# МИНИCTEPCTBO 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:

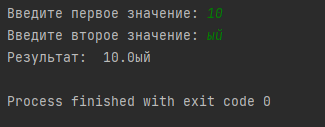
value1 = float(value1) value2 = float(value2) result = value1 + value2

except ValueError:

result = str(value1) + str(value2)

print("Результат: ", result)

Результат работы программы:



Задание 2.

Рисунок 1. Результат работы программы

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:

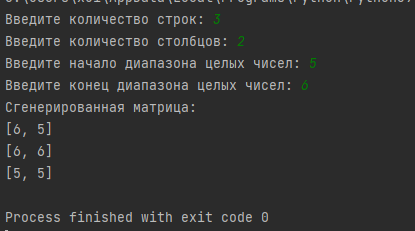


Рисунок 2. Результат работы программы

Индивидуальное задание. Код программы:

#!/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:

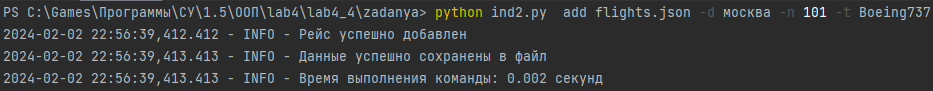


Рисунок 2. Результат логгирования

# 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.

# 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.

# 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.

# Как осуществляется генерация исключений в языке Python?

Для принудительной генерации исключения используется инструкция

raise.

# Как создаются классы пользовательских исключений в языке

**Python?**

Для реализации собственного типа исключения необходимо создать класс, являющийся наследником от одного из классов исключений.

# 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.

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