

Практическая работа 8.2 - Обработчик исключений. Конструкция «try - except»

При выполнении программы могут возникнуть различного рода ошибки (исключения). Нам необходимо уметь отслеживать подобные ошибки и предотвращать их. Сегодня мы изучим конструкцию «try - except» для отлова и обработки исключений.

Что такое исключение?

Предположим, что вы разработали программу «Текстовый редактор». В программе пользователь может создать новый файл, вписать в него данные и далее сохранить файл в системе.

Если код прописан корректно, то никаких ошибок возникать не будет. Но давайте представим ситуацию, что пользователь открыл редактор, открыл нужный файл, записал в него данные, далее вручную удалил файл с компьютера и потом попытался сохранить файл через вашу программу.

При таком раскладе у вас получится ошибка, которая сломает программу и отобьет любое желание у пользователя работать в вашей программе.

Получается, исключение - это ошибка, что возникает в ходе работы самой программы. Отслеживать такие ошибки при помощи условных операторов не всегда возможно, ведь программа уже запущена, поведение пользователя нам неизвестно заранее, а значит и «ловить» ошибку нам нужно в момент её создания.

Отлов ошибок

Для вышеописанного случая как раз и нужна конструкция try except для отлова ошибок в момент их создания.

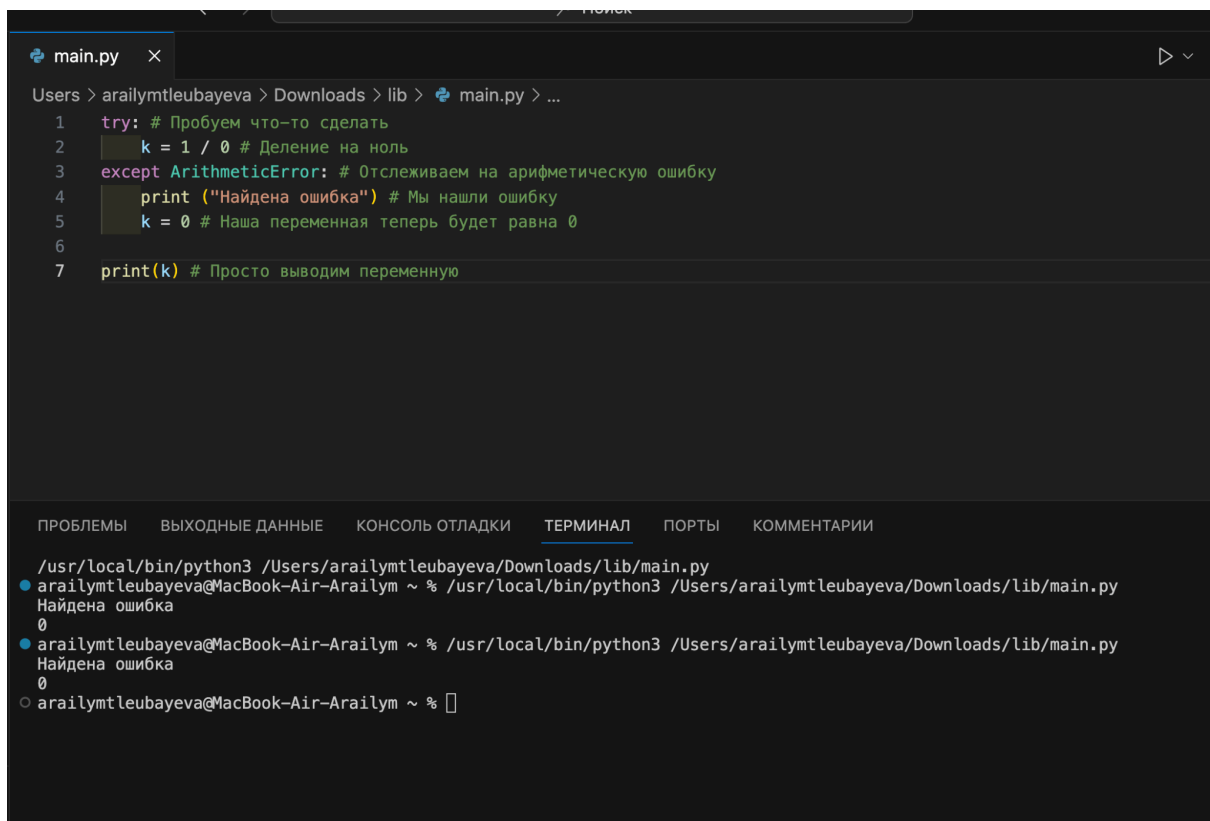
При помощи данной конструкции вы можете отслеживать ошибки различных форматов и событий. Можно отследить неверное открытие файла, можно отследить неверное подключение класса, можно отследить неверное написание переменной или функции, да чего уж там, можно отследить даже деление чисел на ноль!

Конструкция try в Python используется для обработки исключений. Основная цель — позволить разработчикам тестировать блок кода на ошибки. Вот основные причины использования try:

- **Предотвращение сбоев программы:** Если возникает ошибка в блоке кода внутри try, вместо остановки всей программы, управление передаётся блоку except. Это дает возможность обработать ошибку должным образом и продолжить выполнение программы или корректно завершить её.
- **Обработка ожидаемых исключений:** Иногда ошибки являются ожидаемой частью работы программы. Например, при попытке открыть файл, который не существует, или при делении на ноль. Конструкция try позволяет грациозно обработать такие ситуации.
- **Логирование ошибок:** Когда исключение возникает, его можно залогировать в блоке except, что может помочь в отладке программы.
- **Управление потоком программы:** Использование try и except позволяет определить чёткие пути исполнения кода в случае различных исключений.
- **Гарантирование выполнения кода:** С помощью блока finally, который следует за try, можно гарантировать выполнение определенного кода после блока try, вне зависимости от того, было ли исключение или нет.

Реализации конструкции try except

Для добавления отслеживания ошибок можно прописать следующий код:



```
main.py x
Users > arailymtleubayeva > Downloads > lib > main.py > ...
1 try: # Попробуем что-то сделать
2     k = 1 / 0 # Деление на ноль
3 except ArithmeticError: # Отслеживаем на арифметическую ошибку
4     print("Найдена ошибка") # Мы нашли ошибку
5     k = 0 # Наша переменная теперь будет равна 0
6
7 print(k) # Просто выводим переменную

ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ  КОММЕНТАРИИ

/usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
● arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
Найдена ошибка
0
● arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
Найдена ошибка
0
○ arailymtleubayeva@MacBook-Air-Arilym ~ %
```

В коде выше представлен очень простой пример использования исключений. Не обращайте внимание на его простоту, так как главное это рассмотреть общую конструкцию блока try except.

При использовании такой конструкции, какая бы ошибка не получилось в ходе программы сама программа работать не перестанет и пользователь не получит плохой опыт в её использовании.

Если необходимо отследить несколько классов с ошибками, то можно добавить несколько блоков except. Все они добавляются друг под другом точно также как в условных операторах.

Получение данных

```
← → Поиск
main.py x
Users > arailymtleubayeva > Downloads > lib > main.py > ...
1 # Инициализация переменной x значением 0
2 x = 0
3
4 # Запуск бесконечного цикла while
5 while x == 0:
6     try:
7         # Попытка получить от пользователя число через стандартный ввод
8         x = int(input("Введите число: "))
9         # Если ввод корректен (является числом), прибавляем к x число 5
10        x += 5
11        # Выводим полученное значение x
12        print(x)
13    except ValueError:
14        # Если введенные данные не могут быть преобразованы в число,
15        # выводим сообщение об ошибке и цикл начинается заново
16        print("Введите лучше число!")
17

ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ  КОММЕНТАРИИ
● arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
Введите число: 10
15
● arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
Введите число: 111
116
● arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
Введите число: /usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
Введите лучше число!
Введите число: 10
e issue  Sign in to Bitbucket  0 0 0  Select Postgres Server  Пробелов: 4  UTF-8  LF  Python  3.11.4 64-bit
```

Простое исключение

```
main.py
Users > arailymtleubayeva > Downloads > lib > main.py > ...
1  try:
2      # Запрашиваем у пользователя ввод. Ожидаем, что он введет число.
3      x = int(input("Введите число: "))
4      # Если ввод был успешно преобразован в целое число, прибавляем к нему 5
5      x += 5
6      # Выводим результат пользователю
7      print(x)
8  except ValueError:
9      # Если при попытке преобразовать ввод пользователя в целое число возникла ошибка,
10     # значит пользователь ввел не число. Выводим соответствующее сообщение
11     print("Введите лучше число!")
12
```

Несколько проверок

```
main.py
Users > arailymtleubayeva > Downloads > lib > main.py > ...
1  try:
2      # Попытка выполнить деление 5 на 1, что является корректной операцией
3      x = 5 / 1
4      # Запрос на ввод числа от пользователя
5      x = int(input())
6  except ZeroDivisionError:
7      # Этот блок выполнится, если в первой строке блока try произойдет деление на ноль
8      print("Деление на ноль!")
9  except ValueError:
10     # Этот блок выполнится, если введенное значение не удастся преобразовать в целое число
11     print("Вы ввели что-то не так")
12 else:
13     # Этот блок выполнится, если исключения не возникли
14     print("else")
15 finally:
16     # Этот блок выполнится в любом случае, вне зависимости от того, были ли исключения
17     print("Finally")
18
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ КОММЕНТАРИИ

```
● arailymtleubayeva@MacBook-Air-Arilym ~ % /usr/local/bin/python3 /Users/arailymtleubayeva/Downloads/lib/main.py
10
else
Finally
○ arailymtleubayeva@MacBook-Air-Arilym ~ %
```

Самостоятельная работа

Задание 1. Просто скажите возраст...

Создайте программу, которая будет получать возраст пользователя.

Программа должна запрашивать ввести возраст до тех пор, пока пользователь не введет число.

Задание 2. Неверный оператор

Создайте исключение, которое сработает при использовании несуществующей переменной.

Имя исключения - `NameError`.

Задание 3. Синтаксическая ошибка

Создайте исключение, которое сработает при использовании несуществующей функции.

Задание 4. Открытие несуществующего файла

Выполните открытие несуществующего файла.

Обработайте исключение `FileNotFoundError`.

Задание 5. Открытие и чтение файла

- Создайте файл «example.txt» и впишите в него слово «Привет».
- Откройте файл при помощи команды «x». Команда позволяет открыть файл для чтения, если такового файла нет.
- Выполните обработку исключения `FileExistsError`.
- В блоке `finally` пропишите закрытие файла, а в блоке `except` - верное открытие файла при помощи функции `open` с типом открытия «a».