

# ІНФОРМАТИКА ТА ПРОГРАМУВАННЯ

---

Тема 15. Обробка помилок та  
виключних ситуацій

# Помилки та виключні ситуації

- Будь-яка людська діяльність не обходиться без помилок. Це, безумовно, стосується і програмування.
- Помилки обов'язковою присутні у програмах. Є навіть жартівлива аксіома про те, що «у будь-якій налагоджені програмі обов'язково існує принаймні одна не знайдена помилка».
- Помилки у програмах не є наслідком недбалого програмування, а впливають з об'єктивної складності розроблюваних програм, яка перевищує можливості людського розуму з точки зору знаходження всіх помилок.
- Близькі до помилок і так звані «виключні ситуації», які можуть трапитись під час виконання програми, але не є суто програмними помилками. Наприклад, переривання з'єднання з мережею під час виконання програми, яка обмінюється даними у мережі.

# Помилки та виключні ситуації.2

- Певний час на помилки намагалися не звертати уваги, вважаючи, що їх нарешті вдасться повністю перемогти. Але потім, з розвитком програмування як індустрії, прийшло розуміння того, що помилки нікуди не дінуться і треба вміти з ними співіснувати.
- Варто відмітити, що нині багато складних систем тривалий час функціонують з відомими помилками, які не є фатальними.
- Як результат цього розуміння, у мовах програмування з'явилися засоби обробки помилок та виключних ситуацій.
- Надалі помилки та виключні ситуації будемо називати **виключеннями**.

# Обробка виключень

- Обробка виключень включає:
  - ідентифікацію виключення
  - класифікація виключення
  - інформування про виключення
  - реагування на виключення.
- Ідентифікація виключення – це фіксація наявності виключення.
- Класифікація виключення – це визначення типу та, можливо, джерела виникнення виключення.
- Інформування про виключення – це розповсюдження та збереження інформації про виключення, отриманої на попередніх двох кроках.
- Реагування на виключення – це виконання дій з нейтралізації наслідків виключення та/або вирішення щодо можливості та режимів подальшого виконання програми.

# Обробка виключень.2

- Найпростішою обробкою виключень є повідомлення про помилку та завершення виконання програми.
- Але така реакція є примітивною та не може бути застосована у складних програмах, які вирішують реальні задачі.
- Уявіть собі, скажімо, програму, що керує неперервним технологічним процесом та буде зупинятись після кожного виявленого виключення.
- Отже, для кожного виключення потрібно приймати самостійне рішення:
  - Чи можна продовжити виконання програми?
  - Якщо можна, то у якому режимі?
  - Якщо не можна, то які дії слід обов'язково виконати перед завершенням програми?

# Засоби обробки виключень у Python

- Ми вже зустрічались з обробкою виключень у Python.
- Якщо виникає помилка, то інтерпретатор завершує обробку програми (команди) та повертає повідомлення про помилку та місце її виникнення.
- Тобто, Python здійснює ідентифікацію стандартних виключень, що не дозволяють нормально продовжувати виконання програми.
- Python також виконує класифікацію виключень.
- Кожне виключення є об'єктом деякого класу.
- Це може бути стандартний клас або клас, що визначений користувачем.
- Усі класи виключень у Python утворюють ієрархію класів.
- Обробка виключень програмістом полягає, в тому числі, у модифікації стандартної поведінки інтерпретатора.

# Блоки try – except

- У Python засобами обробки виключень є блоки try – except.
- Синтаксис блоку try – except виглядає так:

try:

*P*

except *cls*:

*Q*

else:

*R*

- де *P*, *Q*, *R* – інструкції, *cls* – ім'я класу.

# Блоки try – except.2

- Правило виконання try – except.
- 1. Python виконує інструкцію  $P$ .
- 2. Якщо при виконанні  $P$  виникає виключення, що належить класу  $c/s$  або одному з його підкласів то Python перериває виконання  $P$  та виконує інструкцію  $Q$ , інакше завершує виконання  $P$ , після чого виконує інструкцію  $R$ .
- Слід зазначити, що при виникненні виключення, що не належить класу  $c/s$ , Python все одно перериває виконання  $P$ , але інструкція  $Q$  не буде виконуватись.



# Блоки try – except.3

- До одного try можуть відноситись декілька except з різними класами виключень.
- Тоді, у разі виникнення виключення, управління буде передано у той блок except, клас  $cls_i$  якого відповідає цьому виключенню.

try:

$P$

except  $cls_1$ :

$Q_1$

...

except  $cls_n$ :

$Q_n$

else:

$R$

- де  $P, Q_1, \dots, Q_n, R$  – інструкції,  $cls_1, \dots, cls_n$  – ім'я класів.

# Вкладені блоки `try – except`

- Блоки `try – except` також можуть бути вкладеними.
- У цьому випадку, Python спочатку перевіряє відповідність виключення внутрішньому блоку, потім тому, що охоплює внутрішній, і т.д.

# Приклад

- Видалити заданий елемент із заданого списку.

# Виконання заключних дій. *finally*

- Часто потрібно виконувати певні дії незалежно від того, чи відбулось виключення.
- Наприклад: закрити раніше відкритий файл, звільнити з'єднання з базою даних тощо.
- Для виконання таких дій у блок `try` – ехсепт додають `finally`.
- Синтаксис блоку `try` – ехсепт з використанням `finally` виглядає наступним чином:

# Виконання заключних дій. `finally`

`try:`

*P*

`except cls1:`

*Q<sub>1</sub>*

...

`except clsn:`

*Q<sub>n</sub>*

`finally:`

*S*

`else:`

*R*

- де *P*, *Q<sub>1</sub>*, ..., *Q<sub>n</sub>*, *R*, *S* – інструкції, *cls<sub>1</sub>*, ..., *cls<sub>n</sub>* – ім'я класів.
- Інструкція *S* виконується після виконання *P* і в тому випадку, коли було виключення, і тоді, коли виключення не було.

# Атрибути виключень

- Оскільки виключення у Python є класами, вони мають атрибути, як і інші класи.
- Ці атрибути можна використовувати для передачі та аналізу параметрів виключення, що дозволяє уточнити його причину.
- Щоб скористатися атрибутами виключення треба у виразі `except` присвоїти значення виключення об'єкту відповідного класу.
- Щоб зробити це, пишуть

`except c/s as c:`

*Q*

- Після цього *c* набуває значення об'єкту класу *c/s*, тобто, класу виключення.
- Далі у інструкції *Q* можна використовувати *c* як звичайний об'єкт з його полями та методами.

# Ініціювання виключень

- Програміст може самостійно ініціювати виключення. Для цього використовують команду `raise e`
  - де `e` – об'єкт класу виключення, що ініціюється.
- Після виконання цієї команди Python діє аналогічно тому, як він діє при виникненні інших виключень.
- Тобто, якщо є охоплюючий блок `try – except`, який обробляє виключення класу, до якого належить `e`, управління передається відповідному `except`.
- Інакше виконання програми закінчується та виводиться повідомлення про помилку.

# Приклад

- У текстовому файлі *f* записано цілі числа. Переписати всі парні числа у файл *g*. Використати блоки обробки помилок.



# Стандартні класи виключень Python

- У Python є ряд стандартних класів виключень, які утворюють ієрархію. Коротко ці класи показані у таблиці нижче.

Клас виключення	Опис
<code>BaseException</code>	Базовий клас виключень
<code>+-- SystemExit</code>	Вихід з системи (після виконання <code>exit()</code> )
<code>+-- KeyboardInterrupt</code>	Переривання від клавіатури (натиснення <code>ctrl+c</code> )
<code>+-- GeneratorExit</code>	Завершення генерації послідовності
<code>+-- Exception</code>	Основний клас виключення. Всі власні виключення повинні походити від цього класу
<code>+-- StopIteration</code>	Зупинка ітерації (немає наступного елемента для циклу)
<code>+-- ArithmeticError</code>	Арифметична помилка
<code>+-- AssertionError</code>	Помилка твердження (умова твердження хибна)
<code>+-- AttributeError</code>	Помилка у атрибуті
<code>+-- BufferError</code>	Помилка буферизації

# Стандартні класи виключень Python.2

Клас виключення	Опис
+-- EOFError	Помилка кінця файлу
+-- ImportError	Помилка імпорту модуля
+-- LookupError	Помилка індексу або ключа
+-- MemoryError	Помилка нестачі пам'яті
+-- NameError	Помилкове ім'я
+-- OSError	Помилка операційної системи
+-- ReferenceError	Помилка у посиланні
+-- RuntimeError	Помилка під час виконання
+-- SyntaxError	Синтаксична помилка
+-- SystemError	Системна помилка
+-- TypeError	Помилка типу
+-- ValueError	Помилкове значення
+-- Warning	Попередження

# Власні класи виключень

- Власні класи виключень використовують тоді, коли стандартних класів не вистачає для класифікації виключень.
- Тобто, якщо треба більш точно вказати, що джерело виключення – у створеній програмі або класі.
- Щоб створити власне виключення, достатньо описати клас, який походить від стандартного класу `Exception` або його нащадків.
- У цьому класі треба реалізувати методи `__init__` та `__str__`.
- У `__init__` визначають атрибути власного виключення, а у `__str__`, - повідомлення, яке буде видаватись при виключенні.

## Власні класи виключень.2

- Власні класи виключень також можуть мати свою ієрархію.
- Ієрархія доцільна у випадку, якщо, скажімо, деякий модуль (клас) може ініціювати різнотипні виключення, але потрібно показати, що всі вони походять саме з даного модуля (класу).
- Оскільки ехсерт реагує не тільки на вказане виключення, але й на його нащадків, деякі програми можуть обробляти тільки кореневе виключення модуля (класу), а інші – для більш точної класифікації – конкретні виключення, що є підкласами кореневого.

# Приклад

- У чергу додають  $n$  елементів. Потім проводять  $m$  випадкових випробувань. Результат кожного випробування - значення 0 або 1.
- Якщо результат 0, то елемент береться з початку черги.
- Якщо результат 1, то новий елемент додається до черги.
- Показати хід випробувань. Після завершення випробувань показати залишок черги.
- Реалізувати обробку помилки взяття елемента з порожньої черги.

# Твердження про стан програми

- Твердження про стан програми дозволяють перевірити істинність умов, які залежать від змінних.
- Синтаксис твердження про стан програми виглядає так:  
`assert F, msg`
  - де F - умова, msg – рядок.
- Правило виконання assert:
- Python перевіряє умову F. Якщо вона істинна (True), виконання програми продовжується. Якщо умова F хибна, то Python ініціює виключення AssertionError. Це виключення обробляється за загальними правилами.
- Твердження assert подібні до Хоарівських трійок, які ми розглядали раніше.
- Але ці твердження містять тільки одну умову, яка перевіряється в одній точці програми.

## Приклад: "Швидке" обчислення натуральної степені дійсного числа

- "Швидке" обчислення натуральної степені дійсного числа – це обчислення  $x^n$  за приблизно  $\log_2 n$  кроків.

# Резюме

- Ми розглянули:
  1. Поняття помилки та виключної ситуації.
  2. Засоби обробки виключень у Python
  3. Блоки try – except
  4. Атрибути виключень, ініціювання виключень.
  5. Стандартні класи виключень у Python.
  6. Власні виключення
  7. Твердження про програми



# Де прочитати

1. A Byte of Python (Russian) Версия 2.01 Swaroop С Н (Translated by Vladimir Smolyar),  
<http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf>
2. Марк Лутц, Изучаем Python, 4-е издание, 2010, Символ-Плюс
3. Python 3.4.3 documentation
4. Марк Саммерфилд, Программирование на Python 3. Подробное руководство. - Символ-Плюс, 2009.
5. [http://www.python-course.eu/python3\\_exception\\_handling.php](http://www.python-course.eu/python3_exception_handling.php)