Python exceptions


Я НЕ УВЕРЕН, ЧТО ОНИ ПОНИМАЮТ, КАК ЭТИМ ПОЛЬЗОВАТЬСЯ...
НО ЭТО ЖЕ ТАК ОЧЕВИДНО...
ПОЛЬЗОВАТЕЛИ
MONKEYUSER.COM

Исключения (exceptions) – ещё один тип данных в python. Исключения необходимы для того, чтобы сообщать программисту об ошибках.

Самый простейший пример исключения – деление на ноль:

```
a = int(input("Введите делимое: "))
b = int(input("Введите делитель: "))
result = a / b
print(result
```

```
>>Введите делимое: 3
>>Введите делитель: 0
>>ZeroDivisionError: division by zero
```

```
a = int(input("Введите делимое: "))
b = int(input("Введите делитель: "))

try:
   result = a / b
   print(result)
except ZeroDivisionError:
      print("You cannot divide by zero!")
```

```
>>Введите делимое: 3
>>Введите делитель: 0
>>You cannot divide by zero!
```

```
a = int(input("Введите делимое: "))
b = int(input("Введите делитель: "))

try:
   result = a / b
   print(result)
except ZeroDivisionError:
      print("You cannot divide by zero!")
```

```
>>Введите делимое: 6
>>Введите делитель: 2
>>3.0
```

!!! *часть, которая может вызвать исключение, помещается внутри блока try*

голое исключение

Есть еще один способ поймать ошибку, но так делать не стоит. Причина заключается в том, что вы не узнаете, что именно за исключение вы выловите. Когда у вас возникло что-то в духе* ZeroDivisionError, *вы хотите выявить фрагмент, в котором происходит деление на ноль. В коде, написанном ниже, вы не можете указать, что именно вам нужно выявить.

```
a = input("Введите делимое: ")
b = input("Введите делитель: ")

result = int(a) / int(b)
print(result)
```

>>Введите делимое: 3
>>Введите делитель: 0
>>ZeroDivisionError: division by zero

```
a = input("Введите делимое: ")
b = input("Введите делитель: ")

try:
    result = int(a) / int(b)
    print(result)
except:
    print("You cannot divide by zero!")
```

>>Введите делимое: 3
>>Введите делитель: 0
>>You cannot divide by zero!

```
a = input("Введите делимое: ")
b = input("Введите делитель: ")

try:
    result = int(a) / int(b)
    print(result)
except:
    print("You cannot divide by zero!")
```

>>Введите делимое: 6
*>>Введите делитель: **h***
>>You cannot divide by zero!

Класс BaseException — это базовый класс всех исключений.

Он имеет четыре подкласса.

1.Исключение - это базовый класс для всех исключений, не связанных с выходом.
2.GeneratorExit - запрос на выход генератора.
3.KeyboardInterrupt - программа, прерванная пользователем.
4.SystemExit - запрос на выход из интерпретатора.

Встроенные классы исключений

ArithmeticError — это базовый класс для арифметических ошибок.
AssertionError — возникает, когда утверждение не выполняется.
AttributeError — когда атрибут не найден.
EOFError — чтение после конца файла
ImportError — когда импортированный модуль не найден.
LookupError — базовое исключение для ошибок поиска.
MemoryError — при нехватке памяти
NameError — когда имя не найдено глобально.
OSError — базовый класс для ошибок ввода-вывода
StopIteration, StopAsyncIteration
SyntaxError — недопустимый синтаксис
SystemError — внутренняя ошибка интерпретатора Python.
TypeError — недопустимый тип аргумента
ValueError — недопустимое значение аргумента

Встроенные классы предупреждений

BytesWarning — предупреждения, связанные с байтами и буфером, в основном связанные с преобразованием и сравнением строк.
DeprecationWarning — предупреждение об устаревших функциях
FutureWarning — базовый класс для предупреждения о конструкциях, которые будут семантически изменяться в будущем.
ImportWarning — предупреждение об ошибках при импорте модулей
PendingDeprecationWarning — предупреждение о функциях, поддержка которых в будущем будет прекращена.
ResourceWarning — предупреждения об использовании ресурсов
RuntimeWarning — предупреждения о сомнительном поведении во время выполнения.
SyntaxWarning — предупреждение о сомнительном синтаксисе
UnicodeWarning — предупреждения, связанные с преобразованием Unicode
UserWarning — предупреждения, генерируемые кодом пользователя

обработка нескольких ошибок за раз

```
a = input('Введите делимое: ')
b = input('Введите делитель: ')
try:
    print(int(a) / int(b))
except ZeroDivisionError:
    print("You cannot divide by zero!")
except ValueError:
    print("Strings cannot be divided!")
except:
    print("Something was wrong!")
```

Задачи

11.1

Решить задачу с обработкой исключений. Обработка должен сообщать, что пользователь ввел не число

Задача 4.4:
Запросить у пользователя возраст. Если ему 18 и больше, напечатать beer. Меньше 18 — Coca Cola

In: f
Out: Вы ввели символ, а надо число

11.2

Решить задачу с обработкой исключений. Создайте ошибку NameError и выведите сообщение о ней.

Задача 5.12
Посчитайте сумму квадратов всех чисел, находящихся в интервале a, b и выведите ее на экран.

In: 2
3
Out: Переменная не найдена

11.3

Решить задачу с обработкой исключений. Вызвать ошибку IndexError и указывать пользователю последний индекс, чтобы он понимал до какого индекса он может запрашивать информацию

Задача
Пользователь вводит числа или буквы и еще одну цифру. Эта цифра будет индексом, по которому необходимо заменить элемент в получившемся списке.

In: 2 6 4
7
Out: Такого индекса нет, в списке всего 3 элементов

finally

После последнего блока except можно добавить блок finally. Он исполняет инструкции при любых условиях.

```
f = open('1.txt')
ints = []

try:
    for line in f:
        ints.append(int(line))
    except ValueError:
        print('Это не число. Выходим.')
    except Exception:
        print('Это что ещё такое?')
    finally:
        f.close()
        print('Я закрыл файл.')
```

Бывают ситуации, когда некие важные действия необходимо совершить вне зависимости от того, было вызвано исключение или нет. Для этого используется блок finally, содержащий набор инструкций, которые нужно выполнить в любом случае.

else

В некоторых ситуациях вам может потребоваться запустить определенный блок кода, если внутри try выполняется без ошибок.

В этих случаях вы можете использовать необязательное ключевое слово else с оператором try.

Примечание: Исключения в предложении else не обрабатываются предыдущими предложениями except

```
a = int(input('Введите делимое: '))
b = int(input('Введите делитель: '))
try:
    result = a / b
except ZeroDivisionError:
    print("You cannot divide by zero!")
else:
    print(result)
```

```
>>Введите делимое: 6
>>Введите делитель: 3
>>2.0
```

Задачи

11.4

Решить задачу с обработкой исключений. Вызвать ошибку ValueError и сообщить, что пользователь ввел не число:

Задача hometask_4_3:
Написать калькулятор. Пользователь вводит 2 числа и одно из мат действий. В зависимости от действия, вывести результат.

In: 4
f
+
Out: Вы ввели не число!

11.5

Переделать предыдущую задачу с калькулятором.
Вызвать ошибку TypeError и в результате вывести сообщение об ошибке.

In:2
2
+
Out: Математические операции не поддерживаются между строкой и числом

11.6

Решить задачу с обработкой исключений. Вызвать ошибку KeyError и вернуть программу к исполнению команд. Укажите пользователю какое вино он может выбрать.

Немного изменим задачу 8.9:
Катя с Анной пьют вино. Программа принимает сколько бутылок вина выпьет каждая. Еще программа принимает какое вино пьет Катя, а Анна всегда пьет белое.

In: 2
5
pink wine
Out: Есть только ['red wine', 'white wine'], выберите из них

использование исключений

Как правило, один и тот же код можно написать и с использованием исключений, и без них.

```
a = input("Введите число: ")
b = input("Введите второе число: ")

try:
    result = int(a)/int(b)
except ValueError:
    print("Поддерживаются только числа")
except ZeroDivisionError:
    print("На ноль делить нельзя")
else:
    print(result)
```

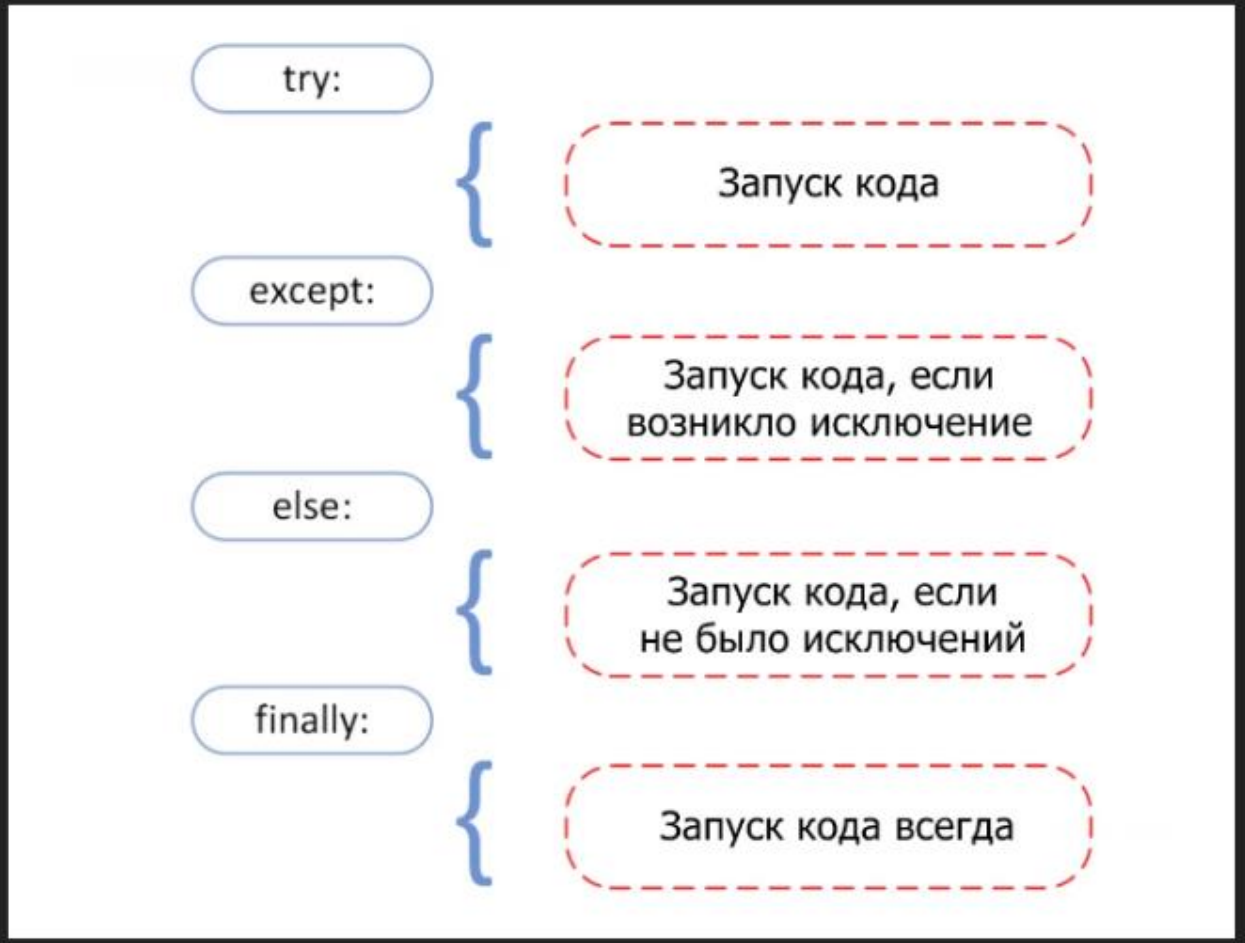
```
a = input("Введите число: ")
b = input("Введите второе число: ")

if a.isdigit() and b.isdigit():
    if int(b) == 0:
        print("На ноль делить нельзя")
    else:
        print(int(a) / int(b))
else:
    print("Поддерживаются только числа")
```

Далеко не всегда аналогичный вариант без использования исключений будет простым и понятным.

Важно в каждой конкретной ситуации оценивать, какой вариант кода более понятный, компактный и универсальный - с исключениями или без.

шпаргалка


try:
Запуск кода
except:
Запуск кода, если возникло исключение
else:
Запуск кода, если не было исключений
finally:
Запуск кода всегда

домашка

ПЕРЕДЕЛАЙТИ 3 ЛЮБЫЕ ЗАДАЧИ НА ВАШ ВЫБОР, ЧТОБЫ ОНИ БЫЛИ С ОБРАБОТКОЙ ОШИБОК.