

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии  
Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**  
**дисциплины «Объектно-ориентированное программирование»**

Выполнил:  
Кожуховский Виктор Андреевич  
3 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной  
техники и автоматизированных систем  
», очная форма обучения

---

(подпись)

Проверил:  
Воронкин Роман Александрович

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2024 г.

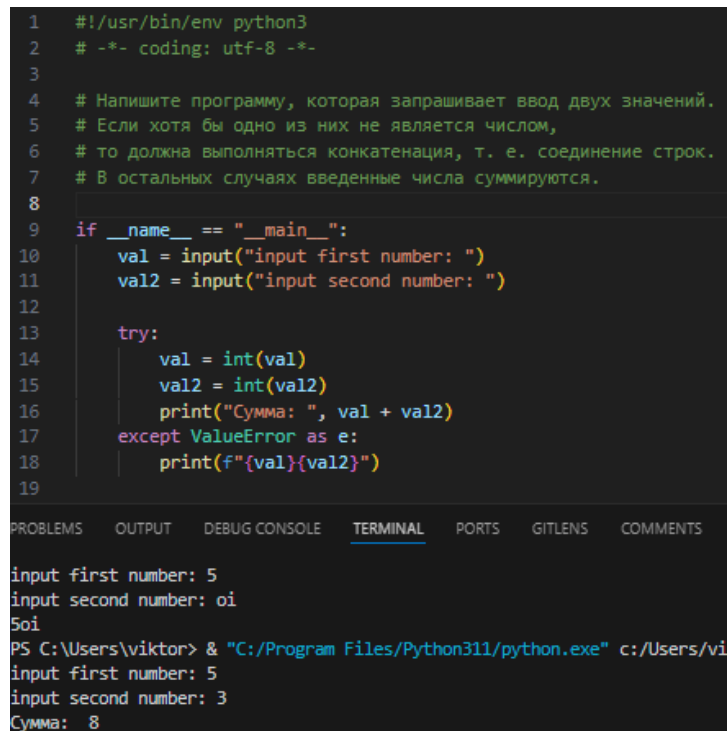
Тема: Работа с исключениями в языке Python

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

Порядок выполнения работы:

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.
3. Выполнил клонирование созданного репозитория.
4. Дополнил файл .gitignore необходимыми правилами для работы с IDE.
5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.
6. Создал проект в папке репозитория.
7. Проработал примеры лабораторной работы.
9. Разработайте программы по следующим описаниям.

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.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Напишите программу, которая запрашивает ввод двух значений.
5  # Если хотя бы одно из них не является числом,
6  # то должна выполняться конкатенация, т. е. соединение строк.
7  # В остальных случаях введенные числа суммируются.
8
9  if __name__ == "__main__":
10     val = input("input first number: ")
11     val2 = input("input second number: ")
12
13     try:
14         val = int(val)
15         val2 = int(val2)
16         print("Сумма: ", val + val2)
17     except ValueError as e:
18         print(f"{val}{val2}")
19
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

```
input first number: 5
input second number: oi
5oi
PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" c:/Users/vi
input first number: 5
input second number: 3
Сумма: 8
```

Рисунок 1. Выполнение общего задания 1

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.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  # Напишите программу, которая будет генерировать матрицу из случайных целых чисел.
5  # Пользователь может указать число строк и столбцов, а также диапазон целых чисел.
6  # Произведите обработку ошибок ввода пользователя.
7
8  from random import randint
9
10
11 def make_matrix(cols, rows, min, max):
12     matrix = [[0]*cols for i in range(rows)]
13     for i in range(rows):
14         for j in range(cols):
15             matrix[i][j] = randint(min, max)
16     return matrix
17
18
19 if __name__ == "__main__":
20     while True:
21         try:
22             cols = int(input("Введите количество столбцов: "))
23             rows = int(input("Введите количество строк: "))
24             min = int(input("Введите минимальное значение: "))
25             max = int(input("Введите максимальное значение: "))
26             if min > max:
27                 raise TypeError("Минимальное значение больше максимального")
28             print(make_matrix(cols, rows, min, max))
29             break
30         except ValueError as e:
31             print("Must be integer")
32         except TypeError as e:
33             print(e)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

Введите количество столбцов: 7  
Введите количество строк: 2  
Введите минимальное значение: j  
Must be integer  
Введите количество столбцов: 7  
Введите количество строк: 3  
Введите минимальное значение: 5  
Введите максимальное значение: 1  
Минимальное значение больше максимального  
Введите количество столбцов: 7  
Введите количество строк: 8  
Введите минимальное значение: 1  
Введите максимальное значение: 5  
[[3, 5, 1, 1, 5, 4, 4], [3, 4, 2, 1, 5, 5, 5], [2, 4, 3, 1, 1, 3, 4], [1, 5, 4, 1, 5, 3, 4], [1, 1, 5, 4, 1], [1, 4, 1, 1, 1, 5, 1]]

Рисунок 2. Выполнение общего задания 2

8. Выполнил индивидуальное задание для варианта 14.

Задание 1

Выполнить индивидуальное задание 1 лабораторной работы 2.19, добавив возможность работы с исключениями и логгирование.

#!/usr/bin/env python3

```

# -*- coding: utf-8 -*-

from pathlib import Path
import argparse
import json
import os
from datetime import datetime
import jsonschema
import logging

logging.basicConfig(level=logging.DEBUG)

person_schema = {
    "type": "object",
    "properties": {
        "name": {"type": "string"},
        "surname": {"type": "string"},
        "date_of_birth": {"type": "string", "format": "date"},
        "zodiac_sign": {"type": "string"}
    },
    "required": ["name", "surname", "date_of_birth", "zodiac_sign"]
}

def validate_person(person_data, schema):
    try:
        jsonschema.validate(person_data, schema)
        return True
    except jsonschema.exceptions.ValidationError as e:
        logging.error(f'Данные человека не соответствуют схеме: {e}')
        return False

def add_person(people, name, surname, date_of_birth, zodiac_sign):
    """
    Добавление нового человека в список.
    Список сортируется по знаку зодиака после добавления нового элемента.
    """
    date_of_birth = datetime.strptime(date_of_birth, '%d.%m.%Y')

    person = {
        'name': name,
        'surname': surname,
        'date_of_birth': date_of_birth,
        'zodiac_sign': zodiac_sign
    }

    people.append(person)
    people.sort(key=lambda item: item.get('zodiac_sign', ''))
    return people

def list_people(people):
    """
    Вывод таблицы людей.
    """
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 20,
        '-' * 20,
        '-' * 15,
        '-' * 13
    )
    print(line)
    print(
        '| {:^4} | {:^20} | {:^20} | {:^15} | {:^12} |'.format(
            "№",
            "Имя",
            "Фамилия",
            "Знак Зодиака",

```

```

        "Дата рождения"
    )
)
print(line)

for idx, person in enumerate(people, 1):
    birth_date_str = person.get('date_of_birth').strftime('%d.%m.%Y')
    print(
        '{:^4} | {:<20} | {:<20} | {:<15} | {:<13} |'.format(
            idx,
            person.get('name', ''),
            person.get('surname', ''),
            person.get('zodiac_sign', ''),
            birth_date_str
        )
    )

print(line)

```

```

def select_people(people, month):
    """
    Вывести список людей, родившихся в заданном месяце.
    """
    count = 0
    for person in people:
        if person.get('date_of_birth').month == month:
            count += 1
            print('{:^4}: {} {}'.format(count, person.get(
                'name', ''), person.get('surname', '')))

    if count == 0:
        print("Люди, родившиеся в указанном месяце, не найдены.")

```

```

def save_people(file_name, staff):
    """
    Сохранить всех работников в файл JSON.
    """
    staff_formatted = [{**person, 'date_of_birth': person.get(
        'date_of_birth').strftime('%d.%m.%Y')} for person in staff]
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        json.dump(staff_formatted, fout, ensure_ascii=False, indent=4)

```

```

def load_people(file_name):
    """
    Загрузить всех людей из файла JSON.
    """
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        staff_loaded = json.load(fin)
        result_people = []
        cnt = 0
        for person in staff_loaded:
            cnt += 1
            if validate_person(person, person_schema):
                try:
                    person['date_of_birth'] = datetime.strptime(
                        person['date_of_birth'], '%d.%m.%Y')
                    result_people.append(person)
                except:
                    logging.error(
                        f"Ошибка при разборе даты в записи, пропуск записи"
                        "{cnt}.")
            else:
                logging.error("Неверные данные человека, пропуск записи.")
        return result_people

```

```

def main():
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )
    # Создание основного парсера.
    parser = argparse.ArgumentParser(description="Управление списком людей")
    subparsers = parser.add_subparsers(dest="command")

    # Создание парсера для добавления человека.
    parser_add = subparsers.add_parser(
        'add', parents=[file_parser], help="Добавить человека")
    parser_add.add_argument("-n", "--name", help="Имя человека")
    parser_add.add_argument("-s", "--surname", help="Фамилия человека")
    parser_add.add_argument(
        "-d", "--date_of_birth", help="Дата рождения (формат ДД.ММ.ГГГГ)")
    parser_add.add_argument("-z", "--zodiac_sign", help="Знак зодиака")

    # Создание парсера для вывода списка людей.
    _ = subparsers.add_parser(
        'list', parents=[file_parser], help="Вывести список людей")

    # Создание парсера для выбора человека по месяцу рождения.
    parser_select = subparsers.add_parser(
        'select', parents=[file_parser], help="Выбрать людей по месяцу рождения")
    parser_select.add_argument(
        "-m", "--month", type=int, help="Месяц рождения")

    # Разбираем аргументы командной строки.
    args = parser.parse_args()

    is_dirty = False

    home_directory = Path.home()
    data_directory = home_directory / 'data'
    # Создаем каталог, если он не существует.
    data_directory.mkdir(exist_ok=True)
    try:
        filename = data_directory / args.filename
    except Exception as e:
        logging.error(
            "Отсутствуют один или несколько обязательных аргументов.")
        exit(1)

    if os.path.exists(filename):
        try:
            people = load_people(filename)
        except Exception as e:
            logging.error(
                "Ошибка при загрузке данных из файла.")
            exit(1)
    else:
        people = []

    # Определяем, какую команду нужно выполнить.
    if args.command == 'add':
        try:
            people = add_person(people, args.name, args.surname,
                                args.date_of_birth, args.zodiac_sign)
            is_dirty = True
        except Exception as e:
            logging.error(
                "Отсутствуют один или несколько обязательных аргументов.")

    elif args.command == 'list':
        list_people(people)

```

```

elif args.command == 'select':
    select_people(people, args.month)

if is_dirty:
    save_people(filename, people)

if __name__ == '__main__':
    main()

```

```

PS C:\Users\viktor\Desktop\ncfu\OOP\OOP-4> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ncfu/OOP/OOP-4/main_3.py
ERROR:root:Отсутствуют один или несколько обязательных аргументов.
PS C:\Users\viktor\Desktop\ncfu\OOP\OOP-4> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ncfu/OOP/OOP-4/main_3.py select
usage: main_3.py select [-h] [-m MONTH] filename
main_3.py select: error: the following arguments are required: filename
PS C:\Users\viktor\Desktop\ncfu\OOP\OOP-4> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ncfu/OOP/OOP-4/main_3.py выдусе
usage: main_3.py [-h] {add,list,select} ...
main_3.py: error: argument command: invalid choice: 'выдусе' (choose from 'add', 'list', 'select')
PS C:\Users\viktor\Desktop\ncfu\OOP\OOP-4> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ncfu/OOP/OOP-4/main_3.py list test.json
ERROR:root:Ошибка при загрузке данных из файла.
PS C:\Users\viktor\Desktop\ncfu\OOP\OOP-4> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ncfu/OOP/OOP-4/main_3.py add -n dasasfd
-s fdsa -d 02.03.2001 test.json
ERROR:root:Отсутствуют один или несколько обязательных аргументов.

```

Рисунок 3. Код решения индивидуального задания 1 и его выполнение

## Задание 2

Изучить возможности модуля logging. Добавить для предыдущего задания вывод в файлы лога даты и времени выполнения пользовательской команды с точностью до миллисекунды.

Изменена только строка `logging.basicConfig(level=logging.DEBUG)` на `logging.basicConfig(level=logging.DEBUG, filename='app.log', encoding='utf-8', format='%(asctime)s - %(levelname)s - %(message)s')`

```

File Edit Format View Help
2024-11-26 10:55:42,672 - ERROR - Отсутствуют один или несколько обязательных аргументов.
2024-11-26 10:57:22,384 - ERROR - Отсутствуют один или несколько обязательных аргументов.
2024-11-26 10:57:48,899 - ERROR - Ошибка при загрузке данных из файла.

```

Рисунок 4. Код решения индивидуального задания 2 и его выполнение

9. Зафиксировал сделанные изменения в репозитории.
10. Выполнил слияние ветки для разработки с веткой master/main.
11. Отправил сделанные изменения на сервер GitHub.

Ссылка: <https://github.com/Viktorkozh/OOP-4>

Контрольные вопросы:

1. Какие существуют виды ошибок в языке программирования Python?

Синтаксические ошибки и исключения.

2. Как осуществляется обработка исключений в языке программирования Python?

С помощью конструкции `try... except` и `finally`.

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.

4. Как осуществляется генерация исключений в языке Python?

Для принудительной генерации исключения используется инструкция `raise`.

5. Как создаются классы пользовательский 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

7. Какие уровни логгирования поддерживаются модулем `logging`?

Приведите примеры, в которых могут быть использованы сообщения с этим уровнем журналирования.

```
logging.basicConfig(level = logging.DEBUG)
```

```
logging.debug("Debug message!")
```

```
logging.info("Info message!")
```

```
logging.warning("Warning message!")
```

```
logging.error("Error message!")
```

```
logging.critical("Critical message!")
```

Вывод: приобрел навыки по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x