Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
дисциплины
«Объектно-ориентированное программирование»
Вариант 20

Выполнила:
Михеева Елена Александровна
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Воронкин Р.А.-доцент департамента цифровых, роботехнических систем и электроники института перспективной инженерии

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

ТЕМА: РАБОТА С ИСКЛЮЧЕНИЯМИ В ЯЗЫКЕ PYTHON

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

Порядок выполнения работы:

Ссылка на Github: https://github.com/helendddd/Oop_4.git

1. Была разработана программа для решения следующей задачи:

Необходимо написать 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.

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

if __name__ == '__main__':
    # Запрашиваем первое значение
    first_value = input("Первое значение: ")

    # Запрашиваем второе значение
    second_value = input("Второе значение: ")

    # Проверяем, являются ли оба значения числами
    try:
        # Пытаемся преобразовать оба значения в числа (float)
        num1 = float(first_value)
        num2 = float(second_value)
        # Если оба значения числа, выполняем их суммирование
        result = num1 + num2
        print("Результат:", result)
    except ValueError:
        # Если хотя бы одно из значений не число, выполняем конкатенацию
        result = first_value + second_value
        print("Результат:", result)
```

Рисунок 1. Программа задания №1

```
(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_4 % python3 program/ex1.py
Первое значение: 4
Второе значение: 5
Результат: 9.0
(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_4 % python3 program/ex1.py
Первое значение: a
Второе значение: 9
Результат: a9
```

Рисунок 2. Результат работы программы

2. Была разработана программа для решения следующей задачи:

Необходимо написать 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.

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


import random


def generate_matrix(rows, cols, min_val, max_val):
    """Генерирует матрицу случайных целых чисел."""
    if rows <= 0 or cols <= 0:
        raise ValueError("Число строк и столбцов должно быть больше нуля.")
    if min_val > max_val:
        raise ValueError("Минимальное значение не может быть больше максимума")

    return [
        [random.randint(min_val, max_val) for _ in range(cols)]
        for _ in range(rows)
    ]


def main():
    print("Программа для генерации матрицы из случайных целых чисел.")
    try:
        rows = int(input("Введите количество строк: "))
        cols = int(input("Введите количество столбцов: "))
        min_val = int(input("Введите минимальное значение: "))
        max_val = int(input("Введите максимальное значение: "))

        matrix = generate_matrix(rows, cols, min_val, max_val)
        print("Сгенерированная матрица:")
        for row in matrix:
            print(row)

    except ValueError as e:
        print(f"Ошибка ввода: {e}")
    except Exception as e:
        print(f"Произошла ошибка: {e}")


if __name__ == "__main__":
    main()

```

Рисунок 3. Программа задания 2

```
Программа для генерации матрицы из случайных целых чисел.
Введите количество строк: 4
Введите количество столбцов: 5
Введите минимальное значение: 2
Введите максимальное значение: 98
Сгенерированная матрица:
[21, 16, 52, 90, 17]
[89, 51, 67, 88, 35]
[46, 91, 65, 76, 38]
[90, 45, 36, 74, 14]
```

Рисунок 4. Результат работы программы

3. Приступили к выполнению индивидуального задания.

Задание: выполнить индивидуальное задание 1 лабораторной работы 2.19, добавив возможность работы с исключениями и логгирование. Изучить возможности модуля logging. Добавить для предыдущего задания вывод в файлы лога даты и времени выполнения пользовательской команды с точностью до миллисекунды.

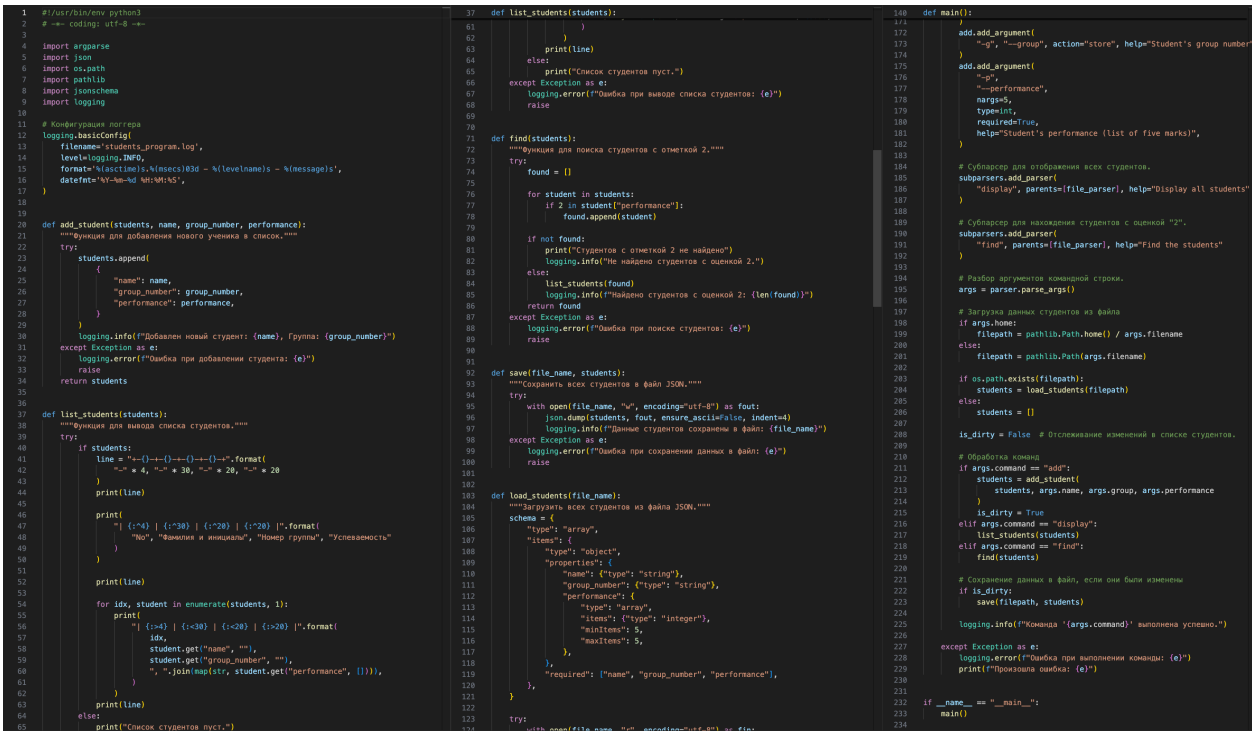

Рисунок 5. Программа индивидуального задания

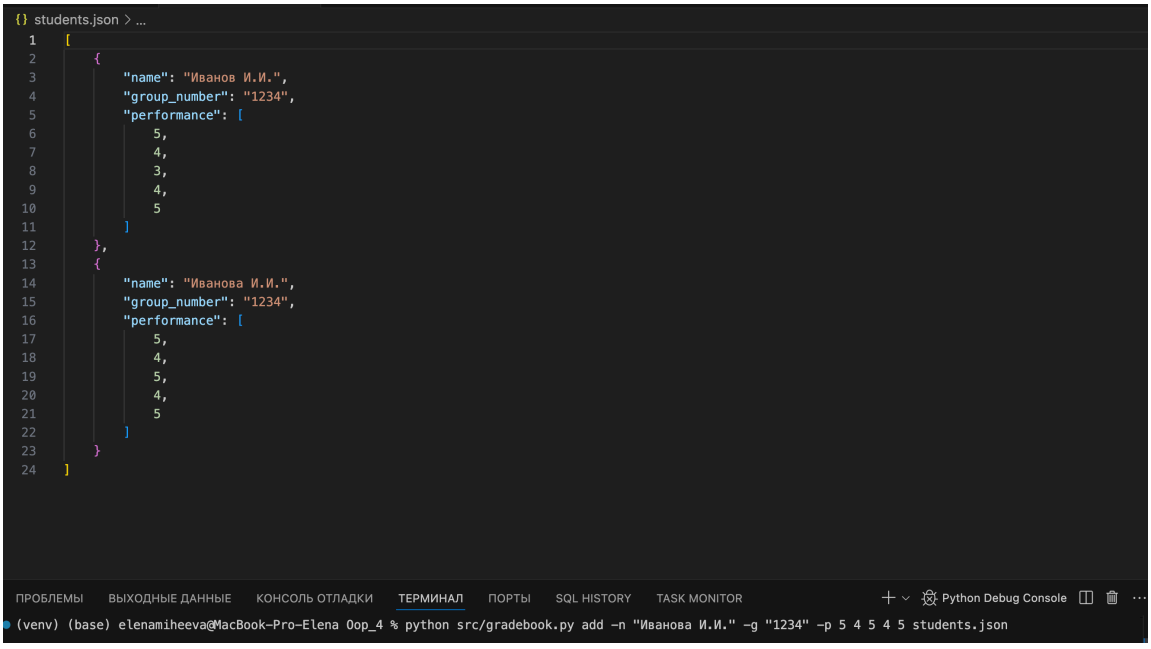

Рисунок 2. Результат работы программы

Ответы на контрольные вопросы:

1. Какие существуют виды ошибок в языке программирования Python?

В Python выделяют два различных вида ошибок: синтаксические ошибки и исключения.

Синтаксические ошибки возникают в случае, если программа написана с нарушениями *требований Python* к синтаксису. Определяются они в процессе парсинга программы.

Второй вид ошибок – это исключения. Они возникают в случае, если синтаксически программа корректна, но в процессе выполнения возникает ошибка (деление на ноль и т.п.).

2. Как осуществляется обработка исключений в языке программирования Python?

Для блока 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.

Например:

```
try:
    raise Exception("Some exception”)
except Exception as e:
    print(“Exception exception ” + str(e))
```

5. Как создаются классы пользовательский исключений в языке Python?

В Python классы пользовательских исключений создаются путем определения нового класса, который наследует от стандартного класса исключений. Обычно все исключения в Python являются подклассами встроенного класса Exception

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.

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

Модуль logging в Python поддерживает следующие уровни логирования:

DEBUG (10) — для отладочной информации, полезной при разработке.

INFO (20) — для обычной информации о ходе работы программы.

WARNING (30) — для предупреждений о возможных проблемах.

ERROR (40) — для ошибок, которые не останавливают программу, но требуют внимания.

CRITICAL (50) — для критических ошибок, приводящих к сбою программы.

Пример:

```
import logging
logging.basicConfig(level=logging.DEBUG)
logging.debug("Отладка")
logging.info("Информация")
logging.warning("Предупреждение")
logging.error("Ошибка")
logging.critical("Критическая ошибка")
```

Каждый уровень фильтрует сообщения по приоритету: чем выше уровень, тем меньше сообщений будет выводиться.

Вывод: были приобретены навыки по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.