

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

**Отчет по лабораторной работе №4**  
**Работа с исключениями в языке Python**  
**По дисциплине «Объектно-ориентированное программирование»**

Выполнил студент группы ИВТ-б-о-20-1

Галяс Д. И. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р. А. \_\_\_\_\_

(подпись)

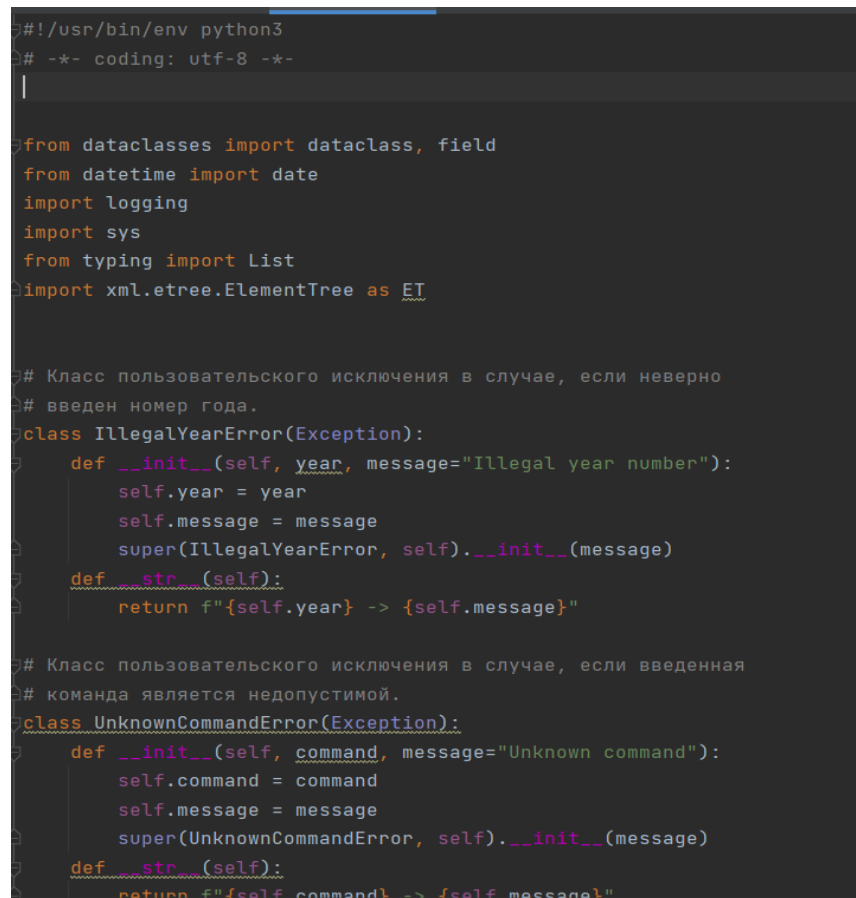
Ставрополь 2022

**Цель работы:** приобретение навыков по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.

**Ссылка на репозиторий:** <https://github.com/DIMITRY-GALYAS1/Laba-4.4.git>

**Ход работы:**

1. Создал новый репозиторий на github, после клонировал его и создал в папке репозитория новый проект PyCharm.
2. Выполнил пример.



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from dataclasses import dataclass, field
from datetime import date
import logging
import sys
from typing import List
import xml.etree.ElementTree as ET

# Класс пользовательского исключения в случае, если неверно
# введен номер года.
class IllegalYearError(Exception):
    def __init__(self, year, message="Illegal year number"):
        self.year = year
        self.message = message
        super(IllegalYearError, self).__init__(message)
    def __str__(self):
        return f"{self.year} -> {self.message}"

# Класс пользовательского исключения в случае, если введенная
# команда является недопустимой.
class UnknownCommandError(Exception):
    def __init__(self, command, message="Unknown command"):
        self.command = command
        self.message = message
        super(UnknownCommandError, self).__init__(message)
    def __str__(self):
        return f"{self.command} -> {self.message}"
```

Рисунок 1. Пример 1

3. Выполнил первое задание.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Решите следующую задачу: напишите программу, которая запрашивает ввод двух значений.
Если хотя бы одно из них не является числом, то должна выполняться конкатенация, т. е.
соединение, строк. В остальных случаях введенные числа суммируются.
"""

class Test:
    def __init__(self, first, second):
        self.first = first
        self.second = second

    def summa(self):
        k = int(self.first) + int(self.second)
        print("Сумма чисел: ", k)

    def concatenation(self):
        z = self.first + self.second
        print("Результат конкатенации: ", z)

if __name__ == "__main__":
    try:
        num1 = input("Введите первое число: ")
        num2 = input("Введите второе число: ")
        test = Test(num1, num2)
        test.summa()
```

Рисунок 2. Первое задание

#### 4. Выполнил второе задание.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from random import randint

"""
Решите следующую задачу: напишите программу, которая будет генерировать матрицу из
случайных целых чисел. Пользователь может указать число строк и столбцов, а также
диапазон целых чисел. Произведите обработку ошибок ввода пользователя.
"""

class Generation:
    def __init__(self, num1, num2, num3, num4):
        self.line = int(num1)
        self.column = int(num2)
        self.min = int(num3)
        self.max = int(num4)

    def make_matrix(self):
        lst = [[randint(self.min, self.max) for z in range(self.column)] for x in range(self.line)]
        for z in lst:
            print()
            for k in z:
                print(k, end=" ")

if __name__ == "__main__":
    try:
```

Рисунок 3. Второе задание

#### 5. Далее приступил к выполнению первого индивидуального задания.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse
import json
import pathlib
import logging

"""
Выполнить индивидуальное задание 1 лабораторной работы 2.19,
добавив возможность работы с исключениями и логгирование.
"""

class MyStudents:
    def __init__(self, line):
        self.line = line

    def select_student(self, undergraduates):
        """
        Выбрать студентов с заданной оценкой.
        """
        print(self.line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "No",
                "Ф.И.О.",
                "Группа",
                "Успеваемость"))
        print(self.line)
```

Рисунок 4. Первое индивидуальное задание

6. Затем выполнил второе индивидуальное задание.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse
import json
import pathlib
import logging
import logging.config

"""
Выполнить индивидуальное задание 1 лабораторной работы 2.19,
добавив возможность работы с исключениями и логгирование.
"""

class MyStudents:
    def __init__(self, line):
        self.line = line

    def select_student(self, undergraduates):
        """
        Выбрать студентов с заданной оценкой.
        """
        print(self.line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "No",
                "Ф.И.О.",
                "Группа",
                "Успеваемость"))
```

Рисунок 5. Второе индивидуальное задание

**Контрольные вопросы:**

1. Какие существуют виды ошибок в языке программирования Python?

- SystemExit;
- KeyboardInterrupt;
- GeneratorExit;
- Exception;
- StopIteration;
- StopAsyncIteration;
- ArithmeticError;
- FloatingPointError;
- OverflowError;
- ZeroDivisionError;
- AssertionError;
- AttributeError;
- BufferError;
- EOFError;
- ImportError;
- ModuleNotFoundError;
- LookupError;
- IndexError;
- KeyError;
- MemoryError;
- NameError;
- UnboundLocalError;
- OSError;
- BlockingIOError;
- ChildProcessError;
- ConnectionError;
- BrokenPipeError;

- ConnectionAbortedError;
- ConnectionRefusedError;
- ConnectionResetError;
- FileExistsError;
- FileNotFoundError;
- InterruptedError;
- IsADirectoryError;
- NotADirectoryError;
- PermissionError;
- ProcessLookupError;
- TimeoutError;
- ReferenceError;
- RuntimeError;
- NotImplementedError;
- RecursionError;
- SyntaxError;
- IndentationError;
- TabError;
- SystemError;
- TypeError;
- ValueError;
- UnicodeError;
- UnicodeDecodeError;
- UnicodeEncodeError;
- UnicodeTranslateError;
- Warning;
- DeprecationWarning;
- PendingDeprecationWarning;
- RuntimeWarning;

- SyntaxWarning;
- UserWarning;
- FutureWarning;
- ImportWarning;
- UnicodeWarning;
- BytesWarning;
- ResourceWarning.

2. Как осуществляется обработка исключений в языке программирования Python?

Обработка исключений нужна для того, чтобы приложение не завершалось аварийно каждый раз, когда возникает исключение. Для этого блок кода, в котором возможно появление исключительной ситуации необходимо поместить во внутрь синтаксической конструкции `try... except`.

3. Для чего нужны блоки `finally` и `else` при обработке исключений?

Не зависимо от того, возникнет или нет во время выполнения кода в блоке `try` исключение, код в блоке `finally` все равно будет выполнен.

Если необходимо выполнить какой-то программный код, в случае если в процессе выполнения блока `try` не возникло исключений, то можно использовать оператор `else`.

4. Как осуществляется генерация исключений в языке Python?

Для принудительной генерации исключения используется инструкция `raise`.

5. Как создаются классы пользовательский исключений в языке Python?

Для реализации собственного типа исключения необходимо создать класс, являющийся наследником от одного из классов исключений.

6. Каково назначение модуля `logging`?

Для вывода специальных сообщений, не влияющих на функционирование программы, в Python применяется библиотека логов.

Чтобы воспользоваться ею, необходимо выполнить импорт в верхней части файла. С помощью logging на Python можно записывать в лог и исключения.

7. Какие уровни логгирования поддерживаются модулем logging? Приведите примеры, в которых могут быть использованы сообщения с этим уровнем журналирования.

DEBUG:root:Debug message!

INFO:root:Info message!

WARNING:root:Warning message!

ERROR:root>Error message!

CRITICAL:root:Critical message!

**Вывод:** в ходе выполнения лабораторной работы приобрел навыки по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.