

**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

Мальцев Н.А. « » \_\_\_\_\_ 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.

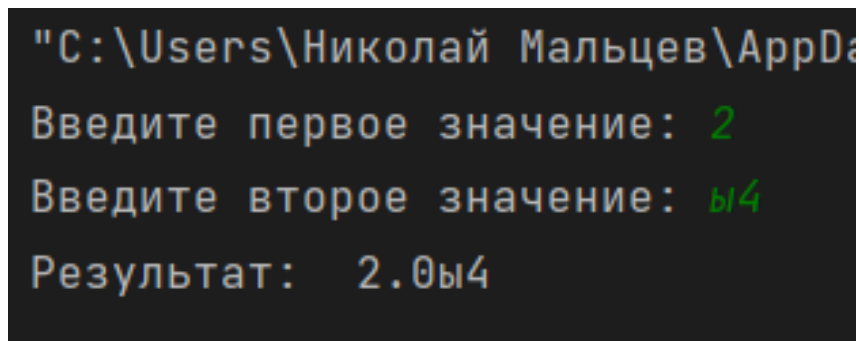
#### Код программы:

```
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 me.InvalidRangeValueException("Неверный диапазон!")
            break
        except me.InvalidRangeValueException 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:

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

import json
```

```

import os.path
import argparse
import pathlib
import logging
import MyExceptions as ME

def add_mans(list_man, name, number, date_):
    # Добавление данных
    list_man.append({"name": name, "number": number, "date": date_})
    return list_man

def list_d(list_man):
    """
    Отображение списка людей
    """
    # Проверить, что список не пуст
    if list_man:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-"
" * 20)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^20} |".format(
                "No", "Имя", "Номер телефона", "Дата рождения"
            )
        )
        print(line)

        # Вывести данные о человеке.
        for idx, man in enumerate(list_man, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:<20} |".format(
                    idx, man.get("name", ""), man.get("number", 0),
man.get("date", 0)
                )
            )

            print(line)
    else:
        print("Список работников пуст: ")

def select_mans(mans_list, numb):
    """
    Отбор пользователей с заданным номером телеофна
    """
    count = 0
    if mans_list:
        for man in mans_list:
            if man.get("number") == numb:
                count += 1
                print("{:>4}: {}".format(count, man.get("name", "")))
                print("Номер телефона:", man.get("number", ""))
                print("Дата рождения:", man.get("date", ""))
    else:
        print("Список пользователей пуст.")
    # Если счетчик равен 0, то человек не найден.
    if count == 0:
        print("Человек не найден.")

def save_workers(file_name_1, staff):

```

```

"""
Сохранить всех пользователей в файл JSON.
"""
try:
    with open(file_name_1, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4, default=str)
    directory = pathlib.Path.cwd().joinpath(file_name_1)
    directory.replace(pathlib.Path.home().joinpath(file_name_1))
except Exception as ex:
    print(str(ex))

def load_workers(file_name_2):
    """
    Загрузить всех работников из файла JSON.
    """
    try:
        with open(file_name_2, "r", encoding="utf-8") as fin:
            return json.load(fin)
    except FileNotFoundError as ex:
        print(str(ex))

def main(command_line=None):
    # Настройка логгера
    logging.basicConfig(
        filename='workers.log',
        level=logging.INFO,
        filemode="w",
        format="%(asctime)s %(levelname)s %(message)s",
        encoding="UTF-8"
    )

    # Создаем родительский парсер для определения имени файла
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument("filename", action="store", help="The date file name")

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

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

    # Субпарсер для добавления пользователей
    add = subparser.add_parser("add", parents=[file_parser], help="Add a new worker")
    add.add_argument(
        "-n", "--name", action="store", required=True, help="The worker name"
    )
    add.add_argument(
        "-N", "--number", action="store", type=str, help="The workers phone number"
    )
    add.add_argument(
        "-y", "--year", action="store", type=int, required=True, help="Man's birthdate"
    )

    # Субпарсер для отображения всех пользователей
    _ = subparser.add_parser(
        "display", parents=[file_parser], help="Display information about users"
    )

    # Субпарсер для выбора пользователей

```

```

select = subparser.add_parser("select", parents=[file_parser],
help="Select users")
select.add_argument(
    "-p", "--phone", action="store", type=str, help="The required period"
)
# Разбор аргументов командной строки
args = parser.parse_args(command_line)
logging.info("Произведён разбор аргументов командной строки")
# Загрузить всех пользователей из файла, если он существует
is_dirty = False
if os.path.exists(args.filename):
    mans = load_workers(args.filename)
    logging.info(f"Пользователи из файла {args.filename} успешно
загружены")
else:
    mans = []
# Добавление пользователя
if args.command == "add":
    try:
        mans = add_mans(mans, args.name, args.number, args.year)
        is_dirty = True
        logging.info("Пользователь успешно добавлен")
    except Exception as ex:
        logging.exception(ex)
        print(str(ex))
# Отображение всех пользователей
elif args.command == "display":
    list_d(mans)
    logging.info("Отображен список сотрудников.")
# Отбор требуемых пользователей
elif args.command == "select":
    select_mans(mans, args.phone)
    logging.info("Отобраны работники")
if is_dirty:
    try:
        save_workers(args.filename, mans)
        logging.warning("Список работников сохранён в файл " +
args.filename)
    except Exception as ex:
        logging.exception(ex)
        print(str(ex))

if __name__ == "__main__":
    main()

```

Результат работы программы:

```

PS C:\Users\Николай Мальцев\OneDrive\Рабочий стол\ООП\Практика\Lab_4.4\Tasks> python Ind_Task.py add "C:\Users\Николай
Мальцев\OneDrive\Рабочий стол\ООП\Практика\Lab_4.4\Tasks\test.json" --name "Nik" --number "89627" --year "2006"
PS C:\Users\Николай Мальцев\OneDrive\Рабочий стол\ООП\Практика\Lab_4.4\Tasks> python Ind_Task.py display test.json
+-----+-----+-----+-----+
| No |      Имя      |   Номер телефона   |   Дата рождения   |
+-----+-----+-----+-----+
|  1 | Nik           | 89627              | 2006              |
+-----+-----+-----+-----+
PS C:\Users\Николай Мальцев\OneDrive\Рабочий стол\ООП\Практика\Lab_4.4\Tasks>

```

Рисунок 3. Результат работы программы

Результат записи в log:

1	2023-11-19 16:46:17,263 INFO Произведён разбор аргументов командной строки
2	2023-11-19 16:46:17,264 INFO Пользователи из файла test.json успешно загружены
3	2023-11-19 16:46:17,264 INFO Отображен список сотрудников.
4	

Рисунок 2. Результат логгирования

## 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.