

**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

Богдашов А.В. « » _____ 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.

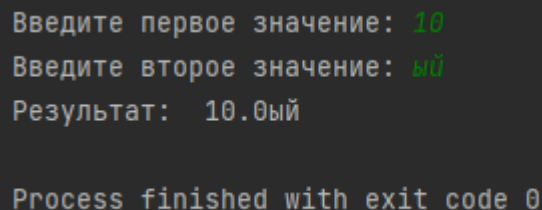
Код программы:

```
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:



```
Введите первое значение: 10
Введите второе значение: 10
Результат: 10.00

Process finished with exit code 0
```

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: 3
Введите количество столбцов: 2
Введите начало диапазона целых чисел: 5
Введите конец диапазона целых чисел: 6
Сгенерированная матрица:
[6, 5]
[6, 6]
[5, 5]

Process finished with exit code 0

```

Рисунок 2. Результат работы программы

Индивидуальное задание.

Код программы:

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

import json
import argparse
import pathlib
import logging
import time

# Настройка логирования
log_format = '%(asctime)s.%(msecs)d - %(levelname)s - %(message)s'
logging.basicConfig(
    level=logging.INFO,
    format=log_format,
    handlers=[
        logging.StreamHandler(),
        logging.FileHandler('flights.log', mode='a') # Логирование в файл с
именем 'flights.log'
    ]
)

# Создать родительский парсер для определения имени файла.
file_parser = argparse.ArgumentParser(add_help=False)
file_parser.add_argument(
    "filename",
    action="store",
    help="Имя файла с данными"
)

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

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

# Создать субпарсер для добавления рейса.
add = subparsers.add_parser(
    "add",
    parents=[file_parser],
    help="Добавить новый рейс"
)
add.add_argument(
    "-d",
    "--destination",
    action="store",
    required=True,
    help="Пункт назначения рейса"
)
add.add_argument(
    "-n",
    "--number",
    action="store",
    type=int,
    required=True,
    help="Номер рейса"
)
add.add_argument(
    "-t",
    "--type",
    action="store",
    required=True,
    help="Тип самолета"
)

```

```

# Создать субпарсер для отображения всех рейсов.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Отобразить все рейсы"
)

# Создать субпарсер для выбора рейсов.
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Выбрать рейсы"
)
select.add_argument(
    "-s",
    "--select",
    action="store",
    required=True,
    help="Необходимый выбор"
)

def add_flight(flights, dst, nmb, tpe):
    try:
        flights.append(
            {
                "destination": dst,
                "number_flight": nmb,
                "type_plane": tpe
            }
        )
        logging.info("Рейс успешно добавлен")
    except Exception as e:
        logging.error(f"Ошибка при добавлении рейса: {e}")
    return flights

def display_flights(flights):
    try:
        for flight in flights:
            print(flight)
    except Exception as e:
        logging.error(f"Ошибка при отображении рейсов: {e}")

def select_flights(flights, t):
    try:
        result = [flight for flight in flights if t in str(flight.values())]
        return result
    except Exception as e:
        logging.error(f"Ошибка при выборе рейсов: {e}")

def save_flights(file_name, flights):
    try:
        with open(file_name, "w", encoding="utf-8") as fout:
            json.dump(flights, fout, ensure_ascii=False, indent=4)
            logging.info("Данные успешно сохранены в файл")
    except Exception as e:
        logging.error(f"Ошибка при сохранении данных в файл: {e}")

def load_flights(file_name):
    try:
        with open(file_name, "r", encoding="utf-8") as fin:
            return json.load(fin)
    except Exception as e:
        logging.error(f"Ошибка при загрузке данных из файла: {e}")

```

```

        return []

def main(command_line=None):
    try:
        start_time = time.time()
        args = parser.parse_args(command_line)
        dst = pathlib.Path(args.filename)
        is_dirty = False
        if dst.exists():
            flights = load_flights(dst)
        else:
            flights = []

        if args.command == "add":
            flights = add_flight(
                flights,
                args.destination,
                args.number,
                args.type
            )
            is_dirty = True

        elif args.command == "display":
            display_flights(flights)

        elif args.command == "select":
            selected_flights = select_flights(flights, args.select)
            display_flights(selected_flights)

        if is_dirty:
            save_flights(dst, flights)

        elapsed_time = time.time() - start_time
        logging.info(f"Время выполнения команды: {elapsed_time:.3f} секунд ")
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == '__main__':
    main()

```

Результат записи в log:

```

PS C:\Games\Программы\CV\1.5\00П\lab4\lab4_4\zadanya> python ind2.py add flights.json -d москва -n 101 -t Boeing737
2024-02-02 22:56:39,412.412 - INFO - Рейс успешно добавлен
2024-02-02 22:56:39,413.413 - INFO - Данные успешно сохранены в файл
2024-02-02 22:56:39,413.413 - INFO - Время выполнения команды: 0.002 секунд

```

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 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 выполнения данной работы были приобретены навыки по обработке исключений и логгированию при написании программ с использованием языка программирования Python версии 3.x.