Винятки та їхня обробка

№ уроку:	4	**Курс:**	Python Essential

Засоби навчання:	Python 3; інтегроване середовище розробки (PyCharm або Microsoft Visual Studio + Python Tools for Visual Studio)

Огляд, мета та призначення уроку

Після завершення уроку учні матимуть уявлення про обробку помилок і виняткових ситуацій, а також зможуть користуватися механізмом обробки винятків у мові Python.

Вивчивши матеріал цього заняття, учень зможе:

- Мати уявлення про різновиди помилок і виняткових ситуацій.
- Мати уявлення про механізм винятків у мові Python.
- Обробляти винятки.
- Викидати винятки.
- Користуватися стандартними класами винятків.
- Створювати власні винятки.
- Користуватися механізмом попереджень.

Зміст уроку

1. Винятки.
2. Обробка винятків.
3. Викид винятків.
4. Синтаксичні помилки.
5. Користувацькі винятки.
6. Попередження.
7. LBYL та EAFP.

Резюме

- **Обробка виняткових ситуацій,** або **Обробка винятків** (англ. exception handling) – механізм мов програмування, який призначений для опису реакції програми на помилки часу виконання та інші можливі проблеми (винятки), які можуть виникнути під час виконання програми та призводять до неможливості (безглуздості) подальшого відпрацювання програмою її базового алгоритму.
- Для обробки винятків у Python використовується спеціальна конструкція **try-except-else-finally**.
- Основними блоками цієї конструкції є **try** та **except**.
- Блок **try** задає зону дії оброблювача винятків. Якщо під час виконання операторів у цьому блоці було викинуто виняток, їхнє виконання переривається, а управління переходить до одного з оброблювачів. Якщо не виникло жодного винятку, блоки **except** пропускаються.
- Python автоматично генерує винятки під час виникненні помилки часу виконання.
- Код на Python може згенерувати виняток за допомогою ключового слова **raise**. Після нього зазначається об'єкт винятку. Також можна вказати клас винятку. У такому разі буде автоматично викликаний конструктор без параметрів. **raise** може викидати як виняток лише екземпляри класу **BaseException** та його спадкоємців, а також (у Python 2) екземпляри класів старого типу.
- Після блоку **try** має йти один або кілька блоків **except** (необов'язково, якщо є блок **finally**). Після ключового слова except вказується клас винятку, який буде опрацьовано. Цей клас має бути спадкоємцем **BaseException** (спадкоємцем якого є **Exception**) або класом старого типу. Цей обробник перехоплюватиме всі винятки зазначеного класу та його спадкоємців.
- Один обробник може перехоплювати кілька різновидів винятків. У такому разі передається кортеж класів (імена класів зазначаються через кому в круглих дужках).

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

- Якщо необхідно в обробнику отримати доступ до екземпляра винятку, його можна прив'язати до імені за допомогою ключового слова **as**. Застарілим синтаксисом, який для зворотної сумісності підтримується в Python 2, є вказівка після класу або кортежу класів винятків імені екземпляра через кому.
- Останнім із блоків **except** може бути стандартний обробник, у якому не зазначаються жодні класи. Він не дозволяє отримати доступ до об'єкта-екземляра.
- Блоки **except** обробляються зверху вниз, і керування передається не більш як одному обробнику. Тому за необхідності по-різному обробляти винятки, які розташовані в ієрархії успадкування. Спочатку потрібно вказувати обробники менш загальних винятків, а потім – загальніших. Також саме тому стандартний блок **except** може бути лише останнім. Причому якщо спочатку розташувати обробники загальніших винятків, то обробники менш загальних будуть просто проігноровані, а стандартний блок except, після якого йдуть інші, є синтаксичною помилкою.
- Якщо жоден із заданих блоків **except** не перехоплює виняток, то він буде перехоплений найближчим зовнішнім блоком **try/except**, в якому є відповідний обробник. Якщо ж програма не перехоплює виняток взагалі, то інтерпретатор завершує виконання програми та виводить інформацію про виняток у стандартний потік помилок **sys.stderr**. У цьому правилі є два винятки:
    - Якщо виняток виник у деструкторі об'єкта, виконання програми не завершується, а в стандартний потік помилок виводиться попередження **«Exception ignored»** з інформацією про виняток.
    - Під час виникнення винятку **SystemExit** відбувається лише завершення програми без виведення інформації про виняток на екран (не стосується попереднього пункту, у деструкторі поведінка цього винятку буде такою самою, як і інших).
- Щоб в обробнику винятку виконати певні дії, а потім передати виняток далі, на один рівень обробників вище (тобто викинути той самий виняток ще раз), використовується ключове слово **raise** без параметрів.
- У Python 3 під час викиду винятку в блоці except, старий виняток зберігається в атрибуті даних __context__ і якщо новий виняток не оброблено, то буде виведена інформація про те, що новий виняток виник під час обробки старого («During handling of the above exception, another exception occurred :»). Також можна пов'язувати винятки в один ланцюг або замінювати старі новими. Для цього використовується конструкція raise новий_виняток from старий_виняток або raise новий_виняток from None. У першому випадку цей виняток зберігається в атрибуті __cause__ і атрибут __suppress_context__ (який пригнічує вивід винятку з __context__) встановлюється в True. Якщо новий виняток не опрацьовано, буде виведена інформація про те, що старий виняток є причиною нового. У другому випадку __suppress_context__ встановлюється в True і __cause__ в None. Тоді під час виведенні винятку він фактично буде замінено новим (хоча старий виняток все ще зберігається в __context__).
- У Python 2 немає зчеплення винятків. Будь-який виняток, викинутий у блоці except, замінює старий.
- Наступним необов'язковим блоком є **else**. Оператори всередині нього виконуються, якщо жоден виняток не виник. Він призначений для того, щоб відокремити код, який може викликати виняток, який має бути оброблений в блоці **try/except**, від коду, який може викликати виняток того ж класу, який має бути перехоплений на рівні вище, і звести до мінімуму кількість операторів у блоці **try**.
- Останнім необов'язковим блоком є **finally**. Оператори всередині нього виконуються незалежно від того, чи виник якийсь виняток. Він призначений для виконання так званих **cleanup actions**, тобто дій з очищення: закриття файлів, видалення тимчасових об'єктів тощо. Якщо виняток не був перехоплений жодним з блоків **except**, він заново викидається інтерпретатором після виконання дій в блоці **finally**. Блок **finally** виконується перед виходом з оператора **try/except** завжди, навіть якщо одна з його гілок містить оператор **return** (коли оператор **try/except** розташований всередині функції), **break** або **continue** (коли оператор **try/except** розташований всередині циклу) або виник інший необроблений виняток під час обробки цього винятку.
- Винятки можуть приймати як параметр конструктора будь-які неіменовані аргументи. Вони містяться в атрибуті даних args як кортеж (незмінного списку). Найчастіше використовується один рядковий параметр, який містить повідомлення про помилку. У всіх винятках визначено метод __str__, який за умовчанням викликає str(self.args). У Python 2 також є атрибут message, у який міститься args[0], якщо len(args) == 1.

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

- У Python 2 стандартні класи винятків описані в модулі exceptions. Проте його не потрібно імпортувати явно. Усі імена класів доступні в __builtins__ автоматично. У Python 3 цей модуль скасований.
- Стандартні класи винятків:
    - Базові:
        - BaseException – базовий клас для всіх винятків.
        - Exception – клас-спадкоємець BaseException, базовий клас для всіх стандартних винятків, які не вказують на обов'язкове завершення програми, та всіх користувацьких винятків.
        - StandardError (Python 2) – базовий клас для всіх вбудованих винятків, окрім StopIteration, GeneratorExit, KeyboardInterrupt та SystemExit.
        - ArithmeticError – базовий клас для всіх винятків, які пов'язані з арифметичними операціями.
        - BufferError – базовий клас для винятків, які пов'язані з операціями над буфером
        - LookupError – базовий клас для винятків, які пов'язані з неправильним ключем або індексом колекції.
        - EnvironmentError (Python 2) – базовий клас для винятків, які пов'язаних із помилками, що відбуваються поза інтерпретатором Python. У Python 3 його роль виконує OSError.
    - Деякі з конкретних стандартних винятків:
        - AssertionError – провал умови оператора assert.
        - AttributeError – помилка звернення до атрибута.
        - FloatingPointError – помилка операції над числами з плавальною точкою.
        - ImportError – помилка імпортування модуля або імені з модуля.
        - IndexError – неправильний індекс послідовності (наприклад, списку).
        - KeyboardInterrupt – завершення програми шляхом натискання Ctrl+C консолі.
        - MemoryError – нестача пам'яті.
        - NameError – ім'я не знайдено.
        - NotImplementedError – дія не реалізована. Призначений, серед іншого, для створення абстрактних методів.
        - OSError – системна помилка.
        - OverflowError – результат арифметичної операції надто великий, щоб бути представлений.
        - RuntimeError – загальна помилка часу виконання, яка не входить до жодної категорії.
        - SyntaxError – помилка синтаксису.
        - IndentationError – підклас SyntaxError – неправильний відступ.
        - TabError – підклас IndentationError – змішане використання символів табуляції та пробілів.
        - SystemError – некритична внутрішня помилка інтерпретатора. Під час виникнення цього винятку варто залишити звіт про помилку на сайті https://bugs.python.org/
        - SystemExit – виняток, що генерується функцією sys.exit(). Слугує для завершення роботи програми.
        - TypeError – помилка невідповідності типів даних.
        - UnboundLocalError – підклас NameError – звернення до ненаявної локальної змінної.
        - ValueError – генерується, коли функції чи операції передано об'єкт коректного типу, але з некоректним значенням, причому цю ситуацію не можна описати точнішим винятком, як-от IndexError.
        - ZeroDivisionError – ділення на нуль.
- **Помилка синтаксису виникає**, коли синтаксичний аналізатор Python стикається з ділянкою коду, який не відповідає специфікації мови та не може бути інтерпретований. Оскільки за синтаксичної помилки в головному модулі вона виникає до початку виконання програми і не може бути перехоплена, підручник для початківців у документації мови Python навіть поділяє синтаксичні помилки та винятки. Однак SyntaxError – це також виняток, який успадковується від

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

Exception. Наявні ситуації, коли він може виникнути під час виконання та бути оброблений, а саме:
    - помилка синтаксису в модулі, який імпортується;
    - помилка синтаксису в коді, який представляється рядком та передається функції eval або exec.
- Можна створювати власні винятки. Вони мають бути спадкоємцями класу Exception. Заведено називати винятки так, аби ім'я їхнього класу закінчувалося словом Error.
- **Попередження** зазвичай виводяться на екран у ситуаціях, коли помилкова поведінка не гарантується. Програма, як правило, може продовжувати роботу, однак користувача варто повідомити про що-небудь.
- Базовим класом для попереджень є Warning, який успадковується від Exception.
- Базовим класом-спадкоємцем Warning для користувацьких попереджень є UserWarning.
- У модулі warning зібрані функції для роботи із попередженнями. Основною є функція warn, яка приймає один обов'язковий параметр message, який може бути або рядком-повідомленням, або екземпляром класу чи підкласу Warning (у такому випадку параметр category встановлюється автоматично) та два опціональні параметри: category (за замовчуванням – UserWarning) – клас попередження та stacklevel (за замовчування – 1) – рівень вкладеності функцій, починаючи з якого необхідно виводити вміст стека викликів.
  (корисно, наприклад, для функцій-обгорток для виведення попереджень, де варто задати stacklevel=2, щоб попередження стосувалося місця виклику цієї функції, а не самої функції).
- LBYL (Look Before You Leap – «сім разів відміряй – один раз відріж») – стиль, який характеризується наявністю безлічі перевірок та умовних операторів. У контексті качиної типізації може означати перевірку наявності необхідних атрибутів за допомогою функції hasattr.
- EAFP (Easier to Ask for Forgiveness than Permission – «простіше попросити вибачення, ніж дозволу») – стиль, що характеризується наявністю блоків try/except. У контексті качиної типізації: написання коду з огляду на припущення, що цей об'єкт реалізує необхідний інтерфейс, та обробка винятку AttributeError в іншому випадку.

Закріплення матеріалу

- Що таке виняткова ситуація (виняток)?
- Який оператор Python призначений для обробки винятків?
- Які його основні блоки? Яке їхнє призначення?
- Який оператор Python призначений для викиду винятків?
- Чим відрізняється створення об'єкта винятку від викиду?
- Як можна отримати екземпляр винятку, який обробляється?
- Як задати однаковий обробник для декількох різних винятків?
- Від якого класу успадковуються усі винятки?
- Що таке зчеплення винятків? Що таке синтаксична помилка?
- Що таке попередження та як його згенерувати?

Додаткові завдання

Завдання 1
Опишіть свій клас винятку. Напишіть функцію, яка викидатиме цей виняток, якщо користувач введе певне значення, і перехопіть цей виняток під час виклику функції.

Завдання 2
Створіть програму спортивного комплексу, у якій передбачене меню: 1 - перелік видів спорту, 2 - команда тренерів, 3 - розклад тренувань, 4 - вартість тренування. Дані зберігати у словниках. Також передбачити пошук по прізвищу тренера, яке вводиться з клавіатури у відповідному пункті меню. Якщо ключ не буде знайдений, створити відповідний клас-Exception, який буде викликатися в обробнику виключень.

Самостійна діяльність учня

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

Завдання 1
Вивчіть основні стандартні винятки, які перераховані в цьому уроці.

Завдання 2
Напишіть програму-калькулятор, яка підтримує такі операції: додавання, віднімання, множення, ділення та піднесення до ступеня. Програма має видавати повідомлення про помилку та продовжувати роботу під час введення некоректних даних, діленні на нуль та зведенні нуля в негативний степінь.

Завдання 3
Опишіть клас співробітника, який вміщує такі поля: ім'я, прізвище, відділ і рік початку роботи. Конструктор має генерувати виняток, якщо вказано неправильні дані. Введіть список працівників із клавіатури. Виведіть усіх співробітників, які були прийняті після цього року.

Рекомендовані ресурси

Документація Python 3
Інформація про механізм винятків
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

Вбудовані винятки
https://docs.python.org/3/library/exceptions.html

Модуль warnings
https://docs.python.org/3/library/warnings.html

Документація Python 3
Інформація про механізм винятків
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

Вбудовані винятки
https://docs.python.org/2/library/exceptions.html

Модуль warnings
https://docs.python.org/2/library/warnings.html

Статті у Вікіпедії про ключові поняття, розглянуті на цьому уроці

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