

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»
Вариант 20

Выполнила:
Михеева Елена Александровна
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 Р.А.-доцент департамента
цифровых, робототехнических систем и
электроники института перспективной
инженерии

(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:

Ссылка на Github: https://github.com/helendddd/Oop_4.git

1. Была разработана программа для решения следующей задачи:

Необходимо написать 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.

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

if __name__ == '__main__':
    # Запрашиваем первое значение
    first_value = input("Первое значение: ")

    # Запрашиваем второе значение
    second_value = input("Второе значение: ")

    # Проверяем, являются ли оба значения числами
    try:
        # Пытаемся преобразовать оба значения в числа (float)
        num1 = float(first_value)
        num2 = float(second_value)
        # Если оба значения числа, выполняем их суммирование
        result = num1 + num2
        print("Результат:", result)
    except ValueError:
        # Если хотя бы одно из значений не число, выполняем конкатенацию
        result = first_value + second_value
        print("Результат:", result)
```

Рисунок 1. Программа задания №1

```
(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_4 % python3 program/ex1.py
Первое значение: 4
Второе значение: 5
Результат: 9.0
(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_4 % python3 program/ex1.py
Первое значение: a
Второе значение: 9
Результат: a9
```

Рисунок 2. Результат работы программы

2. Была разработана программа для решения следующей задачи:

Необходимо написать 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
5  import random
6
7
8  def generate_matrix(rows, cols, min_val, max_val):
9      """Генерирует матрицу случайных целых чисел."""
10     if rows <= 0 or cols <= 0:
11         raise ValueError("Число строк и столбцов должно быть больше нуля.")
12     if min_val > max_val:
13         raise ValueError("Минимальное значение не может быть больше максимума")
14
15     return [
16         [random.randint(min_val, max_val) for _ in range(cols)]
17         for _ in range(rows)
18     ]
19
20
21 def main():
22     print("Программа для генерации матрицы из случайных целых чисел.")
23     try:
24         rows = int(input("Введите количество строк: "))
25         cols = int(input("Введите количество столбцов: "))
26         min_val = int(input("Введите минимальное значение: "))
27         max_val = int(input("Введите максимальное значение: "))
28
29         matrix = generate_matrix(rows, cols, min_val, max_val)
30         print("Сгенерированная матрица:")
31         for row in matrix:
32             print(row)
33
34     except ValueError as e:
35         print(f"Ошибка ввода: {e}")
36     except Exception as e:
37         print(f"Произошла ошибка: {e}")
38
39
40 if __name__ == "__main__":
41     main()
42
```

Рисунок 3. Программа задания 2

```
Программа для генерации матрицы из случайных целых чисел.
Введите количество строк: 4
Введите количество столбцов: 5
Введите минимальное значение: 2
Введите максимальное значение: 98
Сгенерированная матрица:
[21, 16, 52, 90, 17]
[89, 51, 67, 88, 35]
[46, 91, 65, 76, 38]
[90, 45, 36, 74, 14]
```

Рисунок 4. Результат работы программы

3. Приступили к выполнению индивидуального задания.

Задание: выполнить индивидуальное задание 1 лабораторной работы 2.19, добавив возможность работы с исключениями и логгирование. Изучить возможности модуля logging. Добавить для предыдущего задания вывод в файлы лога даты и времени выполнения пользовательской команды с точностью до миллисекунды.

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 import argparse
5 import json
6 import os.path
7 import pathlib
8 import jsonschema
9 import logging
10
11 # Конфигурация логгера
12 logging.basicConfig(
13     filename='students_program.log',
14     level=logging.INFO,
15     format='%(asctime)s,%(msecs)03d - %(levelname)s - %(message)s',
16     datefmt='%Y-%m-%d %H:%M:%S',
17 )
18
19 def add_student(students, name, group_number, performance):
20     """Функция для добавления нового ученика в список."""
21     try:
22         students.append(
23             {
24                 "name": name,
25                 "group_number": group_number,
26                 "performance": performance,
27             }
28         )
29         logging.info(f"Добавлен новый студент: (name), группа: (group_number)")
30     except Exception as e:
31         logging.error(f"Ошибка при добавлении студента: (e)")
32         raise
33     return students
34
35 def list_students(students):
36     """Функция для вывода списка студентов."""
37     try:
38         if students:
39             line = "%-4s | %-20s | %-20s | %-20s" % ("", "", "", "")
40             print(line)
41             for idx, student in enumerate(students, 1):
42                 print(
43                     "%-4s | %-20s | %-20s | %-20s" % (
44                         idx,
45                         student.get("name", ""),
46                         student.get("group_number", ""),
47                         ", ".join(map(str, student.get("performance", []))),
48                     )
49                 )
50             print(line)
51         else:
52             print("Список студентов пуст.")
53
54 def find_students(students):
55     """Функция для поиска студентов с оценкой 2."""
56     found = []
57     for student in students:
58         if 2 in student["performance"]:
59             found.append(student)
60     if not found:
61         print("Студентов с оценкой 2 не найдено")
62         logging.info("Не найдено студентов с оценкой 2.")
63     else:
64         list_students(found)
65     return found
66 except Exception as e:
67     logging.error(f"Ошибка при поиске студентов: (e)")
68     raise
69
70 def save_file_name(students):
71     """Сохранить всех студентов в файл JSON."""
72     try:
73         with open(file_name, "w", encoding="utf-8") as fout:
74             json.dump(students, fout, ensure_ascii=False, indent=4)
75         logging.info(f"Данные студентов сохранены в файл: (file_name)")
76     except Exception as e:
77         logging.error(f"Ошибка при сохранении данных в файл: (e)")
78         raise
79
80 def load_students(file_name):
81     """Загрузить всех студентов из файла JSON."""
82     schema = {
83         "type": "array",
84         "items": {
85             "type": "object",
86             "properties": {
87                 "name": {"type": "string"},
88                 "group_number": {"type": "string"},
89                 "performance": {
90                     "type": "array",
91                     "items": {"type": "integer"},
92                     "minItems": 5,
93                     "maxItems": 5,
94                 },
95             },
96             "required": ["name", "group_number", "performance"],
97         },
98     }
99     try:
100         with open(file_name, "r", encoding="utf-8") as fin:
101             students = json.load(fin)
102             if not isinstance(students, list):
103                 raise ValueError("Некорректная структура данных")
104             for student in students:
105                 if not isinstance(student, dict):
106                     raise ValueError("Некорректная структура данных")
107                 if "name" not in student or "group_number" not in student or "performance" not in student:
108                     raise ValueError("Некорректная структура данных")
109                 if not isinstance(student["performance"], list):
110                     raise ValueError("Некорректная структура данных")
111                 if len(student["performance"]) != 5:
112                     raise ValueError("Некорректная структура данных")
113                 for mark in student["performance"]:
114                     if not isinstance(mark, int):
115                         raise ValueError("Некорректная структура данных")
116                     if mark < 1 or mark > 5:
117                         raise ValueError("Некорректная структура данных")
118             return students
119     except Exception as e:
120         logging.error(f"Ошибка при загрузке данных из файла: (e)")
121         raise
122
123 def main():
124     parser = argparse.ArgumentParser(
125         description="Программа для работы со студентами",
126         prog="students.py",
127     )
128     parser.add_argument(
129         "-a", "--add", action="store", help="Student's group number"
130     )
131     parser.add_argument(
132         "-g", "--group", action="store", help="Student's group number"
133     )
134     parser.add_argument(
135         "-p", "--performance", action="store", help="Student's performance (list of five marks)"
136     )
137     parser.add_argument(
138         "-d", "--display", action="store", help="Display all students"
139     )
140     parser.add_argument(
141         "-f", "--find", action="store", help="Find the students"
142     )
143     parser.add_argument(
144         "-s", "--save", action="store", help="Save students to file"
145     )
146     parser.add_argument(
147         "-l", "--load", action="store", help="Load students from file"
148     )
149     args = parser.parse_args()
150     if args.add:
151         students = add_student(students, args.name, args.group, args.performance)
152     elif args.group:
153         students = add_student(students, args.name, args.group, args.performance)
154     elif args.display:
155         list_students(students)
156     elif args.find:
157         find_students(students)
158     elif args.save:
159         save_file_name(students)
160     elif args.load:
161         students = load_students(args.load)
162     else:
163         list_students(students)
164
165 if __name__ == "__main__":
166     main()
```

Рисунок 5. Программа индивидуального задания

```
() students.json > ...
1 {
2   {
3     "name": "Иванов И.И.",
4     "group_number": "1234",
5     "performance": [
6       5,
7       4,
8       3,
9       4,
10      5
11    ]
12  },
13  {
14    "name": "Иванова И.И.",
15    "group_number": "1234",
16    "performance": [
17      5,
18      4,
19      5,
20      4,
21      5
22    ]
23  }
24 }
```

ПРОБЛЕМЫ Выходные данные КОНСОЛЬ ОТЛАДКИ **ТЕРМИНАЛ** ПОРТЫ SQL HISTORY TASK MONITOR + Python Debug Console

(venv) (base) elenamiheeva@MacBook-Pro-Elena Oop_4 % python src/gradebook.py add -n "Иванова И.И." -g "1234" -p 5 4 5 4 5 students.json

Рисунок 2. Результат работы программы

Ответы на контрольные вопросы:

1. Какие существуют виды ошибок в языке программирования Python?

В Python выделяют два различных вида ошибок: синтаксические ошибки и исключения.

Синтаксические ошибки возникают в случае, если программа написана с нарушениями *требований Python* к синтаксису. Определяются они в процессе парсинга программы.

Второй вид ошибок – это исключения. Они возникают в случае, если синтаксически программа корректна, но в процессе выполнения возникает ошибка (деление на ноль и т.п.).

2. Как осуществляется обработка исключений в языке программирования Python?

Для блока кода, в котором возможно появление исключительной ситуации необходимо поместить во внутрь синтаксической конструкции `try...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`.

Например:

```
try:
    raise Exception("Some exception")
except Exception as e:
    print("Exception exception " + str(e))
```

5. Как создаются классы пользовательских исключений в языке Python?

В Python классы пользовательских исключений создаются путем определения нового класса, который наследует от стандартного класса исключений. Обычно все исключения в Python являются подклассами встроенного класса Exception

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.

7. Какие уровни логгирования поддерживаются модулем logging? Приведите примеры, в которых могут быть использованы сообщения с этим уровнем журналирования.

Модуль logging в Python поддерживает следующие уровни логгирования:

DEBUG (10) — для отладочной информации, полезной при разработке.

INFO (20) — для обычной информации о ходе работы программы.

WARNING (30) — для предупреждений о возможных проблемах.

ERROR (40) — для ошибок, которые не останавливают программу, но требуют внимания.

CRITICAL (50) — для критических ошибок, приводящих к сбою программы.

Пример:

```
import logging
logging.basicConfig(level=logging.DEBUG)
logging.debug("Отладка")
logging.info("Информация")
logging.warning("Предупреждение")
logging.error("Ошибка")
logging.critical("Критическая ошибка")
```

Каждый уровень фильтрует сообщения по приоритету: чем выше уровень, тем меньше сообщений будет выводиться.

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