器。

方法

- str()
返回实例的参数表示，无参数则返回空字符串。
- with_traceback(tb)
将tb设为新的异常回溯信息并返回异常对象。
  - 一个将SomeException实例转换为OtherException实例而保留回溯信息的例子：

```
try:
    ...
except SomeException:
    tb = sys.exception().__traceback__
    raise OtherException(...).with_traceback(tb)
```

- add_note(note)
将字符串note添加到在异常字符串之后的标准回溯显示的注释中。

属性

- args
传递给异常构造器的参数元祖。
某些内置异常接受特定数量的参数并赋予此元组中的元素特殊的意义，而其他异常通常只接受一个给出错误信息的字符串。
- __traceback__
保存关联到该异常的回溯对象的可写字段。
- __notes__
由此异常的注释组成的列表，通过add_node()添加。
该属性在调用add_note()时创建。
- __context__
__cause__
__suppress_context__
这三个属性提供了引发异常所在上下文的信息。

BaseException的子类

- BaseExceptionGroup
见ExceptionGroup。
- GeneratorExit
当一个generator或coroutine被关闭时将被引发，详见generator.close()和coroutine.close()。
它直接继承自BaseException而不是Exception，因为它从技术上来说并不是一个错误。
- KeyboardInterrupt
当用户按下中断键时将被引发。
在执行期间，会定期检测中断信号。
该异常继承自BaseException以确保不会被处理Exception的代码意外捕获，这样可以避免退出解释器。
- SystemExit
此异常由sys.exit()函数引发。
它继承自BaseException而不是Exception以确保不会被处理Exception的代码意外捕获，从而允许此异常正常地向上传播并导致解释器退出。
如果它未被处理，那么Python解释器将会不打印任何栈回溯信息就直接退出。
  - 注意事项
    - 构造器接受的可选参数与传递给sys.exit()的相同：
- 如果该值为整数，那么它指明系统退出状态码（会传递给C的exit()函数）；
- 如果该值为None，那么退出状态码为0；
- 如果改制为其他类型（比如字符串），那么打印对象的值并将退出状态码设置为1。
    - 对sys.exit()的调用会被转换为一个异常以便能够执行清理处理程序（try语句的finally字句），并使得调试器可以执行一段脚本而不冒失去控制的风险。
如果确实需要立即退出（比如在调用os.fork()之后的子进程中），那么可以使用os._exit()。
  - 属性
    - code
传递给构造器的退出状态码或错误信息（默认为None）。
- Exception
所有内置的非系统退出类异常都派生自该类。
所有用户自定义异常都应该派生自此类。
  - 注意事项
    - 内置异常类可以被子类化以定义新的异常，但鼓励程序员从Exception类或它的某个子类而不是BaseException来派生新的异常。

Exception的子类

- ArithmeticError
此基类用于派生针对各种算数类错误而引发的内置
  - 子类
    - OverflowError
当算术运算的结果大到无法表示时将被引发。
    - FloatingPointError
暂未被使用。
    - ZeroDivisionError
当除法或取余运算的第二个参数为零时引发。
关联的值是一个字符串，指明操作数和运算的类型。
- AssertionError
当assert语句失败时被引发。
- AttributeError
当前属性引用或赋值失败时将被引发。
  - 注意事项
    - 从3.10版本开始，可以通过关键字参数为其设置name和object属性，分别代表要访问的属性名称和访问该属性的对象。
- BufferError
当与缓冲区相关的操作无法执行时被引发。
- EOFError
当input()函数未读取任何数据即达到文件结束条件（EOF）时将被引发。
- LookupError
此基类用于派生当映射或序列锁使用的键/索引无效时引发的异常：IndexError、KeyError。
这可以通过codecs.lookup()来直接引发。
- ImportError
当import语句尝试加载模块遇到麻烦时将被引发，当from ... import中的载入列表存在无法找到的名称时也会被引发。
  - 注意事项
    - 从3.3版本开始，可以通过关键字参数为其设置name和patch属性，前者记录尝试导入的模块名称，而后者则指向任何触发异常的文件路径。
- ModuleNotFoundError
ImportError的子类，当一个模块无法被定位时由import引发。
当在sys.modules中找到None时也会被引发。
3.6新版功能。
- IndexError
当序列抽取超出范围时将被引发。
- KeyError
当现有键集合中找不到指定的映射（字典）键时将被引发。
- MemoryError
当一个操作耗尽内存但情况仍然可以（通过删除一些对象）进行挽救时将被引发。
关联的值是一个字符串，指明是哪种（内部）操作耗尽了内存。
- NameError
当某个局部或全局名称未找到时将被引发。此异常仅用于非限定名称。
  - 注意事项
    - 3.10版本新增了name属性，可以使用关键字参数进行设置，用于记录尝试访问的对象名称。
- NotImplementedError
此异常派生自RuntimeError。
在用户自定义的基类中，抽象方法应当要求其派生类重载该方法，或是在其要求所开发的类提示具体实现尚待添加时引发此异常。
- OSError([arg])
OSError(errno, strerror[, filename[, winerror[, filename2]]])
此异常在系统函数返回系统相关错误时被引发，此类错误包括I/O操作失败等。
此构造器实际返回的往往是OSError的某个子类，具体取决于errno的值。
  - 属性
    - errno
来自于C变量errno的数字错误码。
    - winerror
在Windows下，给出原生的Windows错误码。
    - strerror
操作系统提供的相应错误信息。
    - filename
filename2
对于与文件系统路径有关的异常（如open()或os.unlink()），filename是传给函数的文件名。
对于涉及两个文件系统路径的函数（如os.rename()），filename2将是传给函数的第二个文件名。
  - 注意事项
    - 下列异常被保留以便与之前的版本兼容，从Python 3.3开始，它们都是OSError的别名：
EnvironmentError
IOError
WindowsError
  - OSError的子类
    - BlockingIOError
当一个操作将在设置为非阻塞操作的对象（比如套接字）上发生阻塞时引发。
对应于errno EAGAIN、EALREADY、EWOULDBLOCK和EINPROGRESS。
      - 除了OSError已有的属性，这个类还有一个额外属性：
characters_written
一个表示在阻塞前已写入到流的字符数。
当使用来自io模块的带缓冲I/O类时此属性可用。
    - ChildProcessError
当一个子进程上的操作失败时引发。
对应于errno ECHILD。
    - ConnectionError
与连接相关问题的基类
      - 子类
        - BrokenPipeError
当试图写入一个管道而其另一端已关闭，或者试图写入一个套接字而其已关闭写入时引发。
对应于errno EPIPE和ESHUTDOWN。
        - ConnectionAbortedError
当一个连接尝试被对端中止时引发。
对应于errno ECONNABORTED。
        - ConnectionRefusedError
当一个连接尝试被对端拒绝时引发。
对应于errno ECONNREFUSED。
        - ConnectionResetError
当一个连接尝试被对端重置时引发。
对应于errno ECONNRESET。
    - FileExistsError
当试图创建一个已存在的文件或目录时引发。
对应于errno EEXIST。
    - FileNotFoundError
当请求的文件或目录不存在时引发。
对应于errno ENOENT。
    - InterruptedError
当一个系统调用被传入的信号中断时引发。
对应于errno EINTR。
      - 3.5 版更改: 当系统调用被某个信号中断时，Python 现在会重试系统调用，除非该信号的处理程序引发了其它异常 (原理参见 PEP 475) 而不是引发 InterruptedError。
    - IsADirectoryError
当请求对一个目录执行文件操作时引发。
对应于errno EISDIR。
    - NotADirectoryError
当请求对一个非目录执行目录操作时引发。
在大多数POSIX平台上，它还可能在某个操作试图将一个非目录作为目录打开或遍历时引发。
对应于errno ENOTDIR。
    - PermissionError
在没有足够权限的情况下试图运行某个操作时引发——例如文件系统权限。
对应于errno EACCES、EPERM和ENOTCAPABLE。
      - 3.11.1 版更改: WASI 的 ENOTCAPABLE 现在被映射至 PermissionError。
    - ProcessLookupError
当一个系统函数在系统曾经发生超时的情况下引发。
对应于errno ETIMEOUT。
- RecursionError
解释器检测发现超过最大递归深度时被引发。
此类派生自RuntimeError。
- ReferenceError
当weakref.proxy()函数所创建的弱引用访问该应用的某个已被垃圾回收的属性时被引发。
- RuntimeError
当检测到一个不归属于任何其他类别的错误时将被引发。
关联值时一个指明究竟发生了什么问题的字符串。
- StopIteration
由内置函数next()和迭代器的__next__()方法引发，用来表示该迭代器无法产生下一项。
  - 属性
    - value
该异常只有这一个属性，它在构造该异常时作为参数给出，默认为None。
  - 注意事项
    - 当一个generator或coroutine函数返回时，将引发一个新的StopIteration，而函数返回的值则会被用作异常构造器的value形参。
    - 从3.7版本开始，由生成器代码直接或间接引发的StopIteration将被转换为RuntimeError。
- StopAsyncIteration
必须由一个异步迭代器对象的__anext__()方法来引发以停止迭代操作。
3.5版本新功能。
- SyntaxError(message, details)
当解释器遇到语法错误时引发。
这可以发生在import语句，对内置拿书compile()、exec()或eval()的调用，或是读取原始脚本或标准输入（也包括交互模式）的时候。
  - 方法
    - str()
返回错误消息。
错误详情为一个元组，其成员可以在单独的属性中分别获取。
  - 属性
    - filename
发生语法错误所在文件的名称
    - lineno
发生错误所在文件中的行号。
行号从索引1开始：文件中首行的lineno为1。
    - offset
发生错误所在文件中的列号。
列号从索引1开始：文件中首歌字符的offset为1。
    - text
错误所涉及的源代码文件。
    - end_lineno
发生的错误在文件中的末尾行号，该索引从1开始。
3.10版新增。
    - end_offset
发生的错误在文件中的末尾列号，该索引从1开始。
3.10版新增。
- IndentationError
与不正确的缩进相关的语法错误基类。
这是SyntaxError的一个子类。
- TabError
当缩进包含对制表符和空格符不一致的使用时将被引发。
这是IndentationError的一个子类。
- SystemError
当解释器发现内部错误，但情况看起来尚未严重到要放弃所有希望时将被引发。
关联的值是一个指明发生了什么问题的字符串（表示为低层级的符号）。
- TypeError
当一个操作或函数被应用于类型不适当的对象时引发。
关联的值是一个字符串，给出有关类型不匹配的详情。
  - 注意事项
    - 此异常可以由用户代码引发，以表明尝试对某个对象进行的操作不受支持也不应当受支持。
如果某个对象应当支持给定的操作但尚未提供相应的实现，那么应该引发NotImplementedError异常。
    - 传入参数的类型错误（比如要求int却传入了list）应当导致TypeError，但传入参数的值错误（比如传入的数值超过了范围）那么应当导致ValueError。
- UnboundLocalError
当在某个函数或方法中引用某个局部变量，但该变量并未绑定任何值时引发。
此异常时NameError的一个子类。
- UnicodeError
当发生与Unicode相关的编码或解码错误时引发。
此异常时ValueError的一个子类。
  - 属性
    - encoding
引发错误的编码名称。
    - reason
描述特定编码器错误的字符串。
    - object
编码器试图要编码或解码的对象。
    - start
object中无效数据的开始位置索引。
    - end
object中无效数据的末尾位置索引（不含）。
- UnicodeEncodeError
编码过程中发生与Unicode相关的错误时引发。
此异常是UnicodeError的一个子类。
- UnicodeDecodeError
解码过程中发生与Unicode相关的错误时引发。
此异常是UnicodeError的一个子类。
- UnicodeTranslateError
在转写过程中发生与Unicode相关的错误时引发。
此异常是UnicodeError的一个子类。
- ValueError
当操作或函数接收到具有正确类型但值不适合的参数，并且情况不能用更精确的异常如IndexError来描述时引发。
- Warning
警告类别的基类。
  - Warning的子类
    - UserWarning
用户代码产生警告的基类。
    - DeprecationWarning
废弃特性警告的基类，用于警告其他Python开发者。
会被默认警告过滤器忽略。
    - PendingDeprecationWarning
对于已过期并计划在未来弃用，但目前尚未弃用的特性相关警告的基类。
会被默认警告过滤器忽略。
    - SyntaxWarning
与模糊的语法相关的警告基类。
    - RuntimeWarning
与模糊的运行时行为相关的警告基类。
    - FutureWarning
如果所发出的警告是针对Python所编写应用的最终用户的，则以此作为与已弃用特性相关警告的基类。
    - ImportWarning
与在模块导入中可能的错误相关的警告基类。
    - UnicodeWarning
与Unicode相关的警告基类。
    - EncodingWarning
与编码格式相关的警告基类。
3.10版新功能
    - BytesWarning
与bytes和bytearray相关的警告基类。
    - ResourceWarning
资源使用相关警告的基类。
会被默认警告过滤器忽略。
3.2版新功能
- ExceptionGroup(msg, excs)
BaseExceptionGroup(msg, excs)
异常组。
将多个异常包装在序列excs中，msg必须为字符串。
  - 注意事项
    - 这两个类之间的区别在于 BaseExceptionGroup 扩展了 BaseException 并且它可以包装任何异常，而 ExceptionGroup 则扩展了 Exception 并且它只能包装 Exception 的子类。
这样的设计是为了使得 except Exception 只捕获 ExceptionGroup 而不捕获 BaseExceptionGroup。
    - BaseExceptionGroup 构造器返回一个 ExceptionGroup 而不是 BaseExceptionGroup，如果所包含的全部异常都是 Exception 的实例的话，因此它可以被用来制造自动化的选择。 在另一方面，ExceptionGroup 构造器则会引发 TypeError，如果所包含的任何异常不是 Exception 的子类的话。
    - 跟ExceptionGroup一样，任何BaseExceptionGroup的子类也是Exception的子类，只能包装Exception的实例。
    - 3.11新版功能。
  - 属性
    - message
传递给构造器的msg参数。只读属性。
    - exceptions
传递给构造器的excs序列中由异常组成的元组。
只读属性。
  - 方法
    - subgroup(condition)
返回一个只包含来自当前组的匹配condition异常的异常组，如果结果为空则返回None。
      - condition参数是一个函数，用于接受异常并判断它是否应该纳入子分组异常；又或者是一个异常类型；又或者是一个由异常类型组成的元组，用于通过与except子句所使用的相同检测来检测是否匹配。
    - split(condition)
类似于subgroup()，但返回(match, rest)对，其中match为subgroup(codition)而rest为剩余的非匹配部分。
    - derive(excs)
返回一个具有相同message的异常组，但会将异常包装在excs中。
      - 此方法是由 subgroup() 和 split() 使用的。 子类需要重载它以便让 subgroup() 和 split() 返回相应子类的实例而不是 ExceptionGroup。
      - subgroup() 和 split() 会从原始异常组拷贝 __traceback__, __cause__, __context__ 和 __notes__ 字段到 derive() 所返回的异常组，这样这些字段就不需要被 derive() 更新。