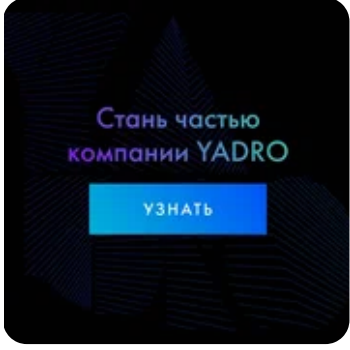


# Встроенные исключения в Python



oneweekoffer.yadro.com

РЕКЛАМА

## Ищем ведущих программистов в команду YADRO.

Ждем амбициозных и талантливых, которые горят инновационными идеями  
Заполни анкету

Узнать больше

[Справочник по языку Python3.](#) / Встроенные исключения в Python

В Python все [исключения](#) должны быть экземплярами класса, производного от [BaseException](#). В [конструкции try с инструкцией except](#), которое упоминает определенный класс, это выражение также обрабатывает любые классы исключений, производные от этого класса, но не классы исключений, из которых он произведен. Два класса исключений, которые не связаны через подклассы, никогда не эквивалентны, даже если они имеют одно и то же имя.

[Встроенные исключения](#), могут быть сгенерированы интерпретатором или встроенными функциями. За исключением некоторых случаев, они имеют "ассоциированное значение", указывающее подробную причину ошибки. Это может быть строка или кортеж из нескольких элементов информации, например код ошибки и строка объясняющая код. Связанное значение обычно передается в качестве аргументов конструктору класса исключений.

Пользовательский код может вызывать встроенные исключения. Это может быть использовано для тестирования обработчика исключений или для сообщения об ошибке, как в ситуации, когда интерпретатор вызывает то же самое исключение. Но будьте осторожны, чтобы ничто не мешало пользовательскому коду вызывать недопустимую ошибку.

Встроенные классы исключений могут быть подклассами для определения новых исключений. Программистам рекомендуется [создавать новые исключения](#) из класса [исключений Exception](#) или одного из его подклассов, а не из [BaseException](#). Дополнительные сведения об определении пользовательских исключений доступны в материале "[Пользовательские исключения](#)".

При вызове или повторном вызове исключения в инструкции `except` или `finally` - `__context__` автоматически устанавливает значение последнего перехваченного исключения. Если новое исключение не обрабатывается, то в конечном итоге отображается обратная трассировка, включающая исходное исключение и окончательное исключение.

При вызове нового исключения вместо использования простого вызова для повторного вызова обрабатываемого в данный момент исключения, контекст неявного исключения можно дополнить явной причиной с [помощью оператора raise](#):

```
raise new_exc from original_exc
```

Выражение следующее за `from`, должно быть исключением или `None`. Оно будет установлен как `__cause__` для вызванного исключения. Установка `__cause__` также неявно устанавливает атрибут `__suppress_context__` в `True`, так что использование `raise new_exc from None` эффективно заменяет старое исключение новым для целей отображения, например преобразование `KeyError` в `AttributeError`, оставляя старое исключение доступным в `__context__` для самоанализа при отладке.

Код отображения обратной трассировки по умолчанию показывает эти связанные исключения в дополнение к обратной трассировке для самого исключения. Явное цепное исключение в `__cause__`, при наличии, всегда отображается. Явно связанное исключение в `__context__` показывается, только если `__cause__` установлено в `None` и `__suppress_context__` ложно.

В любом случае само исключение всегда отображается после любых цепочек исключений, так что последняя строка трассировки всегда показывает последнее исключение, которое было вызвано.

## [Иерархия встроенных исключений в Python](#)

Иерархия встроенных исключений.

## [Базовые классы исключений в Python](#)

Следующие исключения используются в основном как базовые классы для других исключений.

[Исключения, наследуемые от BaseException в Python](#)

Исключения, наследуемые от BaseException.

[Исключения наследуемые от Exception в Python](#)

Все перечисленные здесь встроенные исключения, являются производными от класса Exception, который в свою очередь наследуется от базового класса BaseException

[Арифметические ошибки: ArithmeticError в Python](#)

Исключение ArithmeticError - базовый класс для тех встроенных исключений: OverflowError, ZeroDivisionError, FloatingPointError

[Исключения ExceptionGroup и BaseExceptionGroup в Python](#)

Исключения ExceptionGroup и BaseExceptionGroup заключают исключения в последовательность excs. Аргумент msg должен быть строкой. Разница между этими двумя классами заключается в том, что BaseExceptionGroup расширяет BaseException и может обертывать любое исключение, а ExceptionGroup расширяет Эксер

[Исключения операционной системы: OSError в Python](#)

В разделе представлены исключения, которые возникают в результате ошибок, возникающих в ходе взаимодействия приложения Python и операционной системы и являются подклассами OSError.

[Категория исключений: Warning в Python](#)

Следующие исключения используются в качестве категорий предупреждений.


[Ошибка кодировки: UnicodeError в Python](#)

Исключение UnicodeError поднимается, когда возникает ошибка кодирования или декодирования, связанная с Unicode. Это подкласс ValueError

Содержание раздела:
<ul style="list-style-type: none"><li><a href="#">ОБЗОРНАЯ СТРАНИЦА РАЗДЕЛА</a></li><li><a href="#">Иерархия встроенных исключений</a></li><li><a href="#">Базовые классы исключений</a></li><li><a href="#">Исключения, наследуемые от BaseException</a></li><li><a href="#">Исключения наследуемые от Exception</a></li><li><a href="#">Арифметические ошибки: ArithmeticError</a></li><li><a href="#">Исключения ExceptionGroup и BaseExceptionGroup</a></li><li><a href="#">Исключения операционной системы: OSError</a></li><li><a href="#">Категория исключений: Warning</a></li><li><a href="#">Ошибка кодировки: UnicodeError</a></li></ul>

ХОЧУ ПОМОЧЬ ПРОЕКТУ

РЕКЛАМА · 16+



S

skillbox.ru

Обучение Data Scientist - Обновили курс в 2023 году!

После 9 месяцев обучения сможете зарабатывать от 120 тыс. руб./месяц.

Освойте английский с нуля

Трудоустройство

Содержание курса

Формат обучения

Узнать больше