GeekBrains

Погружение в Python

Урок 13
Исключения

Содержание урока

План курса

1. Основы Python
2. Простые типы данных
3. Коллекции
4. Функции
5. Итераторы и генераторы
6. Модули
7. Файлы и файловая система
8. Сериализация
9. Декораторы
10. ООП. Начало
11. ООП. Особенности Python
12. ООП. Финал
13. Исключения
14. Основы тестирования
15. Обзор стандартной библиотеки Python

Что будет на уроке сегодня

- Разберёмся с обработкой ошибок в Python
- Изучим иерархию встроенных исключений
- Узнаем о способе принудительного поднятия исключения в коде
- Разберёмся в создании собственных исключений

Обработка исключительных ситуаций в Python

Ошибки без обработки прерывают выполнение кода

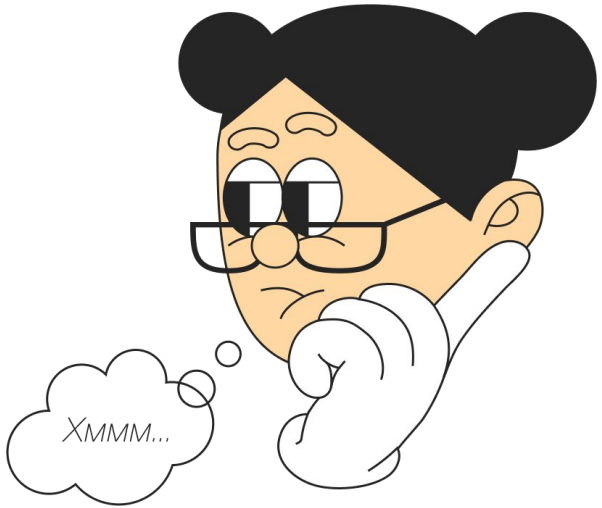
Python имеет отличную систему информирования разработчика об ошибках.

```
Введите целое число: сорок два
Traceback (most recent call last):
  File "C:\Users\main.py", line 1, in <module>
    num = int(input('Введите целое число: '))
ValueError: invalid literal for int() with base 10: 'сорок два'

Process finished with exit code 1
```

Команда try

Для перехвата ошибок создают блоки try

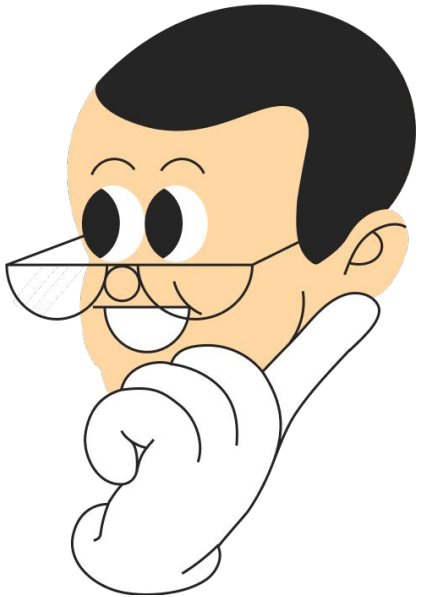
```
    ...
    try:
        # отлавливаем ошибки
        ...

    # продолжаем нормальную работу
    ...
```



Команда except

except — обязательный блок кода после try.
Отвечает за обработку исключения.

```
    ...
    try:
        # отлавливаем ошибки
        ...
    except NameError as e:
        # действия при перехвате ошибки NameError
        ...
    # продолжаем нормальную работу
    ...
```

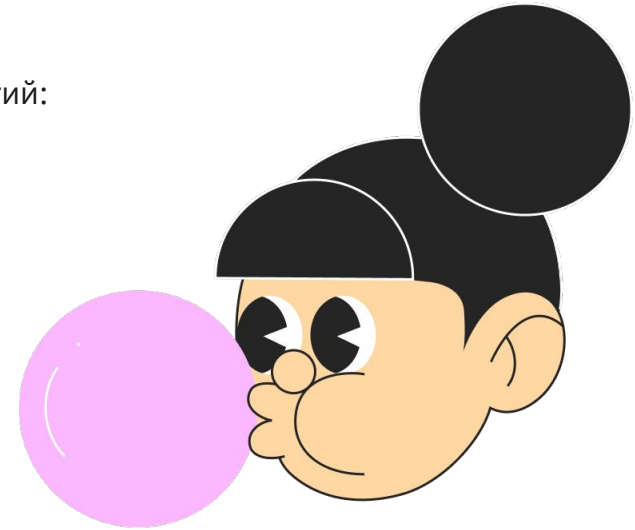


Команда else

Дополнительные действия можно указать в блоке else.

Блок else **не сработает**, если внутри try произошло любое из событий:

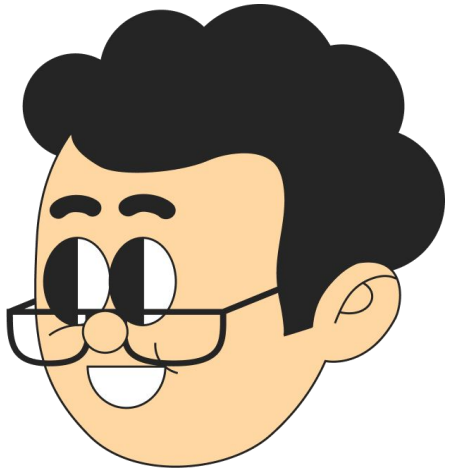
- 💡 возбуждено исключение
- 💡 выполнена команда return
- 💡 выполнена команда break
- 💡 выполнена команда continue



Команда finally

Блок finally выполняется в любом случае

```
    ...
    try:
        # отлавливаем ошибки
        ...
    except NameError as e:
        # действия при перехвате ошибки NameError
        ...
    finally:
        # выполняется в любом случае
        ...
    # продолжаем нормальную работу
    ...
```

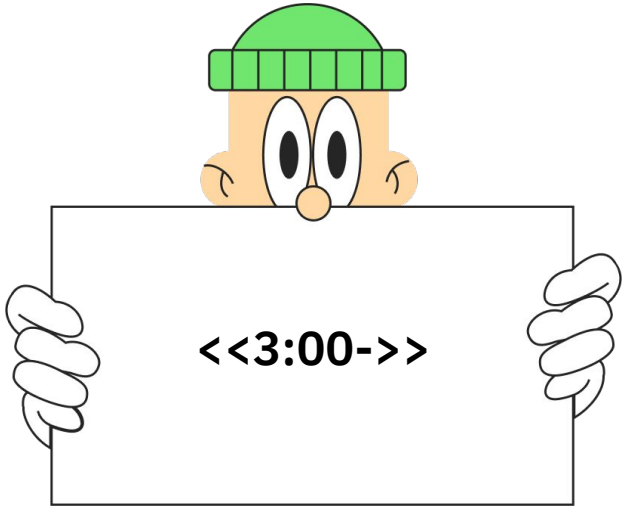


Перед вами несколько строк кода.
Напишите в чат что они вернут
не запуская программу.

У вас 3 минуты.

Обработка исключительных ситуаций

```
d = {'42': 73}
try:
    key, value = input('Ключ и значение: ').split()
    if d[key] == value:
        r = 'Совпадение'
except ValueError as e:
    print(e)
except KeyError as e:
    print(e)
else:
    print(r)
finally:
    print(d)
```

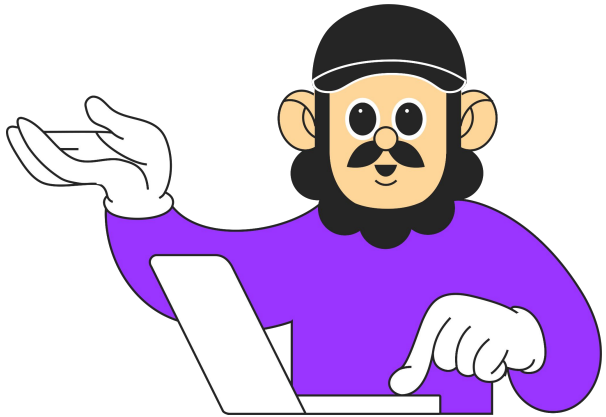



```
>>> Ключ и значение: 42 13
>>> Ключ и значение: 42 73 3
>>> Ключ и значение: 73 42
>>> Ключ и значение: 42 73
```

Иерархия исключений в Python

Иерархия исключений

```
BaseException
 ├── BaseExceptionGroup
 ├── GeneratorExit
 ├── KeyboardInterrupt
 ├── SystemExit
 └── Exception
      |
```

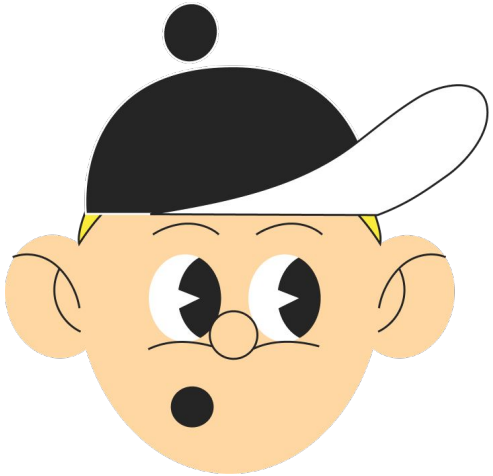


Иерархия исключений

```
└── Exception
    ├── ArithmeticError
    │   ├── FloatingPointError
    │   ├── OverflowError
    │   └── ZeroDivisionError
    ├── AssertionError
    ├── AttributeError
    ├── BufferError
    ├── EOFError
    ├── ExceptionGroup [BaseExceptionGroup]
    ├── ImportError
    │   └── ModuleNotFoundError
    ├── LookupError
    │   ├── IndexError
    │   └── KeyError
    ├── MemoryError
    ├── NameError
    │   └── UnboundLocalError
```

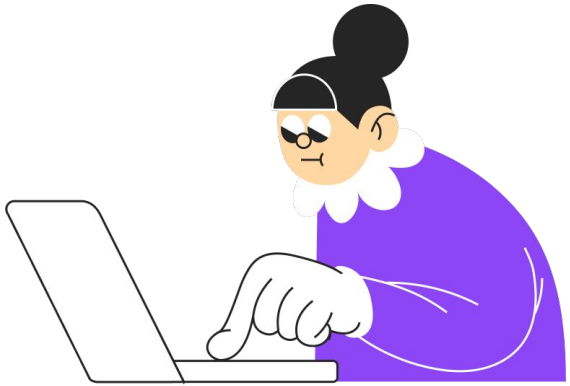
Иерархия исключений

```
└── Exception
    ├── OSError
    │   ├── BlockingIOError
    │   ├── ChildProcessError
    │   ├── ConnectionError
    │   │   ├── BrokenPipeError
    │   │   ├── ConnectionAbortedError
    │   │   ├── ConnectionRefusedError
    │   │   └── ConnectionResetError
    │   ├── FileExistsError
    │   ├── FileNotFoundError
    │   ├── InterruptedError
    │   ├── IsADirectoryError
    │   ├── NotADirectoryError
    │   ├── PermissionError
    │   ├── ProcessLookupError
    │   └── TimeoutError
```



Иерархия исключений

```
└── Exception
    ├── ReferenceError
    ├── RuntimeError
    │   ├── NotImplementedError
    │   └── RecursionError
    ├── StopAsyncIteration
    ├── StopIteration
    ├── SyntaxError
    │   └── IndentationError
    │       └── TabError
    ├── SystemError
    ├── TypeError
    ├── ValueError
    │   └── UnicodeError
    │       ├── UnicodeDecodeError
    │       ├── UnicodeEncodeError
    │       └── UnicodeTranslateError
```



Иерархия исключений

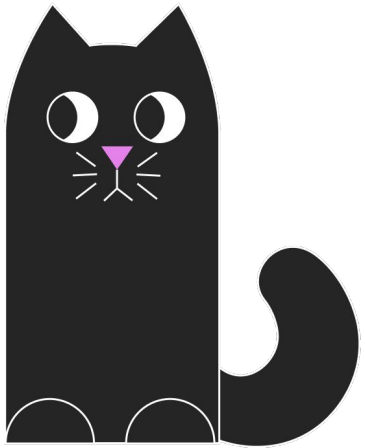
```
└── Exception
    └── Warning
        ├── BytesWarning
        ├── DeprecationWarning
        ├── EncodingWarning
        ├── FutureWarning
        ├── ImportWarning
        ├── PendingDeprecationWarning
        ├── ResourceWarning
        ├── RuntimeWarning
        ├── SyntaxWarning
        ├── UnicodeWarning
        └── UserWarning
```

Ключевое слово raise

Кот Д может самостоятельно поднимать исключения

Команда raise позволяет программисту
принудительно вызвать указанное исключение

- после raise необходимо указать класс исключения
- класс может принять поясняющий текст исключения в качестве аргумента



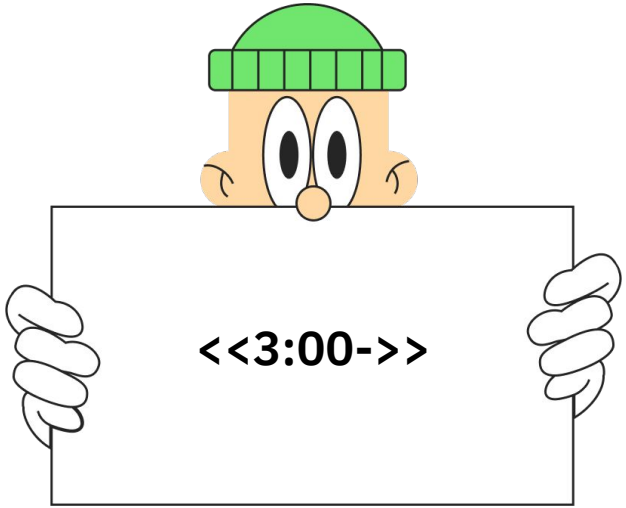
Перед вами несколько строк кода.
Напишите в чат что они вернут
не запуская программу.

У вас 3 минуты.

Ключевое слово raise

```
def func(a, b, c):
    if a < b < c:
        raise ValueError('Не перемешано!')
    elif sum((a, b, c)) == 42:
        raise NameError('Это имя занято!')
    elif max(a, b, c, key=len) < 5:
        raise MemoryError('Слишком глуп!')
    else:
        raise RuntimeError('Что-то не так!!!')
```

```
func(11, 7, 3)   # 1
func(3, 2, 3)   # 2
func(73, -40, 9)   # 3
func(10, 20, 30)   # 4
```

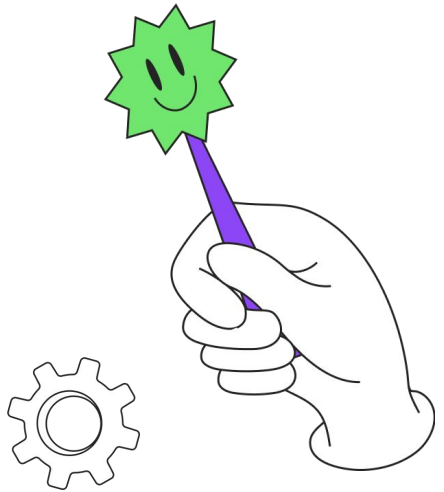


Создание собственных исключений

Создание собственных исключений

Разработчик может создавать свои классы исключения

- Исключение не должно дублировать стандартное исключение
- Собственные исключения наследуются от класса Exception
    - Если исключений несколько, создают собственное родительское исключение от Exception. Остальные исключения наследуются от него.
- Для исключения достаточно определить два дандер метода:
    - __init__
    - __str__



Итоги занятия

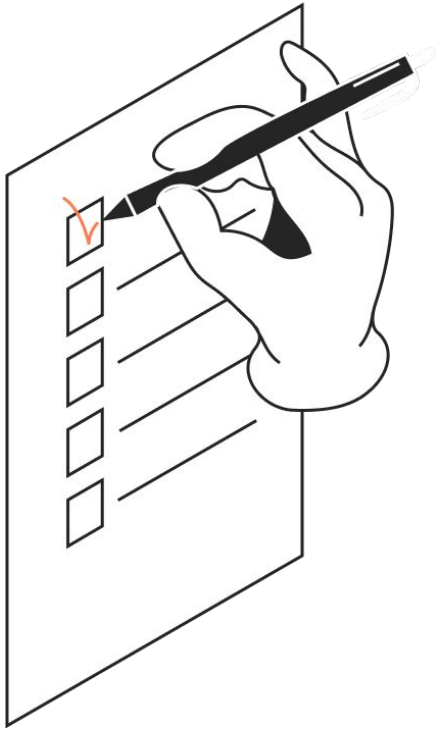
На этой лекции мы

- Разобрались с обработкой ошибок в Python
- Изучили иерархию встроенных исключений
- Узнали о способе принудительного поднятия исключения в коде
- Разобрались в создании собственных исключений

Задание

Возьмите 1-3 задачи из прошлых занятий. Добавьте обработку исключений так, чтобы код продолжал работать и выполнял задачу.

*Для любителей сложностей создайте собственные исключения для каждой из задач.



Спасибо за внимание