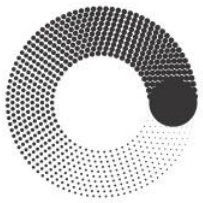
федеральное государственное автономное образовательное учреждение высшего образования



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
(ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ)
(Факультет информационных технологий)

(Институт Принтмедиа и информационных технологий) Кафедра Информатики и информационных технологий

направление подготовки 09.03.02
«Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 4

Дисциплина: Backend-разработка (Часть 2 - Python).

Тема: Исключения.

Выполнил(а): студент(ка) группы 221-3711

Мироненко Р. Е.

(Фамилия И.О.)

Дата, подпись 13.11.2024 _____

(Дата) (Подпись)

Проверил: _____ _____

(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____

(Дата) (Подпись)

Замечания: _____
_____
_____

Москва 2024

```python
'''Точка входа'''

from functions import (
    divide, root, calculate, load_file,
    is_palindrome, factorial, check_age,
    validate_levels, validate_email,
    validate_phone_number, insert,
    add_unique, convert_to_int, can_convert_to_int)


def run():
    '''Запуск вссех функций'''
    try:
        print("Шаг 1.1:")
        print(divide(10, 2))
        print(divide(10, 0))
    except ValueError as e:
        print(f"Необработанная ошибка: {e}")

    try:
        print("\nШаг 1.2")
        print(root(16))
        print(root(-4))
    except ValueError as e:
        print(f"Необработанная ошибка: {e}")

    try:
        print("\nШаг 2")
        print(calculate(10, 2))
        print(calculate(10, 0))
        print(calculate(-10, 2))

        print("\nШаг 3")
        load_file("file1.txt", lambda res: print(f"Результат загрузки: {res}"))
        load_file("file2", lambda res: print(f"Результат загрузки: {res}"))

        print("\nШаг 4.1")
        print(is_palindrome("Aboba"))
        print(is_palindrome(""))
        print(is_palindrome(123))

        print("\nШаг 4.2")
        print(factorial(5))
        print(factorial(-1))
        print(factorial(5.5))
        print(factorial(21))

        print("\nШаг 4.3")
        check_age(30)
        check_age(-5)
```

```
        check_age(200)

        print("\nШаг 5")
        validate_levels([1, 2, 3])
        validate_levels([])
        validate_levels([1, -2, 3])

        print("\nШаг 6.1 (7)")
        validate_email("rmiro@gmail.com")
        validate_email("rmiro.com")

        print("\nШаг 6.2 (7)")
        validate_phone_number("1234567890")
        validate_phone_number("12345")

        print("\nШаг 6.3 (7)")
        my_list = [1, 2, 3]
        insert(my_list, 4, 1)
        print(my_list)
        insert(my_list, 5, 5)

        print("\nШаг 8.1")
        my_list = [1, 2, 3]
        add_unique(my_list, 4)
        print(my_list)
        add_unique(my_list, 4)

        print("\nШаг 8.2")
        print(convert_to_int("1"))
        print(convert_to_int("a"))

        print("\nШаг 8.3")
        print(can_convert_to_int("1"))
        print(can_convert_to_int("a"))

    except Exception as e:
        print(f"Необработанная ошибка: {e}")


if __name__ == "__main__":
    run()
```

```
''' Функции '''

from exceptions import InvalidEmailError, InvalidPhoneNumberError, OutOfRangeError


# Шаг 1.1
def divide(a, b):
    '''Деление двух чисел'''
    if b == 0:
```

```
        raise ValueError("Ошибка деления на 0")
    return a / b


# Шаг 1.2
def root(x):
    '''Извлечение корня'''
    if x < 0:
        raise ValueError("Невозможно извлечь корень из отрицательного числа")
    return x ** 0.5


# Шаг 2
def calculate(a, b):
    '''Возвращает корень деления a на b'''
    try:
        result_divide = divide(a, b)
        result_sqrt = root(result_divide)
        return result_sqrt

    except Exception as e: # Жалуется линтер тк слишком обобщённое исключение
        return f"Ошибка: {e}"


# Шаг 3
def load_file(name, callback):
    '''
    Загружает файл
    Выбрасывает исключения при ошибке загрузки
    '''

    print(f"Файл {name} открыт")

    try:
        if len(name) <= 3 or "." not in name:
            raise UnicodeError("Ошибка загрузки")
        print(f"Файл {name} загружен")
        callback(True)

    except Exception as e:
        print(f"Ошибка: {e}")
        callback(False)
    finally:
        print(f"Файл {name} закрыт")


# Шаг 4.1
def is_palindrome(s):
    '''
    Проверка, является ли строка палиндромом
    Выбрасывает исключения при неверных входных параметрах
```

```
    '''
    try:
        if not isinstance(s, str):
            raise TypeError("Входное значение должно быть строкой")
        if len(s) == 0:
            raise ValueError("Строка не должна быть пустой")

        cleaned = ''.join(s.split()).lower()
        return cleaned == cleaned[::-1]

    except TypeError as e:
        print(f"Ошибка: {e}")
    except ValueError as e:
        print(f"Ошибка: {e}")
    except Exception as e:
        print(f"Ошибка: {e}")
    return False


# Шаг 4.2
def factorial(n):
    '''
    Факториала числа
    Выбрасывает исключения при отрицательных значениях или если входной параметр не целое число
    '''
    try:
        if not isinstance(n, int):
            raise TypeError("Входное значение должно быть целым числом")
        if n < 0:
            raise ValueError("Входное значение должно быть положительным")
        if n > 20:
            raise OverflowError("Слишком большое число")

        result = 1
        for i in range(2, n + 1):
            result *= i
        return result

    except ValueError as e:
        print(f"Ошибка: {e}")
    except TypeError as e:
        print(f"Ошибка: {e}")
    except OverflowError as e:
        print(f"Ошибка: {e}")
    except Exception as e:
        print(f"Ошибка: {e}")
    return 0


# Шаг 4.3
def check_age(age):
```

```
    '''
    Функция для проверки возраста
    Выбрасывает исключения при отрицательном или слишком большом возрасте
    '''
    try:
        if not isinstance(age, int):
            raise TypeError("Возраст должен быть целым числом")
        if age < 0:
            raise ValueError("Возраст не может быть отрицательным")
        if age > 150:
            raise ValueError("Возраст не может превышать 150 лет")

        print(f"Возраст {age} допустим")

    except ValueError as e:
        print(f"Ошибка: {e}")
    except TypeError as e:
        print(f"Ошибка: {e}")
    except Exception as e:
        print(f"Ошибка: {e}")


# Шаг 5
def validate_levels(levels):
    '''
    Валидирует уровни
    Выбрасывает исключения при отсутствии уровней или отрицательном номере
    '''
    try:
        if len(levels) == 0:
            raise ValueError("Список уровней пуст")
        for number in levels:
            if number <= 0:
                raise ValueError("Уровень с отрицательным номером не допустим")

    except ValueError as e:
        print(f"Ошибка: {e}")


# Шаг 6.1 (7)
def validate_email(email):
    '''
    Валидирует почту
    Выбрасывает исключение при отсутствии необходимых символов в почте
    '''
    try:
        if "@" not in email or "." not in email.split("@")[-1]:
            raise InvalidEmailError("Неверный формат электронной почты")
        print("Электронная почта валидна")

    except InvalidEmailError as e:
```

```
        print(f"Ошибка: {e}")


# Шаг 6.2 (7)
def validate_phone_number(phone_number):
    '''
    Валидирует номер телефона
    Выбрасывает исключение при неправильной длине номера
    '''
    try:
        if len(phone_number) != 10:
            raise InvalidPhoneNumberError("Неверный формат номера телефона")
        print("Номер телефона валиден")

    except InvalidPhoneNumberError as e:
        print(f"Ошибка: {e}")


# Шаг 6.3 (7)
def insert(ls, value, index):
    '''
    Вставляет элемент в список
    Выбрасывает исключение при выходе за пределы списка
    '''
    try:
        if index < 0 or index > len(ls) - 1:
            raise OutOfRangeError("Выход за пределы списка")
        ls.insert(index, value)

    except OutOfRangeError as e:
        print(f"Ошибка: {e}")


# Шаг 8.1
def add_unique(ls, value):
    '''
    Добавляет уникальный элемент в список
    Выбрасывает исключение, если в списке уже существует такой элемент
    '''
    try:
        if value in ls:
            raise ValueError("Элемент уже существует")
        ls.append(value)

    except ValueError as e:
        print(f"Ошибка: {e}")


# Шаг 8.2
def convert_to_int(value):
    '''
```

```
    Конвертирует значение в число
    Выбрасывает ошибку при невозможности конвертации
    '''
    try:
        if not can_convert_to_int(value):
            raise ValueError("Конвертация невозможна")
        return int(value)

    except ValueError as e:
        print(f"Ошибка конвертации: {e}")


# Шаг 8.3
def can_convert_to_int(value):
    '''Проверка на возможность конвертировать в int'''
    try:
        int(value)
        return True
    except ValueError:
        return False
```

```
'''Пользовательские исключения'''


class InvalidEmailError(Exception):
    '''Исключение, возникающее при неверном формате электронной почты'''


class InvalidPhoneNumberError(Exception):
    '''Исключение, возникающее при неверном формате номера телефона'''


class OutOfRangeError(Exception):
    '''Исключение, возникающее при выходе значения за пределы допустимого диапазона'''
```

Единственное, на что жалуется линтер – слишком обобщённые проверки исключений и размер функции с примерами. Оба аспекта требуются по заданию.

```
PS C:\Users\rmiro\Desktop\Лабы\3курс\Бек\lab4-Roman784> pylint main.py
************* Module main
main.py:86:11: W0718: Catching too general exception Exception (broad-exception-caught)
main.py:11:0: R0915: Too many statements (62/50) (too-many-statements)

-------------------------------------------------------------------
Your code has been rated at 9.69/10 (previous run: 9.69/10, +0.00)
```

```
PS C:\Users\rmiro\Desktop\Лабы\3курс\Бек\lab4-Roman784> pylint functions.py
************* Module functions
functions.py:30:11: W0718: Catching too general exception Exception (broad-exception-caught)
functions.py:49:11: W0718: Catching too general exception Exception (broad-exception-caught)
functions.py:75:11: W0718: Catching too general exception Exception (broad-exception-caught)
functions.py:105:11: W0718: Catching too general exception Exception (broad-exception-caught)
functions.py:130:11: W0718: Catching too general exception Exception (broad-exception-caught)

-------------------------------------------------------------------
Your code has been rated at 9.61/10 (previous run: 9.53/10, +0.08)
```

```
PS C:\Users\rmiro\Desktop\Лабы\3курс\Бек\lab4-Roman784> pylint exceptions.py

-----------------------------------
Your code has been rated at 10.00/10
```

```
PS C:\Users\rmiro\Desktop\Лабы\3курс\Бек\lab4-Roman784> python main.py
Шаг 1.1:
5.0
Необработанная ошибка: Ошибка деления на 0

Шаг 1.2
4.0
Необработанная ошибка: Невозможно извлечь корень из отрицательного числа

Шаг 2
2.23606797749979
Ошибка: Ошибка деления на 0
Ошибка: Невозможно извлечь корень из отрицательного числа

Шаг 3
Файл file1.txt открыт
Файл file1.txt загружен
Результат загрузки: True
Файл file1.txt закрыт
Файл file2 открыт
Ошибка: Ошибка загрузки
Результат загрузки: False
Файл file2 закрыт

Шаг 4.1
True
Ошибка: Строка не должна быть пустой
False
Ошибка: Входное значение должно быть строкой
False

Шаг 4.2
120
Ошибка: Входное значение должно быть положительным
0
Ошибка: Входное значение должно быть целым числом
0
Ошибка: Слишком большое число
0

Шаг 4.3
Возраст 30 допустим
Ошибка: Возраст не может быть отрицательным
Ошибка: Возраст не может превышать 150 лет
```

```
Шаг 5
Ошибка: Список уровней пуст
Ошибка: Уровень с отрицательным номером не допустим

Шаг 6.1 (7)
Электронная почта валидна
Ошибка: Неверный формат электронной почты

Шаг 6.2 (7)
Номер телефона валиден
Ошибка: Неверный формат номера телефона

Шаг 6.3 (7)
[1, 4, 2, 3]
Ошибка: Выход за пределы списка

Шаг 8.1
[1, 2, 3, 4]
Ошибка: Элемент уже существует

Шаг 8.2
1
Ошибка конвертации: Конвертация невозможна
a

Шаг 8.3
True
False
```