**Практическая работа №6**

Шульга Иван 21.11.2024

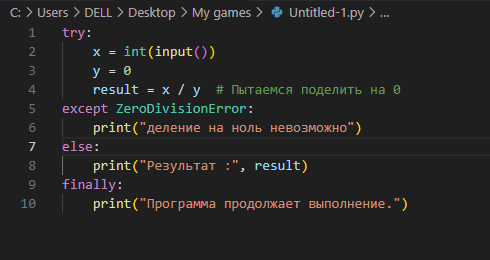
Организация обработки исключений

**Цель работы –** изучить методы и подходы к обработке исключений, научиться применять конструкции try-catch, обрабатывать и регистрировать исключения для повышения стабильности программного обеспечения

**Практическая часть**

***Задание 1: Базовая обработка исключений***

1. Создали простой модуль, в котором возникает ошибка при деление на 0

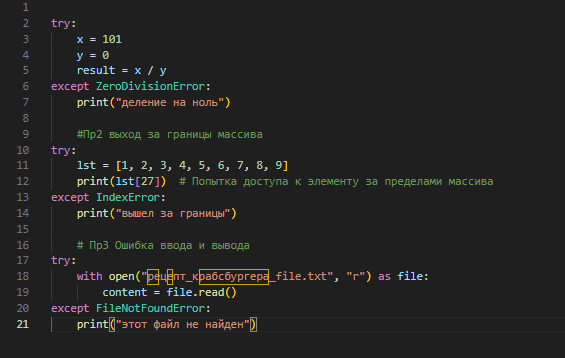


Здесь мы делаем обычную обработку исключений с помощью try-except. Программа пытается выполнить деление но 0 приводит нас к исключению ZeroDivisionError и это исключение обрабатывается и выводит сообщение об ошибке. Блок finally выполняется в любом случае и продолжается работы программы после обработки исключения

***Задание 2: Иерархия исключений***

1. Изучили как работают разные типы исключений на python

2. Создали 3 примеров обработки исключений, а именно деление на ноль, выход за границы массива и ошибки ввода-вывода  
3. Задокументировали различия между ними и поняли как обрабатывать каждую категорию



Тут мы изучили и написали как работают базовые исключения на python и создали 3 примера обработки разных исключений которые могут возникнут. Где на первом примере показано деление на ноль, на втором примере выход из массива и на третьем примере ошибка ввода и вывода

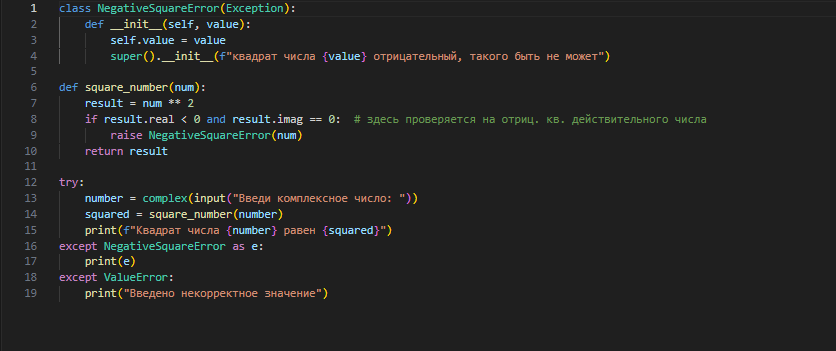
Исключение ZeroDivisionError появляется когда пытается произойти деление на 0

Исключение IndexError появляется когда программа пытается получить доступ к некоторым спискам или строки с индексом, которые находится за пределами допустимого диапазона

Исключение FileNotFoundError появляется когда программа пытается открыть файл, которого не существует по указанному пути

***Задание 3: Генерация пользовательских исключений***

1. Создали класс исключений для обработки специфических ситуаций в проекте  
2. Написали пример кода, где происходит генерация и обрабатывается исключение

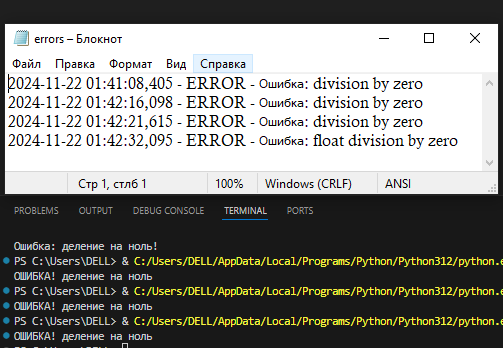
  
Этот код реализует подсчёт квадратного числа. Мы написали функцию square\_number которая принимает один аргумент «num», который может быть как действительным, так и комплексным числом. После чего «num» возводим в квадрат и проверяем, что result.real является отрицательным, а часть result.imag равна 0 и это значит что квадрат числа отрицательный, но если условие не выполняется, то в таком случае мы возращаем результат вычисления квадрата

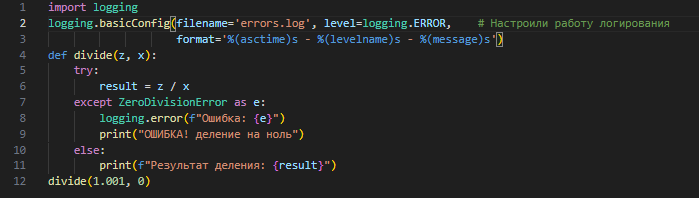


В этом примере мы видим что вводится комплексное число с положительным квадратом

***Задание 4: Логирование исключений***

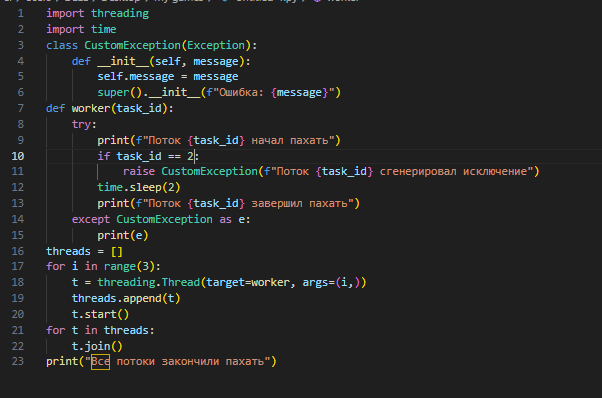
1. Реализовали простой механизм логирования исключений в файл их для анализа в будущем  
2. Проверили правильность логирования и убедитесь, что каждая ошибка записывается с указанием времени и указывает тип ошибки

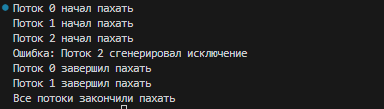
  
3. Запомнили подходы к устранению исключений на основе логов



Этот код показал нам как происходит механизм логирования исключений с использованием модуля logging. Функция divide пытается выполнить деление на ноль, что приводит к возникновению исключения ZeroDivisionError. И из-за этого в блоке это исключение логируется в файл errors.log, и выводит нам сообщение об ошибке. Как мы выяснили, логирование позволяет сохранять информацию об ошибках

***Задание 5: Обработка исключений в многопоточности***



Мы в этом коде создали новый тип ошибок под названием CustomException где мы создаем эту ошибку и передаем ей сообщение, если ошибка произойдет. Аргумент task\_id принимает некоторое число, которое говорит, какой поток будет выполняет эту функцию и сначала функция показывает что потоки начали работу. Тут task\_id равен 2, и функция генерирует пользовательскую ошибку CustomException и выводит сообщение об ошибке. Если поток не равен 2, то всё пройдёт хорошо.  


Вот какой результат выводит данный код

**Вывод:** в ходе этой работы мы изучили основные механизмы обработки исключений с помощью конструкций try-except в Python, включая создание пользовательских исключений и логирование ошибок. Мы поняли, как предотвратить аварийное завершение программы и научились правильно завершать работу программы даже при возникновении проблем в одном из потоков