**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

Урусов М.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:

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:

Рисунок 2. Результат работы программы

Индивидуальное задание.

Код программы:

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


# Класс пользовательского исключения в случае, если неверно
# введен номер года.
2 usages  new *
class IllegalYearError(Exception):
    new *
    def __init__(self, year, message="Illegal year number"):
        self.year = year
        self.message = message
        super(IllegalYearError, self).__init__(message)

    new *
    def __str__(self):
        return f"{self.year} -> {self.message}"


# Класс пользовательского исключения в случае, если введенная
# команда является недопустимой.
2 usages  new *
class UnknownCommandError(Exception):
    new *
    def __init__(self, command, message="Unknown command"):
        self.command = command
        self.message = message
        super(UnknownCommandError, self).__init__(message)

    new *
    def __str__(self):
        return f"{self.command} -> {self.message}"


3 usages  new *
@dataclass(frozen=True)
class Worker:
    name: str
    post: str
```

```python
class Worker:
    name: str
    post: str
    year: int


1 usage  new *
@dataclass
class Staff:
    workers: List[Worker] = field(default_factory=lambda: [])

    1 usage  new *
    def add(self, name, post, year):
        # Получить текущую дату.
        today = date.today()

        if year < 0 or year > today.year:
            raise IllegalYearError(year)

        self.workers.append(Worker(name=name, post=post, year=year))
        self.workers.sort(key=lambda worker: worker.name)

    new *
    def __str__(self):
        # Заголовок таблицы.
        table = []
        line = "+-{}-+-{}-+-{}-+-{}-+".format( *args: "-" * 4, "-" * 30, "-" * 20, "-" * 8)
        table.append(line)
        table.append(
            "| {:^4} | {:^30} | {:^20} | {:^8} |".format(
                *args: "№", "Ф.И.О.", "Должность", "Год"
            )
        )
        table.append(line)
        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(self.workers, 1):
            table.append(
                "| {:>4} | {:<30} | {:<20} | {:>8} |".format(
                    *args: idx, worker.name, worker.post, worker.year
```

```
                )
            )
        table.append(line)

        return "\n".join(table)

    1 usage  new *
    def select(self, period):
        # Получить текущую дату.
        today = date.today()

        result = []
        for worker in self.workers:
            if today.year - worker.year >= period:
                result.append(worker)

        return result

    1 usage  new *
    def load(self, filename):
        with open(filename, "r", encoding="utf8") as fin:
            xml = fin.read()

        parser = ET.XMLParser(encoding="utf8")
        tree = ET.fromstring(xml, parser=parser)

        self.workers = []
        for worker_element in tree:
            name, post, year = None, None, None

            for element in worker_element:
                if element.tag == "name":
                    name = element.text
                elif element.tag == "post":
                    post = element.text
                elif element.tag == "year":
                    year = int(element.text)
                if name is not None and post is not None and year is not None:
                    self.workers.append(Worker(name=name, post=post, year=year))

```

```python


if __name__ == "__main__":
    # Выполнить настройку логгера.
    logging.basicConfig(filename="workers.log", level=logging.INFO)

    # Список работников.
    staff = Staff()

    # Организовать бесконечный цикл запроса команд.
    while True:
        try:
            # Запросить команду из терминала.
            command = input(">>> ").lower()
            # Выполнить действие в соответствие с командой.
            if command == "exit":
                break
            elif command == "add":
                # Запросить данные о работнике.
                name = input("Фамилия и инициалы? ")
                post = input("Должность? ")
                year = int(input("Год поступления? "))

                # Добавить работника.
                staff.add(name, post, year)
                logging.info(
                    f"Добавлен сотрудник: {name}, {post}, "
                    f"поступивший в {year} году."
                )
            elif command == "list":
                # Вывести список.
                print(staff)
                logging.info("Отображен список сотрудников.")
            elif command.startswith("select "):
                # Разбить команду на части для выделения номера года.
                parts = command.split(maxsplit=1)
                # Запросить работников.
                selected = staff.select(parts[1])
                # Вывести результаты запроса.
                if selected:
```

Результат работы программы:

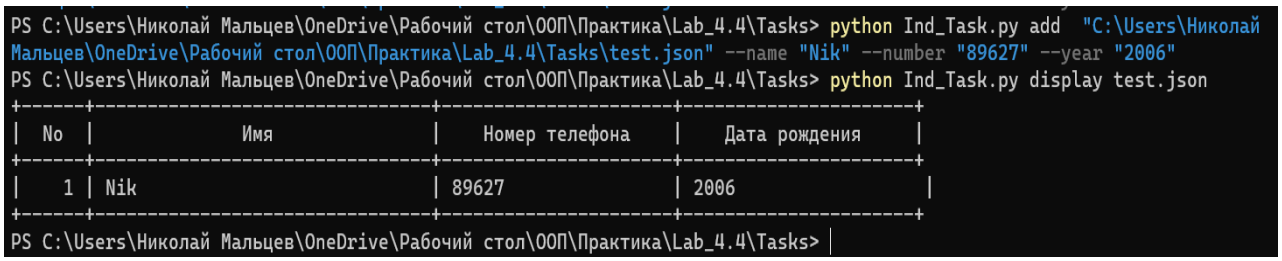


Рисунок 3. Результат работы программы

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 finnally 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 логирования, 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 логирования 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 логирования — по умолчанию используется именно warning.
- Error: этот уровень логирования 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.