

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»**

Кафедра инфокоммуникаций

**Отчет по лабораторной работе № 4.4
«Работа с исключениями в языке Python»**

по дисциплине «Объектно-ориентированное программирование»

Выполнил студент группы ИВТ-б-о-20-1

Бобров Н.В. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

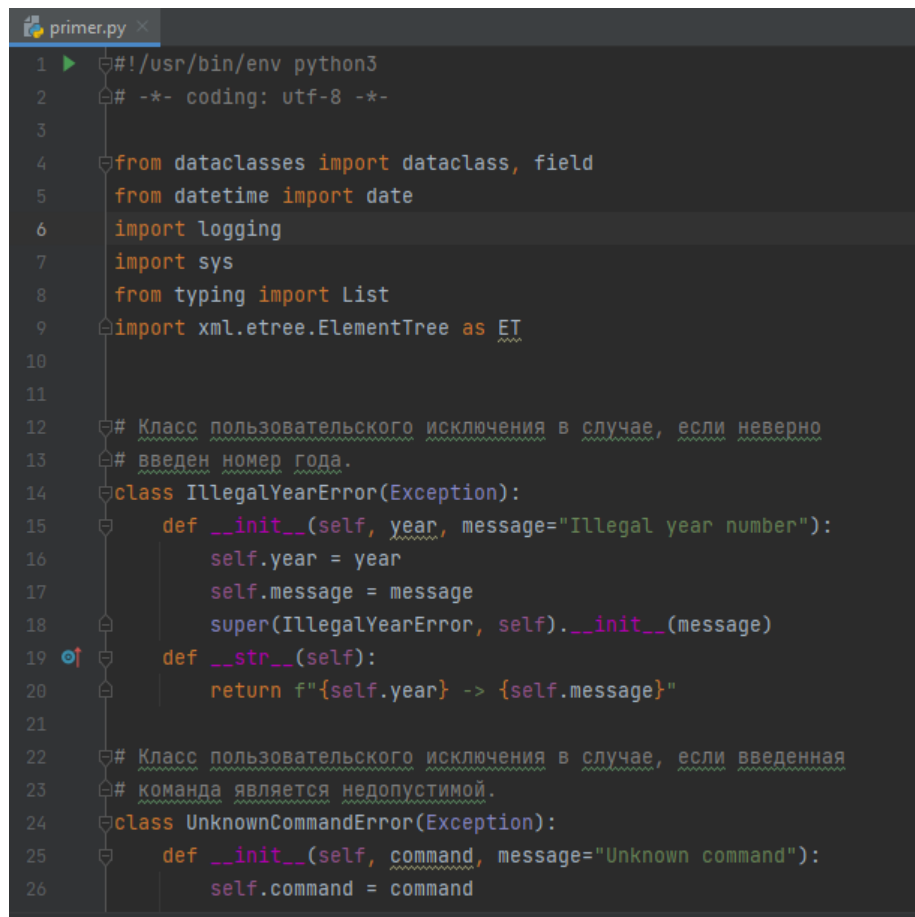
Проверил Воронкин Р.А. _____

(подпись)

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

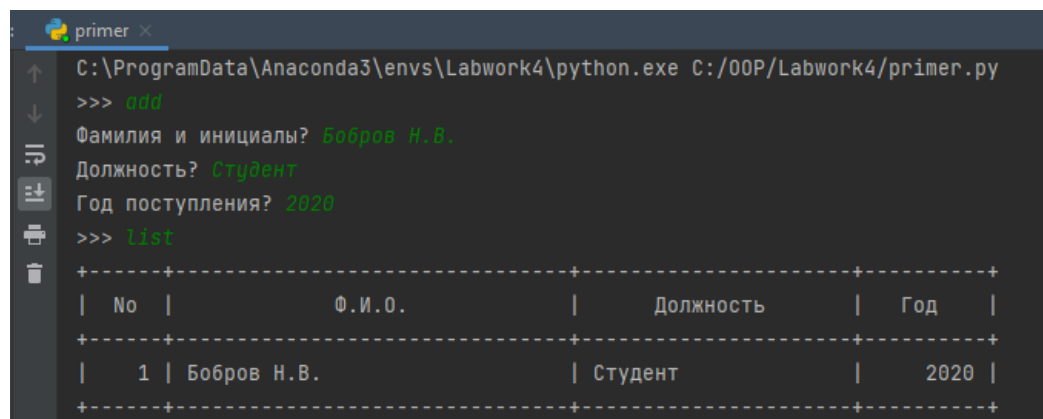
Ход работы:

1. Создал общедоступный репозиторий и клонировал его на локальный сервер.
2. Изучив теоретический материал, приступил к выполнению примера.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 from dataclasses import dataclass, field
5 from datetime import date
6 import logging
7 import sys
8 from typing import List
9 import xml.etree.ElementTree as ET
10
11
12 # Класс пользовательского исключения в случае, если неверно
13 # введен номер года.
14 class IllegalYearError(Exception):
15     def __init__(self, year, message="Illegal year number"):
16         self.year = year
17         self.message = message
18         super(IllegalYearError, self).__init__(message)
19     def __str__(self):
20         return f"{self.year} -> {self.message}"
21
22 # Класс пользовательского исключения в случае, если введенная
23 # команда является недопустимой.
24 class UnknownCommandError(Exception):
25     def __init__(self, command, message="Unknown command"):
26         self.command = command
```

Рисунок 1 – Код примера



```
C:\ProgramData\Anaconda3\envs\Labwork4\python.exe C:/00P/Labwork4/primer.py
>>> add
Фамилия и инициалы? Бобров Н.В.
Должность? Студент
Год поступления? 2020
>>> list
```

No	Ф.И.О.	Должность	Год
1	Бобров Н.В.	Студент	2020

Рисунок 2 – Результат работы кода

workers.log – Блокнот

Файл Правка Формат Вид Справка

```
INFO:root:Добавлен сотрудник: Бобров Н.В., Студент, поступивший в 2020 году.  
INFO:root:Отображен список сотрудников.  
INFO:root:Сохранены данные в файл workers.json.  
INFO:root:Добавлен сотрудник: Бобров Н.В., Студент, поступивший в 2020 году.  
INFO:root:Отображен список сотрудников.
```

Рисунок 3 – Запись в логах

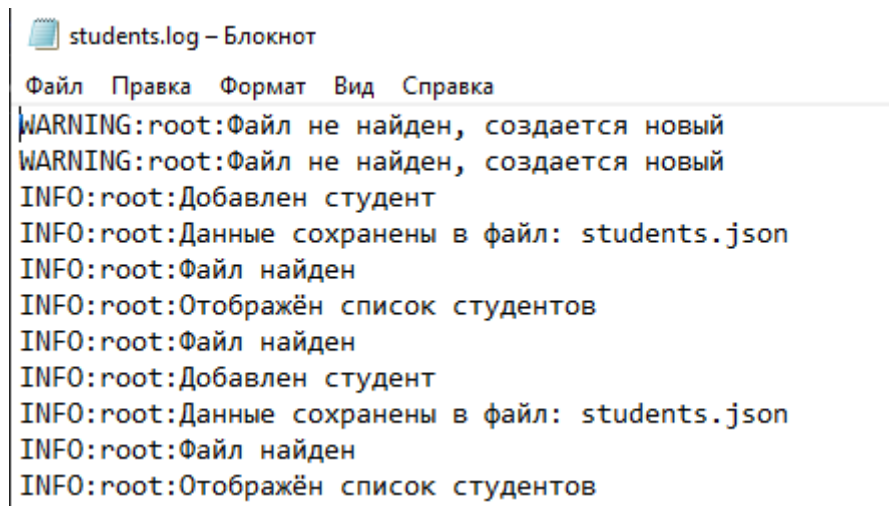
3. Приступил к выполнению индивидуального задания, взял за основу код с лабораторной работы 2.19, добавив возможность работы с исключениями и логгирование.

```
163  
164     try:  
165         students = st.load_student(home)  
166         logging.info("Файл найден")  
167     except FileNotFoundError:  
168         students = []  
169         logging.warning("Файл не найден, создается новый")  
170  
171     if args.command == "add":  
172         students = st.add_student(students, args.name, args.group, args.grade)  
173         is_dirty = True  
174         logging.info("Добавлен студент")  
175     elif args.command == 'display':  
176         st.display(students)  
177         logging.info("Отображён список студентов")  
178     elif args.command == "select":  
179         st.select_student(students)  
180         logging.info("Выбраны студенты с нужными оценками")  
181  
182     if is_dirty:  
183         st.save_student(args.filename, students)
```

Рисунок 4 – Дополнение к коду

```
Terminal: Local x + v  
PS C:\00P\Labwork4> python individual1.py add students.json -n "Бобров Николай" -g "1" -gr "5 5 5 5 5"  
PS C:\00P\Labwork4> python individual1.py display students.json  
+-----+  
| No |          Ф.И.О.          |      Группа      |  Успеваемость  |  
+-----+  
| 1  | Бобров Н.В.              | 5                 | 5 5 5 5 5      |  
| 2  | Бобров Николай          | 1                 | 5 5 5 5 5      |  
+-----+
```

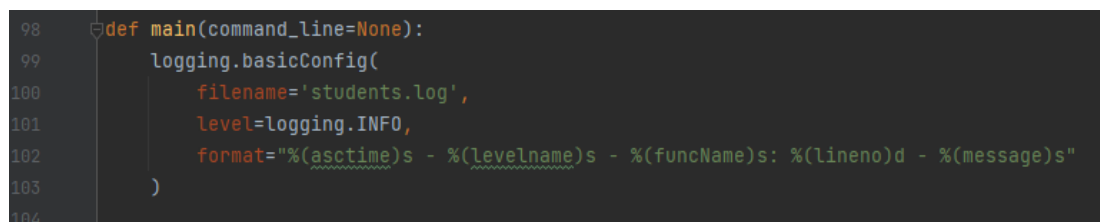
Рисунок 5 – Работа кода



```
students.log – Блокнот
Файл  Правка  Формат  Вид  Справка
WARNING:root:Файл не найден, создается новый
WARNING:root:Файл не найден, создается новый
INFO:root:Добавлен студент
INFO:root:Данные сохранены в файл: students.json
INFO:root:Файл найден
INFO:root:Отображён список студентов
INFO:root:Файл найден
INFO:root:Добавлен студент
INFO:root:Данные сохранены в файл: students.json
INFO:root:Файл найден
INFO:root:Отображён список студентов
```

Рисунок 6 – Список логов

4. Приступил к выполнению второго индивидуального задания, добавив вывод в файлы лога даты и времени выполнения пользовательской команды с точностью до миллисекунды.

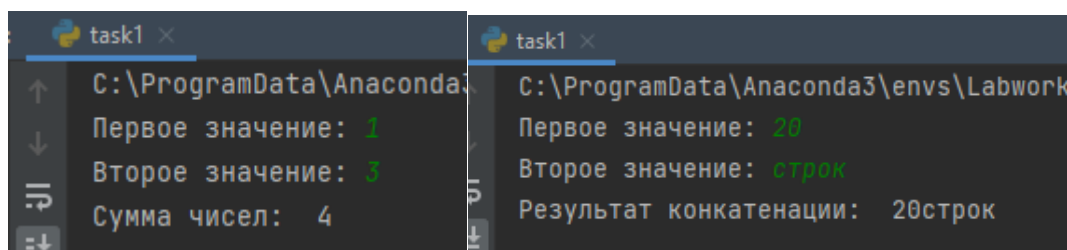


```
98 def main(command_line=None):
99     logging.basicConfig(
100         filename='students.log',
101         level=logging.INFO,
102         format="%(asctime)s - %(levelname)s - %(funcName)s: %(lineno)d - %(message)s"
103     )
104
```

Рисунок 7 – Вывод времени в логах

5. Выполнил общие задания.

Задание 1. 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.



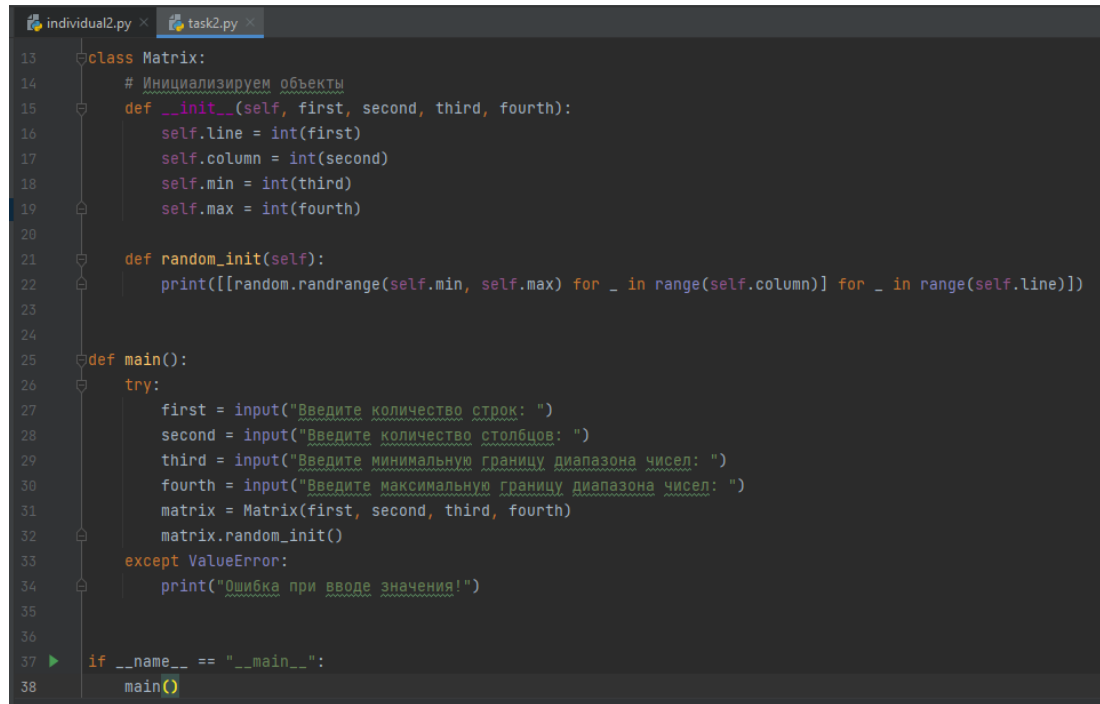
```
task1 x C:\ProgramData\Anaconda3\envs\Labwork
Первое значение: 1
Второе значение: 3
Сумма чисел: 4

task1 x C:\ProgramData\Anaconda3\envs\Labwork
Первое значение: 20
Второе значение: строка
Результат конкатенации: 20строка
```

Рисунок 8 – Результат выполнения задания

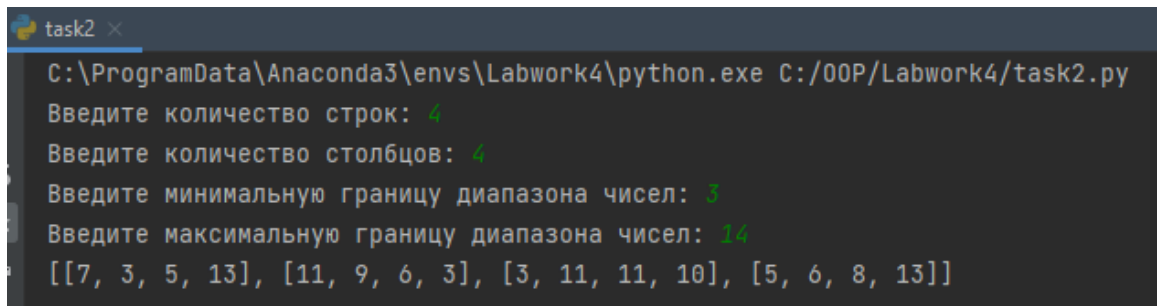
Задание 2. Напишите программу, которая будет генерировать матрицу из случайных целых чисел. Пользователь может указать число строк и

столбцов, а также диапазон целых чисел. Произведите обработку ошибок ввода пользователя.



```
13 class Matrix:
14     # Инициализируем объекты
15     def __init__(self, first, second, third, fourth):
16         self.line = int(first)
17         self.column = int(second)
18         self.min = int(third)
19         self.max = int(fourth)
20
21     def random_init(self):
22         print([[random.randrange(self.min, self.max) for _ in range(self.column)] for _ in range(self.line)])
23
24
25 def main():
26     try:
27         first = input("Введите количество строк: ")
28         second = input("Введите количество столбцов: ")
29         third = input("Введите минимальную границу диапазона чисел: ")
30         fourth = input("Введите максимальную границу диапазона чисел: ")
31         matrix = Matrix(first, second, third, fourth)
32         matrix.random_init()
33     except ValueError:
34         print("Ошибка при вводе значения!")
35
36
37 if __name__ == "__main__":
38     main()
```

Рисунок 9 – Код второго задания



```
task2 x
C:\ProgramData\Anaconda3\envs\Labwork4\python.exe C:/OOP/Labwork4/task2.py
Введите количество строк: 4
Введите количество столбцов: 4
Введите минимальную границу диапазона чисел: 3
Введите максимальную границу диапазона чисел: 14
[[7, 3, 5, 13], [11, 9, 6, 3], [3, 11, 11, 10], [5, 6, 8, 13]]
```

Рисунок 10 – Результат выполнения кода

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 были приобретены простейшие навыки по работе с исключениями в языке программирования Python.