Исключения

№ урока:	3	**Курс:**	Python Essential

Средства обучения:	Python 3; интегрированная среда разработки (PyCharm или Microsoft Visual Studio + Python Tools for Visual Studio)

Обзор, цель и назначение урока

После завершения урока обучающиеся будут иметь представление об обработке ошибок и исключительных ситуаций и смогут пользоваться механизмом исключений в языке Python.

Изучив материал данного занятия, учащийся сможет:

- Иметь представление о разных видах ошибок и исключительных ситуаций
- Иметь представление о механизме исключений в языке Python
- Обрабатывать исключения
- Выбрасывать исключения
- Пользоваться стандартными классами исключений
- Создавать собственные исключения
- Пользоваться механизмом предупреждений

Содержание урока

1. Исключения
2. Обработка исключений
3. Выброс исключений
4. Синтаксические ошибки
5. Пользовательские исключения
6. Предупреждения
7. LBYL и EAFP

Резюме

- *Обработка исключительных ситуаций* или *обработка исключений* (англ. exception handling) — механизм языков программирования, предназначенный для описания реакции программы на ошибки времени выполнения и другие возможные проблемы (исключения), которые могут возникнуть при выполнении программы и приводят к невозможности (бессмысленности) дальнейшей отработки программой её базового алгоритма.
- Для обработки исключений в Python используется специальная конструкция try-except-else-finally.
- Основными блоками данной конструкции являются try и except.
- try задаёт область действия обработчика исключений. Если при выполнении операторов в данном блоке было выброшено исключение, их выполнение прерывается и управление переходит к одному из обработчиков. Если не возникло никакого исключения, блоки except пропускаются.
- Python автоматически генерирует исключения при возникновении ошибки времени выполнения.
- Код на Python может сгенерировать исключение при помощи ключевого слова raise. После него указывается объект исключения. Также можно указать класс исключения, в таком случае будет автоматически вызван конструктор без параметров. raise может выбрасывать в качестве исключений только экземпляры класса BaseException и его наследников, а также (в Python 2) экземпляры классов старого типа.
- После блока try должен следовать один или несколько блоков except (необязательно, если есть блок finally). После ключевого слова except указывается класс исключения, которое будет

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

обработано. Данный класс должен быть наследником BaseException (наследником которого является и Exception) либо классом старого типа. Данный обработчик будет перехватывать все исключения указанного класса и его наследников.

- Один обработчик может перехватывать несколько видов исключений. В таком случае передаётся кортеж классов (имена классов указываются через запятую в круглых скобках).
- Если необходимо в обработчике получить доступ к экземпляру исключения, то его можно привязать к имени при помощи ключевого слова as. Устаревшим синтаксисом, который в целях обратной совместимости поддерживается в Python 2, является указание после класса или кортежа классов исключений имени экземпляра через запятую.
- Последним из блоков except может быть стандартный обработчик, в котором не указываются никакие классы. Он не позволяет получить доступ к объекту-экземпляру.
- Блоки except обрабатываются сверху вниз и управление передаётся не больше, чем одному обработчику. Поэтому при необходимости по-разному обрабатывать исключения, находящиеся в иерархии наследования, сначала нужно указывать обработчики менее общих исключений, а затем – более общих. Также именно поэтому стандартный блок except может быть только последним. Причём если сначала расположить обработчики более общих исключений, то обработчики менее общих будут просто проигнорированы, а стандартный блок except, после которого следуют другие, является синтаксической ошибкой.
- Если ни один из заданных блоков except не перехватывает возникнувшее исключение, то оно будет перехвачено ближайшим внешним блоков try/except, в котором есть соответствующий обработчик. Если же программа не перехватывает исключение вообще, то интерпретатор завершает выполнение программы и выводит информацию об исключении в стандартный поток ошибок sys.stderr. Из этого правила есть два исключения:
    - Если исключение возникло в деструкторе объекта, выполнение программы не завершается, а в стандартный поток ошибок выводится предупреждение “Exception ignored” с информацией об исключении.
    - При возникновении исключения SystemExit происходит только завершение программы без вывода информации об исключении на экран (не касается предыдущего пункта, в деструкторе поведение данного исключения будет таким же, как и остальных).
- Для того, чтобы в обработчике исключения выполнить определённые действия, а затем передать исключение дальше, на один уровень обработчиков выше (то есть, выбросить то же самое исключение ещё раз), используется ключевое слово raise без параметров.
- В Python 3 при выбросе исключения в блоке except, старое исключение сохраняется в атрибуте данных __context__ и если новое исключение не обработано, то будет выведена информация о том, что новое исключение возникло при обработке старого («During handling of the above exception, another exception occurred:»). Также можно связывать исключения в одну цепь или заменять старые новыми. Для этого используется конструкция raise новое_исключение from старое_исключение либо raise новое_исключение from None. В первом случае указанное исключение сохраняется в атрибуте __cause__ и атрибут __suppress_context__ (который подавляет вывод исключения из __context__) устанавливается в True. Тогда, если новое исключение не обработано, будет выведена информация о том, что старое исключение является причиной нового («The above exception was the direct cause of the following exception:»). Во втором случае __suppress_context__ устанавливается в True и __cause__ в None. Тогда при выводе исключения оно, фактически, будет заменено новым (хотя старое исключение всё ещё хранится в __context__).
- В Python 2 нет сцепления исключений. Любое исключение, выброшенное в блоке except, заменяет старое.
- Следующим необязательным блоком является else. Операторы внутри него выполняются, если никакое исключение не возникло. Он предназначен для того, чтобы отделить код, который может вызвать исключение, которое должно быть обработано в данном блоке try/except, от кода, который может вызвать исключение того же класса, которое должно быть перехвачено на уровне выше, и свести к минимуму количество операторов в блоке try.
- Последним необязательным блоком является finally. Операторы внутри него выполняются независимо от того, возникло ли какое-либо исключение. Он предназначен для выполнения так называемых cleanup actions, то есть действий по очистке: закрытие файлов, удаление временных объектов и т.д. Если исключение не было перехвачено ни одним из блоков except,

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

то оно заново выбрасывается интерпретатором после выполнения действий в блоке finally. Блок finally выполняется перед выходом из оператора try/except всегда, даже если одна из его веток содержит оператор return (когда оператор try/except находится внутри функции), break или continue (когда оператор try/except находится внутри цикла) или возникло другое необработанное исключение при обработке данного исключения.

- Исключения могут принимать в качестве параметра конструктора любые неименованные аргументы. Они помещаются в атрибуте данных args в виде кортежа (неизменяемого списка). Чаще всего используется один строковой параметр, который содержит сообщение об ошибке. Во всех исключениях определён метод __str__, который по умолчанию вызывает str(self.args). В Python 2 также имеется атрибут message, в который помещается args[0], если len(args) == 1.
- В Python 2 стандартные классы исключений описаны в модуле exceptions, однако его не нужно импортировать явно и все имена классов доступны в __builtins__ автоматически. В Python 3 этот модуль упразднён.
- Стандартные классы исключений:
    - Базовые:
        - BaseException – базовый класс для всех исключений.
        - Exception – класс-наследник BaseException, базовый класс для для всех стандартных исключений, которые не указывают на обязательное завершение программы, и всех пользовательских исключений.
        - StandardError (Python 2) – базовый класс для всех встроенных исключений, кроме StopIteration, GeneratorExit, KeyboardInterrupt и SystemExit.
        - ArithmeticError – базовый класс для всех исключений, связанных с арифметическими операциями.
        - BufferError – базовый класс для исключений, связанных с операциями над буфером.
        - LookupError – базовый класс для исключений, связанных с неверным ключом или индексом коллекции.
        - EnvironmentError (Python 2) – базовый класс для исключений, связанных с ошибками, которые происходят вне интерпретатора Python. В Python 3 его роль выполняет OSError.
    - Некоторые из конкретных стандартных исключений:
        - AssertionError – провал условия в операторе assert.
        - AttributeError – ошибка обращения к атрибуту.
        - FloatingPointError – ошибка операции над числами с плавающей точкой.
        - ImportError – ошибка импортирования модуля или имени из модуля.
        - IndexError – неверный индекс последовательности (например, списка).
        - KeyboardInterrupt – завершение программы путём нажатия Ctrl+C в консоли.
        - MemoryError – нехватка памяти.
        - NameError – имя не найдено.
        - NotImplementedError – действие не реализовано. Предназначено, среди прочего, для создания абстрактных методов.
        - OSError – системная ошибка.
        - OverflowError – результат арифметической операции слишком большой, чтобы быть представлен.
        - RuntimeError – общая ошибка времени выполнения, которая не входит ни в одну из категорий.
        - SyntaxError – ошибка синтаксиса.
        - IndentationError – подкласс SyntaxError – неверный отступ.
        - TabError – подкласс IndentationError – смешанное использование символов табуляции и пробелов.
        - SystemError – некритичная внутренняя ошибка интерпретатора. При возникновении данного исключения следует оставить отчёт об ошибке на сайте https://bugs.python.org/
        - SystemExit – исключение, которое генерируется функцией sys.exit(). Служит для завершения работы программы.
        - TypeError – ошибка несоответствия типов данных.

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

        - UnboundLocalError – подкласс NameError – обращение к несуществующей локальной переменной.
        - ValueError – генерируется, когда функции или операции передан объект корректного типа, но с некорректным значением, причём эту ситуацию нельзя описать более точным исключением, таким как IndexError.
        - ZeroDivisionError – деление на ноль.
- Ошибка синтаксиса возникает, когда синтаксический анализатор Python сталкивается с участком кода, который не соответствует спецификации языка и не может быть интерпретирован. Поскольку, в случае синтаксической ошибки в главном модуле, она возникает до начала выполнения программы и не может быть перехвачена, учебник для начинающих в документации языка Python даже разделяет синтаксические ошибки и исключения. Однако SyntaxError – это тоже исключение, которое наследуется от Exception, и существуют ситуации, когда оно может возникнуть во время исполнения и быть обработано, а именно:
    - ошибка синтаксиса в импортируемом модуле;
    - ошибка синтаксиса в коде, который представляется строкой и передаётся функции eval или exec.
- Можно создавать собственные исключения. Они должны быть наследниками класса Exception. Принято называть исключения так, что имя их класса заканчивается словом Error.
- *Предупреждения* обычно выводятся на экран в ситуациях, когда не гарантируется ошибочное поведение и программа, как правило, может продолжать работу, однако пользователя следует уведомить о чём-либо.
- Базовым классом для предупреждений является Warning, который наследуется от Exception.
- Базовым классом-наследником Warning для пользовательских предупреждений является UserWarning.
- В модуле warning собраны функции для работы с предупреждениями. Основной является функция warn, которая принимает один обязательный параметр message, который может быть либо строкой-сообщением, либо экземпляром класса или подкласса Warning (в таком случае параметр category устанавливается автоматически) и два опциональных параметра: category (по умолчанию – UserWarning) – класс предупреждения и stacklevel (по умолчанию – 1) – уровень вложенности функций, начиная с которого необходимо выводить содержимое стека вызовов (полезно, например, для функций-обёрток для вывода предупреждений, где следует задать stacklevel=2, чтобы предупреждение относилось к месту вызова данной функции, а не самой функции).
- *LBYL* (*Look Before You Leap* – «семь раз отмерь, один раз отрежь») – стиль, который характеризуется наличием множества проверок и условных операторов. В контексте утиной типизации может означать проверку наличия необходимых атрибутов при помощи функции hasattr.
- *EAFP* (*Easier to Ask for Forgiveness than Permission* – «проще попросить прощения, чем разрешения») – стиль, который характеризуется наличием блоков try/except. В контексте утиной типизации – написание кода исходя из предположения, что данный объект реализует необходимый интерфейс, и обработка исключения AttributeError в противном случае.

Закрепление материала

- Что такое исключительная ситуация (исключение)?
- Какой оператор в Python предназначен для обработки исключений?
- Какие его основные блоки? Какое их назначение?
- Какой оператор в Python предназначен для выброса исключений?
- Чем отличается создание объекта исключения от выброса исключения?
- Каким образом можно получить экземпляр обрабатываемого исключения?
- Каким образом задать одинаковый обработчик для нескольких разных исключений?
- От какого класса наследуются все исключения?
- Что такое сцепление исключений?
- Что такое синтаксическая ошибка?
- Что такое предупреждение и как его сгенерировать?

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

Дополнительное задание

Задание
Опишите свой класс исключения. Напишите функцию, которая будет выбрасывать данное исключение, если пользователь введёт определённое значение, и перехватите это исключение при вызове функции.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные стандартные исключения, которые перечислены в данном уроке.

Задание 2
Напишите программу-калькулятор, которая поддерживает следующие операции: сложение, вычитание, умножение, деление и возведение в степень. Программа должна выдавать сообщения об ошибке и продолжать работу при вводе некорректных данных, делении на ноль и возведении нуля в отрицательную степень.

Задание 3
Опишите класс сотрудника, который включает в себя такие поля, как имя, фамилия, отдел и год поступления на работу. Конструктор должен генерировать исключение, если заданы неправильные данные. Введите список работников с клавиатуры. Выведите всех сотрудников, которые были приняты после заданного года.

Рекомендуемые ресурсы

Документация Python 3
Информация о механизме исключений
https://docs.python.org/3/tutorial/errors.html
https://docs.python.org/3/tutorial/classes.html#exceptions-are-classes-too
https://docs.python.org/3/reference/executionmodel.html#exceptions
Встроенные исключения
https://docs.python.org/3/library/exceptions.html
Модуль warnings
https://docs.python.org/3/library/warnings.html

Документация Python 2
Информация о механизме исключений
https://docs.python.org/2/tutorial/errors.html
https://docs.python.org/2/tutorial/classes.html#exceptions-are-classes-too
https://docs.python.org/2/reference/executionmodel.html#exceptions
Встроенные исключения
https://docs.python.org/2/library/exceptions.html
Модуль warnings
https://docs.python.org/2/library/warnings.html

Статьи в Википедии о ключевых понятиях, рассмотренных на этом уроке
https://ru.wikipedia.org/wiki/Обработка_исключений

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com