ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ Р.МОЛДОВЫ
ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ
ДЕПАРТАМЕНТ ИНФОРМАТИКИ

Калинкова София

Отчет
по дисциплине „ПРОГРАММИРОВАНИЕ В PYTHON”

Руководитель: _____ Плешка Наталья, лектор
(подпись)

Автор: _____
(подпись)

Кишинев, 2024

- **Краткая постановка задачи:**

  2%3 = 2

  Вариант 2. «Учет успеваемости 9А класса, лицея «Гипоталламус»» по математике

  Написанный код должен позволить пользователю выбрать из меню опцию:

  1) Записать данные об успеваемости ученика в файл, если они удовлетворяют всем требованиям;
  2) Вывести среднюю арифметическую оценку по всему классу (сумма всех оценок делится на количество оценок);
  3) Вывести список учеников с оценками меньше 5;
  4) Вывести список учеников, имеющие оценки больше 8;
  5) Выход.

  Каждая строка, записанная в файл будет содержать информацию об ученике и его оценке: фамилия, имя, дата, оценка.
  Проверяется если дата валидна – не раньше 01.09.2023 и не позже текущего дня. Также проверяется чтобы введенные значения соответствовали шаблону дд.мм.гггг, а дд из интервала 01-31 (можно и доп. проверки по месяцам), мм – из интервала 01-12, а год – 2023-2024.

  Проверяется чтобы введенная фамилия и имя состояли из букв. Возможен ввод сложных имен и фамилий, разделенных через тире.

  Оценки могут быть от 1 до 10.

  Я выбрала формат файла txt, потому что этот формат для меня более привычен, также он обеспечивает доступность на всех операционных системах. Это удобно для обработки и хранения данных. Текстовые файлы легко контролировать и отслеживать изменения, так как их можно просматривать и редактировать в любом текстовом редакторе.

- **Логика реализованных алгоритмов:**

1. Запись данных об успеваемости ученика в файл **(input_data**):
    - Программа запрашивает у пользователя фамилию, имя, дату и оценку ученика.
    - Проверяются введенные данные на соответствие заданным шаблонам и допустимым значениям (например, проверяется формат даты, соответствие оценки допустимому диапазону и т.д.).
    - Если данные введены корректно, они записываются в текстовый файл в формате, предусматривающем разделение данных о фамилии, имени, дате и оценке ученика.

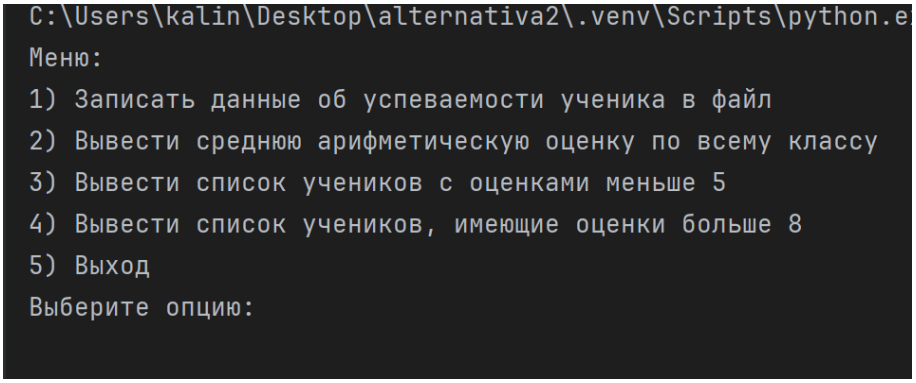
2. Расчет средней арифметической оценки по всему классу (**calculate_grade**):
    - Программа читает данные из файла, выделяет из каждой строки оценку и сохраняет их в список.
    - После чего производится расчет средней арифметической оценки.
    - Результат выводится на экран.

3. Вывод списка учеников с оценками меньше 5 (**grades_below_5)**:
    - Программа считывает данные из файла, проверяет оценки учеников и формирует список тех учеников, у которых оценка меньше 5.
    - Если такие ученики есть, их данные выводятся на экран.

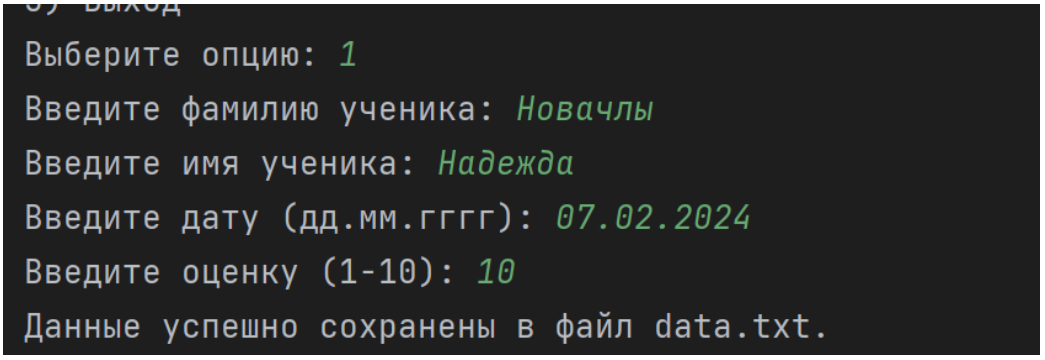
4. Вывод списка учеников с оценками больше 8 **(grades_above_8**):
    - Аналогично предыдущему пункту, программа считывает данные из файла и формирует список учеников с оценками выше 8.
    - Если такие ученики есть, их данные выводятся на экран.

Таким образом, программой предусмотрены проверки вводимых данных на корректность, алгоритмы обработки данных из файла и вывод результатов пользователю в соответствии с выбранными опциями из меню.

- **Скрины интерфейсов для взаимодействия с пользователем и их назначение:**

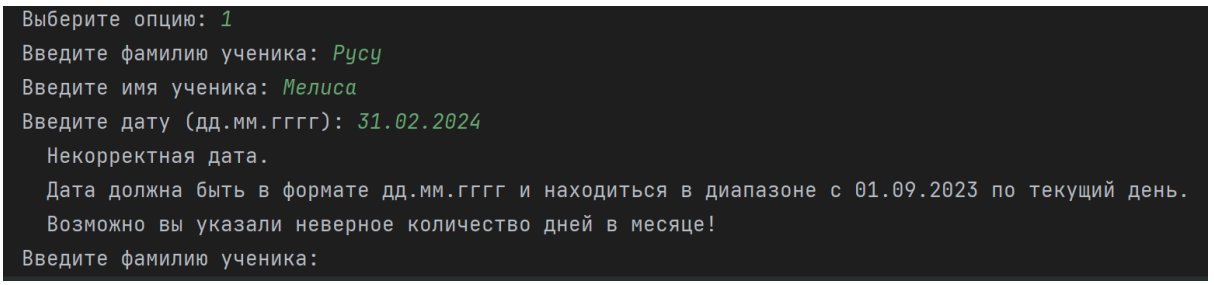


В терминале отображается меню из 5 опций: Записать данные об успеваемости ученика в файл; Вывести среднюю арифметическую оценку по всему классу, Вывести список учеников с оценками меньше 5; Вывести список учеников, имеющие оценки больше 8; Выход.



Выбрав цифру “**1**” начинается ввод данных об ученике, если все поля соответствуют шаблонам, например, в имени может быть только одно тире (это сделано для возможности ввода сложных имен), дата должна быть записана только в формате дд.мм.гггг и т.д, то строка содержащая данные о ученике сохранится в файл с именем “**data.txt**” и выводится соответствующее сообщение в терминале,

В обратном случае выводится сообщение о том что ввод некорректен и ввод начнется повторно:



Валидные данные записанные файл выглядят следующим образом:

File Edit View

```
Калинкова       София          01.09.2023  9
Пашева          Мария-Магдалина 02.09.2023  9
Ciobanu         Ion            12.11.2023  4
Ciobanu         Ion            12.12.2023  10
Новачлы         Надежда        07.02.2024  10
Rusu            Melisa         31.03.2024  7
```

Если пользователь выбирает “2”, то выводится средняя оценка, т.е рассчитывается среднее арифметическое всех оценок из файла:

```
Выберите опцию: 2
Средняя оценка класса: 8.17
```

При выборе “3” в терминал выводится список учеников с оценкой ниже 5:

```
Выберите опцию: 3
Список учеников с оценками меньше 5:
Ciobanu Ion
```

При выборе “4” на экран выводится список учеников у которых оценка выше 8:

```
Выберите опцию: 4
Список учеников с оценками больше 8:
Калинкова София
Пашева Мария-Магдалина
Ciobanu Ion
Новачлы Надежда
```

- **Структуры данных, с которыми работали в приложении(использовались для хранения данных):**

В данном приложении используются следующие структуры данных для хранения информации об учениках и их успеваемости:

- Списки:
    - В программе используются списки для временного хранения данных об учениках и их оценках.
    - Например, при чтении данных из файла в функциях **calculate_grade(), grades_below_5(), grades_above_8()** данные оценок учеников временно хранятся в списках для последующего анализа и обработки.
- Строки:
    - Вся информация о каждом ученике хранится в виде строки в текстовом файле.
    - При чтении данных из файла каждая строка разбивается на отдельные значения (фамилия, имя, дата, оценка) для дальнейшей обработки.

- **Описание функционала мини-приложения, строками кода (кусками) и пояснениями:**

  Моё мини-приложение по управлению данными учеников включает в себя несколько основных функций, каждая из которых отвечает за определенные операции с данными.

```
def validate_date(date_str):
  try:
    date_obj = datetime.datetime.strptime(date_str, '%d.%m.%Y')
    start_date = datetime.datetime(2023, 9, 1)
    current_date = datetime.datetime.now()
    return start_date <= date_obj <= current_date
  except ValueError:
    return False
```

  Функция **validate_date** проверяет, соответствует ли введенная дата формату "дд.мм.гггг" и находится ли она в допустимом диапазоне. Она преобразует строку **date_str** в объект даты и затем сравнивает ее с датой начала диапазона (1 сентября 2023 года) и текущей датой. Если дата попадает в этот диапазон, функция возвращает **True**, в противном случае - **False**.

```
def input_data():
  while True:
```

```
    surname = input("Введите фамилию ученика: ")
    if not re.match(name_pattern, surname):
        print("  Некорректная фамилия. Убедитесь, что фамилия написана правильно.")
        continue
```

Это часть функции **input_data():** запрашивает у пользователя ввод фамилии ученика. Затем она проверяет, соответствует ли введенная фамилия заданному шаблону с помощью функции **re.match(name_pattern, surname)**. Если введенная фамилия не соответствует шаблону, выводится сообщение об ошибке, и программа продолжает ожидать корректного ввода. Если фамилия введена корректно, цикл завершается, и выполнение программы переходит к следующим действиям.

```
formatted_data = f"{surname:<30}\t{name:<30}\t{date:<15}\t{grade:<5}"
with open("data.txt", mode='a', encoding='utf-8') as f:
  f.write(formatted_data + "\n")
print("Данные успешно сохранены в файл data.txt.")
break
```

Эта часть кода из функции **input_data():** Если поле name, surname, date и grade соответствуют шаблонам, то создается отформатированная строка, где каждый элемент данных (фамилия, имя, дата и оценка) выравнивается по определенной ширине с помощью спецификаторов формата **<**, чтобы обеспечить равномерное отображение. Например, **{surname:<30}** означает, что фамилия будет выровнена по левому краю и займет не менее 30 символов.

Затем эта отформатированная строка добавляется в текстовый файл "**data.txt**" с помощью операции записи. Режим **'a'** указывает на добавление данных в конец файла.

После записи данных в файл выводится сообщение о успешном сохранении, и цикл завершается с помощью оператора **break**.

```
def calculate_grade():

  with open("data.txt", mode="r", encoding='utf-8') as f:
```

```
    grades = [int(line.split()[-1]) for line in f.readlines() if line.strip()]
    if grades:
        average = sum(grades) / len(grades)
        print(f"Средняя оценка класса: {average:.2f}")
    else:
        print("Нет данных для расчета.")
```

Эта функция открывает файл **"data.txt"** для чтения и считывает оценки учеников из каждой строки файла. Затем она вычисляет среднюю арифметическую оценку по всем ученикам. Если в файле есть данные об оценках, то функция вычисляет сумму всех оценок и делит ее на количество оценок, чтобы получить среднее значение. Результат выводится на экран с двумя знаками после запятой, указывая среднюю оценку класса. Если данных об оценках в файле нет, функция выводит сообщение о том, что нет данных для расчета.

```
def grades_below_5():
  with open("data.txt", mode="r", encoding='utf-8') as f:
    students = [line.strip().split()[:2] for line in f if int(line.split()[-1]) < 5]
    if not students:
        print("Учеников с оценками меньше 5 нет.")
    else:
        print("Список учеников с оценками меньше 5:")
        for student in students:
            print(' '.join(student))
```

функция **grades_below_5()** открывает файл "data.txt" для чтения и проверяет оценки учеников. Если оценка ученика ниже 5, его фамилия и имя добавляются в список **students**. После этого функция проверяет, есть ли ученики с оценками ниже 5 в списке. Если такие ученики есть,

они выводятся на экран вместе с сообщением "Список учеников с оценками меньше 5:". Если учеников с оценками ниже 5 нет, выводится сообщение о том, что их нет.

```
def grades_above_8():
    with open("data.txt", mode="r", encoding='utf-8') as f:
        students = [line.strip().split()[:2] for line in f if int(line.split()[-1]) > 8]


        if not students:
            print("Учеников с оценками больше 8 нет.")
        else:
            print("Список учеников с оценками больше 8:")
            for student in students:
                surname, name = student
                print(f"{surname} {name}")
```

функция **grades_above_8()** открывает файл "data.txt" для чтения и создает список **students**, содержащий фамилии и имена учеников с оценками выше 8. Затем она проверяет, есть ли такие ученики в списке. Если есть, выводит сообщение "Список учеников с оценками больше 8:" и перечисляет их фамилии и имена. Если учеников с оценками выше 8 нет, выводит сообщение о том, что их нет.

- **Вывод:**

После реализации задания с данными об успеваемости учеников стало очевидно, что Python предоставляет мощные инструменты для работы с файлами и обработки данных. Работа с текстовыми файлами упростила сохранение и обработку данных, а встроенные модули **datetime** и **re** обеспечили удобные средства для работы с датами и выполнения проверок вводимых данных. Использование list comprehension и

методов работы со строками позволило компактно и эффективно обрабатывать данные из текстового файла. Реализованный функционал для вычисления средней оценки по классу и вывода списков учеников с оценками ниже 5 и выше 8 предоставил полезные инструменты для анализа успеваемости класса и выявления студентов, требующих дополнительной поддержки или выделения.

- **Список используемой литературы:**

Презентации на Moodle
сайт https://pythonworld.ru/osnovy
сайт https://ravesli.com
различные видеоуроки YouTube