# **Введение в обработку исключений. Блоки try / except**

Курс по Python ООП: <https://stepik.org/a/116336>

[Смотреть материал на видео](https://www.youtube.com/watch?v=MFIlltaeIgs&list=PLA0M1Bcd0w8zPwP7t-FgwONhZOHt9rz9E)

Начиная с этого занятия, мы коснемся довольно важной темы в программировании – обработка исключений.  Давайте вначале я поясню, что стоит за этими «умными» словами. Предположим, мы в программе строчка за строчкой выводим некий многострочный текст:

**print**("Я к вам пишу – чего же боле?")**print**("Что я могу еще сказать?")**print**("Теперь, я знаю, в вашей воле")**print**("Меня презреньем наказать.")**print**("Но вы, к моей несчастной доле")**print**("Хоть каплю жалости храня,")**print**("Вы не оставите меня.")

И где-нибудь в серединке, после третьего print(), запишем команду:

**print**(a)

Разумеется, в момент выполнения этой программы, возникнет ошибка, а точнее – исключение с типом:

NameError

Причем, первые три строчки будут выполнены, то есть, до момента исключения интерпретатор Python выполняет программу строчка за строчкой. Это пример исключения в момент выполнения программы. Причем, типы исключений могут быть самыми разными, в зависимости от возникшей ошибки. Например, если попробовать выполнить деление на ноль:

10/0

то отобразится тип исключения:

ZeroDivisionError

которое также относится к моменту выполнения программы, так как первые три строчки (до ошибки) были успешно выполнены.

Другой вариант ошибок, это, например, неверный синтаксис в программе. Если, допустим, прописать строчку:

**if** True     **print**("True")

то увидим исключение типа:

SyntaxError

которое возникает в момент компиляции текста программы, поэтому, ни одной строчки не было отображено в консоли. То есть, исключения можно разбить на две группы:

* исключения в момент исполнения;
* исключения при компиляции (до исполнения кода).

И при обработке исключений нас будет интересовать первая группа – момента исполнения, так как именно из-за них исполнение программы может внезапно остановиться и, как вы понимаете, это не лучший сценарий развития.

На первый взгляд может показаться, что исключения возникают по вине программиста, из-за того, что он что-то не учел. Иногда такое бывает, но не всегда. Классический пример обработки исключений момента выполнения программы – открытие файлов на чтение или запись. Мы с вами уже об этом говорили, когда открывается файл:

file = open("myfile.txt")

то программист не может заранее знать, есть ли такой файл на диске или нет. Поэтому, такую строчку нужно выполнять в специальном блоке try с обработкой исключения типа FileNotFoundError:

**try**:     file = open("myfile2.txt")**except** FileNotFoundError:     **print**("Невозможно открыть файл")

Или, при делении одного числа на другое:

**try**:     x, y = map(int, input().split())     res = x / y **except** ZeroDivisionError:     **print**("Делить на ноль нельзя!")

Здесь заранее нельзя сказать чему будет равно значение y, поэтому деление лучше поместить в блок try с обработкой исключения ZeroDivisionError. Кстати, здесь же может появиться и другое исключение ValueError, если мы введем с клавиатуры не целые числа, а, например, строки.

Чтобы отловить и это второе исключение, можно прописать еще один блок except:

**try**:     x, y = map(int, input().split())     res = x / y **except** ZeroDivisionError:     **print**("Делить на ноль нельзя!")**except** ValueError:     **print**("Ошибка типа данных")

И таких блоков except может быть сколько угодно.

Чтобы отловить в блоке except сразу несколько типов исключений, их можно указать в круглых скобках через запятую после ключевого слова except:

**except** (ZeroDivisionError, ValueError):     res = "деление на ноль или нечисловое значение"

Но все же чаще используют раздельную запись, поэтому я ее верну.

Если мы хотим при возникновении ошибок взаимодействовать с объектами классов исключений, то это делается так:

**try**:     x, y = map(int, input().split())     res = x / y **except** ZeroDivisionError **as** z:     **print**(z)**except** ValueError **as** z:     **print**(z)

То есть, после имени исключения ставится ключевое слово as и дальше переменная, которая будет ссылаться на объект класса ValueError, в котором хранится служебная информация о конкретной ошибке.

Разумеется, так обрабатывать можно только исключения на этапе исполнения. Про синтаксические ошибки мы сейчас не говорим. Так вот, существует определенная иерархия классов исключений, во главе которой стоит базовый класс BaseException:

Здесь приведен лишь ее фрагмент. Самих классов гораздо больше. Из этого рисунка хорошо видно, что большая часть классов наследуется от базового класса Exception и именно эта ветка нас будет интересовать.

На что влияет эта иерархия и как ее можно использовать при обработке исключений? Например, мы видим обработку исключения деления на ноль, которое относится к базовому классу ArithmeticError, а он, в свою очередь, к базовому классу Exception. Поэтому, вместо указания конкретного типа ZeroDivisionError можно указать любой из этих базовых классов:

**try**:     x, y = map(int, input().split())     res = x / y **except** ArithmeticError:     **print**("Делить на ноль нельзя!")**except** ValueError:     **print**("Ошибка типа данных")

Но здесь есть один важный нюанс. Если вместо ArithmeticError прописать класс Exception – общий для классов ArithmeticError и ValueError, то первым блоком exception будут отлавливаться все типовые исключения. А блок с ValueError никогда выполнен не будет. Поэтому здесь есть одно простое правило. Сначала прописываются блоки со специализированными классами исключений, а затем с более общими (базовыми).

Если поменяем местами наши блоки except:

**try**:     x, y = map(int, input().split())     res = x / y **except** ValueError:     **print**("Ошибка типа данных")**except** Exception:     **print**("Делить на ноль нельзя!")

То ValueError будет отлавливать свой тип исключения, а второй блок – все остальные, унаследованные от класса Exception.

И последнее, что я хочу отметить про блок except. Его можно прописывать без указания каких-либо классов, следующим образом:

**try**:     x, y = map(int, input().split())     res = x / y **except**:     **print**("Ошибка")

В этом случае он будет отлавливать все исключения, возникающие в блоке try.

На этом завершим первое знакомство с исключениями. Здесь я вас лишь хотел познакомить с блоками try и except, а также показать на что влияет иерархия классов исключений при их отслеживании.

Курс по Python ООП: <https://stepik.org/a/116336>