

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

**Кафедра инфокоммуникаций**

**Объектно-ориентированное программирование**

**Отчет по лабораторной работе №4.4**

Работа с исключениями в языке Python

Выполнил студент группы

ИВТ-б-о-21-1

Харченко Б.Р. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

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

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А.

\_\_\_\_\_  
(подпись)

Ставрополь 2023

## Наследование и полиморфизм в языке Python.

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

### Порядок выполнения работы:

#### Задание 1.

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.

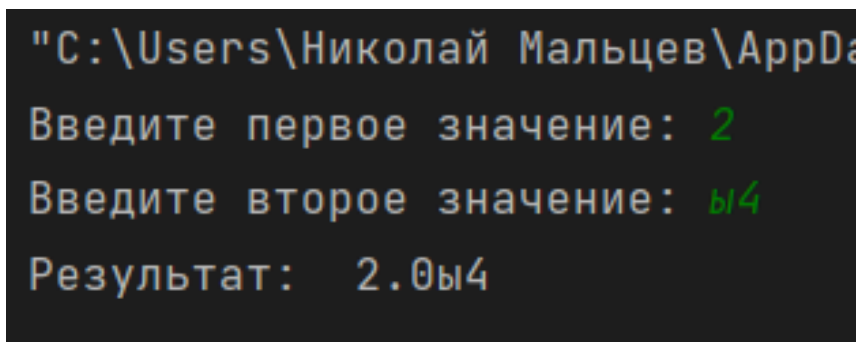
#### Код программы:

```
value1 = input("Введите первое значение: ")
value2 = input("Введите второе значение: ")

try:
    value1 = float(value1)
    value2 = float(value2)
    result = value1 + value2
except ValueError:
    result = str(value1) + str(value2)

print("Результат: ", result)
```

#### Результат работы программы:



```
"C:\Users\Николай Мальцев\AppData
Введите первое значение: 2
Введите второе значение: ы4
Результат: 2.0ы4
```

Рисунок 1. Результат работы программы

#### Задание 2.

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.

#### Код программы:

```
import random
import MyExceptions as me
```

```
def generate_matrix(rows, columns, range_start, range_end):
    matrix = []
    for _ in range(rows):
        row = []
        for _ in range(columns):
            row.append(random.randint(range_start, range_end))
        matrix.append(row)
    return matrix

if __name__ == "__main__":
    while True:
        try:
            rows = int(input("Введите количество строк: "))
            columns = int(input("Введите количество столбцов: "))
            range_start = int(input("Введите начало диапазона целых чисел: "))
            range_end = int(input("Введите конец диапазона целых чисел: "))

            if rows <= 0 or columns <= 0 or range_start > range_end:
                raise ValueError("Неверный диапазон!")
            break
        except ValueError as e:
            print(str(e))

    matrix = generate_matrix(rows, columns, range_start, range_end)
    print("Сгенерированная матрица:")
    for row in matrix:
        print(row)
```

Результат работы программы:

```
Введите количество строк: 4
Введите количество столбцов: 5
Введите начало диапазона целых чисел: 12
Введите конец диапазона целых чисел: 2
Error, Неверный диапазон!
Введите количество строк: 4
Введите количество столбцов: 4
Введите начало диапазона целых чисел: 1
Введите конец диапазона целых чисел: 9
Сгенерированная матрица:
[9, 4, 5, 8]
[1, 1, 2, 9]
[9, 6, 3, 3]
[7, 4, 8, 4]
```

Рисунок 2. Результат работы программы

Индивидуальное задание.

Код программы:

```
# Настройка логгирования
logging.basicConfig(filename='students.log', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

def add_student(students, name, group, progress):
    """
    Запросить данные о студенте.
```

```

"""
students.append(
    {
        'name': name,
        'group': group,
        'progress': progress,
    }
)
return students

def display_students(students):
    """
    Отобразить список студентов.
    """
    # Проверить, что список студентов не пуст.
    if students:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "No",
                "Ф.И.О.",
                "Группа",
                "Успеваемость"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(students, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>15} |'.format(
                    idx,
                    student.get('name', ''),
                    student.get('group', ''),
                    student.get('progress', 0)
                )
            )
            print(line)

    else:
        print("Список студентов пуст")

def select_students(undergraduates):
    """
    Выбрать студентов с заданной оценкой.
    """
    # Сформировать список студентов.
    result = []
    # Просмотреть оценки студентов.
    for pupil in undergraduates:
        # Делаем список оценок
        evaluations = pupil.get('progress')
        list_of_rating = list(evaluations)
        # Ищем нужную оценку.
        for i in list_of_rating:
            if i == '2':
                result.append(pupil)
    # Возвратить список выбранных студентов.
    return result

def save_students(file_name, undergraduates):

```

```

"""
Сохранить всех студентов в файл JSON.
"""
try:
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(undergraduates, fout, ensure_ascii=False, indent=4)
    directory = pathlib.Path.cwd().joinpath(file_name)
    directory.replace(pathlib.Path.home().joinpath(file_name))
    print("Данные сохранены")
    logging.info(f'Данные сохранены в файл: {file_name}')
except Exception as e:
    print(f"Ошибка при сохранении данных: {str(e)}")
    logging.error(f"Ошибка при сохранении данных: {str(e)}")

def load_students(file_name):
    """Загрузить всех студентов из файла JSON."""
    try:
        # Открыть файл с заданным именем для чтения.
        with open(file_name, "r", encoding="utf-8") as fin:
            return json.load(fin)
    except FileNotFoundError:
        print(f"Файл {file_name} не найден.")
        logging.error(f"Файл {file_name} не найден.")
        return []
    except Exception as e:
        print(f"Ошибка при загрузке данных: {str(e)}")
        logging.error(f"Ошибка при загрузке данных: {str(e)}")
        return []

def main(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )

    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("students")
    parser.add_argument(
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )

    subparsers = parser.add_subparsers(dest="command")

    # Создать субпарсер для добавления студента.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new student"
    )
    add.add_argument(
        "-n",
        "--name",
        action="store",
        required=True,
        help="The student's name"
    )
    add.add_argument(
        "-g",
        "--group",
        action="store",
        help="The worker's group"
    )

```

```

add.add_argument(
    "-p",
    "--progress",
    action="store",
    required=True,
    help="Academic performance"
)

# Создать субпарсер для отображения всех студентов.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all students"
)

# Создать субпарсер для выбора студентов.
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the students"
)
select.add_argument(
    "-e",
    "--estimation",
    action="store",
    type=int,
    required=False,
    help="The required estimation"
)

# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)

# Загрузить всех студентов из файла, если файл существует.
is_dirty = False
if os.path.exists(args.filename):
    students = load_students(args.filename)
else:
    students = []

# Добавить студента.
if args.command == "add":
    students = add_student(
        students,
        args.name,
        args.group,
        args.progress
    )
    is_dirty = True

# Отобразить всех студентов.
elif args.command == "display":
    display_students(students)

# Выбрать требуемых студентов.
elif args.command == "select":
    selected = select_students(students)
    display_students(selected)

# Сохранить данные в файл, если список студентов был изменен.
if is_dirty:
    save_students(args.filename, students)

if __name__ == "__main__":
    main()

```



## **Ответы на вопросы:**

### **1. Какие существуют виды ошибок в языке программирования Python?**

Синтаксические ошибки, возникающие, если программа написана с нарушением требований Python к синтаксису, и исключения, если в процессе выполнения возникает ошибка.

### **2. Как осуществляется обработка исключений в языке программирования Python?**

Блок кода, в котором возможно появление исключительной ситуации необходимо поместить во внутрь синтаксической конструкции `try... except`. Если в блоке `try` возникнет ошибка, программа выполнит блок `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 пользовательских 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 логгирования.**

- **Debug:** самый низкий уровень логгирования, предназначенный для отладочных сообщений, для вывода диагностической информации о приложении.
- **Info:** этот уровень предназначен для вывода данных о фрагментах кода, работающих так, как ожидается.
- **Warning:** этот уровень логгирования предусматривает вывод предупреждений, он применяется для записи сведений о событиях, на которые программист обычно обращает внимание. Такие события вполне могут привести к проблемам при работе приложения. Если явно не задать уровень логгирования — по умолчанию используется именно warning.
- **Error:** этот уровень логгирования предусматривает вывод сведений об ошибках — о том, что часть приложения работает не так как ожидается, о том, что программа не смогла правильно выполниться.
- **Critical:** этот уровень используется для вывода сведений об очень серьёзных ошибках, наличие которых угрожает нормальному функционированию всего приложения. Если не исправить такую ошибку — это может привести к тому, что приложение прекратит работу.

**Вывод:** в ходе работы были приобретены навыки по обработке исключений и логгированию при написании программ с использованием языка программирования Python версии 3.x.