Исключения в Python • генерация исключений • типы исключений • обработка исключений Типы исключений • исключения стандартной библиотеки Python • пользовательские исключения Иерархия исключений In [ ]: BaseException +-- SystemExit +-- KeyboardInterrupt +-- GeneratorExit +-- Exception +-- StopIteration +-- AssertionError +-- AttributeError +-- LookupError +-- IndexError +-- KeyError +-- OSError +-- SystemError +-- TypeError +-- ValueError Обработка исключений 1 / 0 except: print("Ошибка") Обработка исключений In [ ]: | try: 1 / 0 except Exception: print("Ошибка") Обработка ожидаемого исключения while True: try: raw = input("введите число: ") number = int(raw) break except: print("некорректное значение") Обработка исключения ValueError while True: try: raw = input("введите число: ") number = int(raw) break except ValueError: print("некорректное значение") Блок else while True: try: raw = input("введите число: ") number = int(raw) except ValueError: print("некорректное значение!") else: break Обработка нескольких исключений while True: try: raw = input("введите число: ") number = int(raw) break except ValueError: print("некорректное значение!") except KeyboardInterrupt: print("выход") break Обработка нескольких исключений In [ ]: | total\_count = 100\_000 while True: try: raw = input("введите число: ") number = int(raw) total\_count = total\_count / number break except (ValueError, ZeroDivisionError): print("некорректное значение!") Обработка нескольких исключений, наследование # +-- LookupError +-- IndexError +-- KeyError Python 3.6.1 [GCC 5.4.0 20160609] on linux Type "help", "copyright", "credits" or "license" for more information. >>> issubclass(KeyError, LookupError) True >>> issubclass(IndexError, LookupError) True >>> Обработка нескольких исключений, наследование database = { "red": ["fox", "flower"], "green": ["peace", "M", "python"] try: color = input("введите цвет: ") number = input("введите номер по порядку: ") label = database[color][int(number)] print("вы выбрали:", label) # except (IndexError, KeyError): except LookupError: print("Объект не найден") Блок finally In [ ]: f = open("/etc/hosts") try: for line in f: print(line.rstrip("\n")) 1 / 0 f.close() except OSError: print("ошибка") Блок finally In [ ]: f = open("/etc/hosts") try: for line in f: print(line.rstrip("\n")) 1 / 0 except OSError: print("ошибка") finally: f.close() Исключения, подводим итоги • генерация исключений • типы исключений • обработка исключений Исключения в Python (Часть 2) • доступ к объекту исключения • генерация исключений, инструкция raise • исключения типа AssertionError • вопросы производительности • работа с собственными исключениями Доступ к объекту исключения In [ ]: try: with open("/file/not/found") as f: content = f.read() except OSError as err: print(err.errno, err.strerror) Доступ к объекту исключения, атрибут args # ampuбут args import os.path filename = "/file/not/found" try: if not os.path.exists(filename): raise ValueError("файл не существует", filename) except ValueError as err: message, filename = err.args[0], err.args[1] print(message, code) Доступ к стеку вызовов In [ ]: import traceback try: with open("/file/not/found") as f: content = f.read() except OSError as err: trace = traceback.print\_exc() print(trace) Генерация исключения, инструкция raise In [ ]: try: raw = input("введите число: ") if not raw.isdigit(): raise ValueError except ValueError: print("некорректное значение!") Инструкция raise для экземпляра ValueError In [ ]: try: raw = input("введите число: ") if not raw.isdigit(): raise ValueError("плохое число", raw) except ValueError as err: print("некорректное значение!", err) Проброс исключения "выше" In [ ]: | try: raw = input("введите число: ") if not raw.isdigit(): raise ValueError("плохое число", raw) except ValueError as err: print("некорректное значение!", err) # делегирование обработки исключения raise Исключение через raise from Exception In [ ]: try: raw = input("введите число: ") if not raw.isdigit(): raise ValueError("плохое число", raw) except ValueError as err: print("ошибка:", err.args[0], err.args[1]) raise TypeError("ошибка") from err Инструкция assert Python 3.6.1 [GCC 5.4.0 20160609] on linux Type "help", "copyright", "credits" or "license" for more information. >>> assert True >>> **assert** 1 == 0 Traceback (most recent call last): File "<stdin>", line 1, in <module> AssertionError >>> >>> assert 1 == 0, "1 не равен 0" Traceback (most recent call last): File "<stdin>", line 1, in <module> AssertionError: 1 не равен 0 Инструкция assert, флаг -O def get\_user\_by\_id(id): assert isinstance(id, int), "id должен быть целым числом" print("выполняем поиск") **if** \_\_name\_\_ == "\_\_main\_\_": get\_user\_by\_id("foo") Производительность исключений my\_dict = {"foo": 1} for \_ in range(1000): try: my\_dict["bar"] except KeyError: pass 1000 loops, best of 3: 511 μs per loop my\_dict = {"foo": 1} for \_ in range(1000): if "bar" in my\_dict: \_ = my\_dict["bar"] 10000 loops, best of 3: 78.3 μs per loop Работа с собственными исключениями, библиотека requests In [ ]: response = requests.get("https://github-not-found.com") In [ ]: try: response = requests.get("https://github-not-found.com") except requests.RequestException as err: print(err) >>> HTTPSConnectionPool(host='github-not-found.com', port=443): Max retries exceeded with url: / >>> (Caused by NewConnectionError( >>> Failed to establish a new connection: [Errno -2] >>> Name or service not known',)) >>> # requests.\_\_file\_\_ import requests import time timeout = 0.2 for \_ in range(5): try: response = requests.get("https://github.com/not\_found", timeout=timeout) response.raise\_for\_status() break except requests.Timeout: print("попробуйте позже timeout:", timeout) timeout \*= 2 time.sleep(timeout) except requests.HTTPError as err: print(err.response.status\_code) raise