

Программирование, лекция 11. Исключения. Итерируемые объекты, генераторы

Кафедра ИУ7 МГТУ им. Н. Э. Баумана,
2022 год

Виды ошибок

Синтаксические ошибки - ошибки интерпретации исходного текста программы при её запуске (компиляции). Наличие синтаксических ошибок не позволит программе запуститься (скомпилироваться).

Ошибки времени выполнения - ошибки, возникающие в процессе выполнения программы: деление на 0, некорректное обращение к типам данных, ошибки при работе с различными объектами, в том числе файлами, и т. д.

Исключения

Исключения - тип данных, позволяющий классифицировать ошибки времени выполнения и обрабатывать их.

```
try:
    ...операторы...
except ExceptionType [as err]:
    ...код обработки ошибки...
except ExceptionType2:
    ... код обработки ошибки 2...
except Exception:
    ... код обработки всех остальных ошибок...
else:
    ... при обработке не было ошибок...
finally:
    ... завершение обработки ...
```

Создание исключений, оператор raise

```
raise Exception('some error...')
```

```
raise ValueError
```

Иерархия встроенных исключений

```
BaseException
├── BaseExceptionGroup
├── GeneratorExit
├── KeyboardInterrupt
├── SystemExit
├── Exception
│   ├── ArithmeticError
│   │   ├── FloatingPointError
│   │   ├── OverflowError
│   │   └── ZeroDivisionError
│   ├── AssertionError
│   ├── AttributeError
│   ├── BufferError
│   ├── EOFError
│   ├── ExceptionGroup [BaseExceptionGroup]
│   ├── ImportError
│   │   └── ModuleNotFoundError
│   ├── LookupError
│   │   ├── IndexError
│   │   └── KeyError
│   ├── MemoryError
│   ├── NameError
│   │   └── UnboundLocalError
│   ├── OSError
│   │   ├── BlockingIOError
│   │   ├── ChildProcessError
│   │   ├── ConnectionError
│   │   │   ├── BrokenPipeError
│   │   │   ├── ConnectionAbortedError
│   │   │   ├── ConnectionRefusedError
│   │   │   └── ConnectionResetError
│   │   ├── FileExistsError
│   │   ├── FileNotFoundError
│   │   ├── InterruptedError
│   │   ├── IsADirectoryError
│   │   ├── NotADirectoryError
│   │   ├── PermissionError
│   │   ├── ProcessLookupError
│   │   └── TimeoutError
│   ├── ReferenceError
│   ├── RuntimeError
│   │   ├── NotImplementedError
│   │   └── RecursionError
│   ├── StopAsyncIteration
│   ├── StopIteration
│   ├── SyntaxError
│   │   ├── IndentationError
│   │   └── TabError
│   ├── SystemError
│   ├── TypeError
│   ├── ValueError
│   │   ├── UnicodeError
│   │   │   ├── UnicodeDecodeError
│   │   │   ├── UnicodeEncodeError
│   │   │   └── UnicodeTranslateError
│   └── Warning
│       ├── BytesWarning
│       ├── DeprecationWarning
│       ├── EncodingWarning
│       ├── FutureWarning
│       ├── ImportWarning
│       ├── PendingDeprecationWarning
│       ├── ResourceWarning
│       ├── RuntimeWarning
│       ├── SyntaxWarning
│       ├── UnicodeWarning
│       └── UserWarning
```

Цепочки исключений

В процессе обработки одного исключения может произойти (или быть выброшено через `raise`) другое исключение, так, что обработка исключений будет выполняться по цепочке.

Класс Exception. Дочерние классы исключений

```
class MyException(Exception):  
    pass
```

```
class ParametrizedException(Exception):  
    def __init__(self, my_param, msg=None):  
        super().__init__(msg)  
    ...
```

Ленивые (отложенные) вычисления

Ленивые вычисления - применяемая в некоторых языках программирования стратегия, согласно которой вычисления следует откладывать до тех пор, пока не понадобится их результат

Итерируемые объекты

Функция `next()`

В классе должны быть методы:

`__next__`, `__iter__`

Генераторы

- генераторные выражения: `(i**2 for i in range(10))`
- `yield`