**МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет Математики и Информатики**

**Департамент Информатики**

Шелестян Анастасия

Отчет индивидуальной работы

по дисциплине «Программирование в PYTHON»

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Плешка Наталья, лектор

(подпись)

Автор: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шелестян Анастасия

(подпись)

Кишинев, 2024

Оглавление

[Цель работы. 3](#_Toc164983772)

[Логика реализованных алгоритмов. 3](#_Toc164983773)

[Выполнение задания: 4](#_Toc164983774)

[Main.py 4](#_Toc164983775)

[Ioemployees.py 7](#_Toc164983776)

[Analysedata.py 9](#_Toc164983777)

[Результат программы: 12](#_Toc164983778)

[Выводы. 14](#_Toc164983779)

[Библиография 14](#_Toc164983780)

# Цель работы.

# Цель данной работы — разработать графический пользовательский интерфейс (GUI) для ввода и просмотра данных о сотрудниках с использованием библиотеки Dear PyGui. Программа должна позволять пользователям вводить данные о сотрудниках, такие как фамилия, имя, отдел и зарплата, проверять их на корректность, сохранять эти данные в файл и предоставлять возможность просматривать сохраненные данные.

# Логика реализованных алгоритмов.

Для того чтоб выполнить поставленную задачу необходимо следующее:

* 1. Импорт библиотек:

Импортируются библиотеки re для работы с регулярными выражениями и dearpygui.dearpygui для создания интерфейса.

Функция input\_data:

Считывание данных из полей ввода.

Проверка данных на корректность с помощью регулярных выражений и проверки диапазона значений для зарплаты.

В случае ошибки валидации — отображение сообщения об ошибке в текстовом поле.

Если данные валидны, они записываются в файл data.txt.

Очищение полей ввода и текстового поля ошибок после успешного ввода данных.

Функция view\_data:

Открытие файла data.txt и считывание данных.

Формирование строк для отображения в текстовом поле.

Обновление текстового поля в GUI с данными сотрудников.

В случае отсутствия файла — отображение сообщения об ошибке.

Создание окон и элементов интерфейса:

Создание окна для ввода данных о сотрудниках с текстовым заголовком, полями ввода для фамилии, имени, отдела и зарплаты, кнопкой "Submit" и текстовым полем для отображения ошибок.

Создание окна для просмотра данных с кнопкой "Refresh Data" для обновления и текстовым полем для отображения списка сотрудников.

Запуск GUI:

Создание контекста и вьюпорта для Dear PyGui.

Настройка и отображение вьюпорта.

Запуск основного цикла Dear PyGui.

Завершение работы и уничтожение контекста.

# Выполнение задания:

Для начала создаю документа с разрешением .ру. Первый для самого меню программы, второй для созданных функций ввода и вывода, третий для созданных функций для анализа нашей программы. Также у нас создается текстовый документ в который и будут сохраняться данный введенные пользователь с дальнейшим выводом информации на экран.

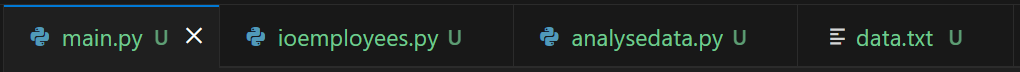
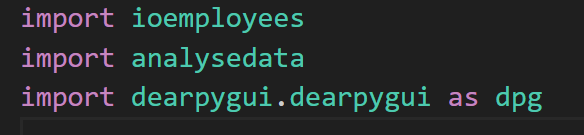


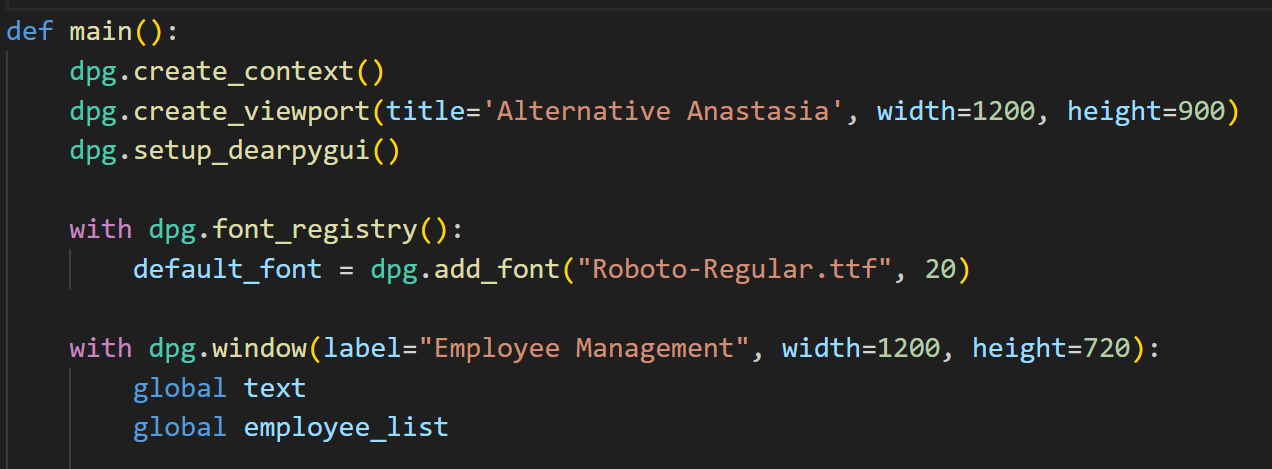
Рис. 1

# Main.py

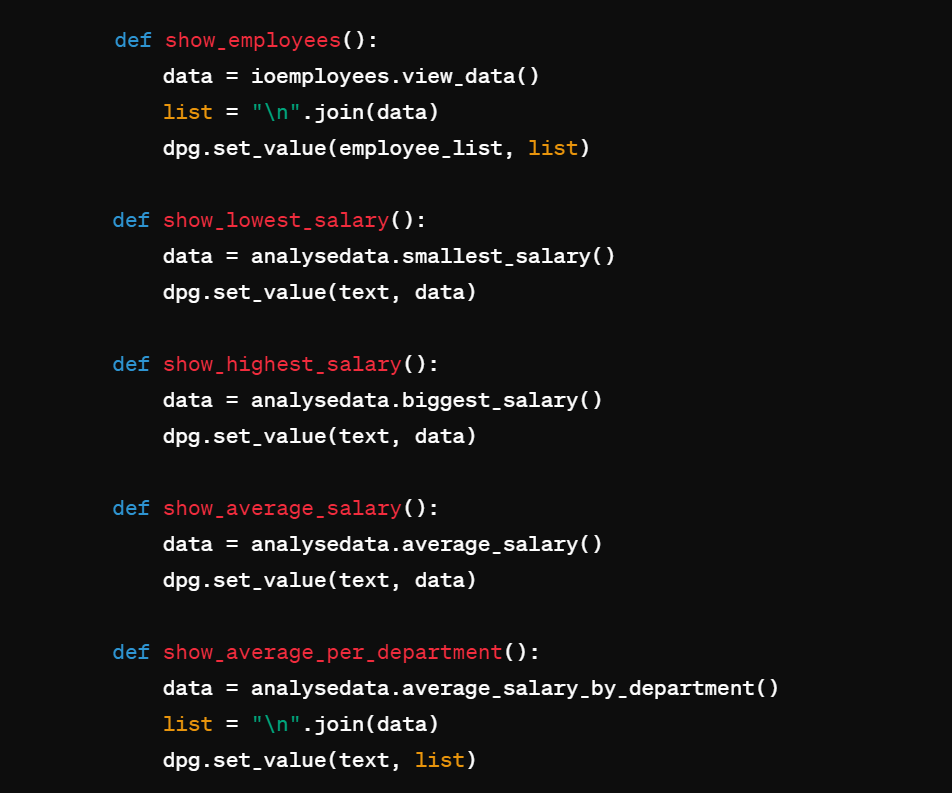
1. **Импорт модулей:**
   * Импортируются модули **ioemployees** и **analysedata**, которые содержат функции для ввода и анализа данных соответственно.
   * Импортируется библиотека **dearpygui.dearpygui** для создания графического интерфейса.



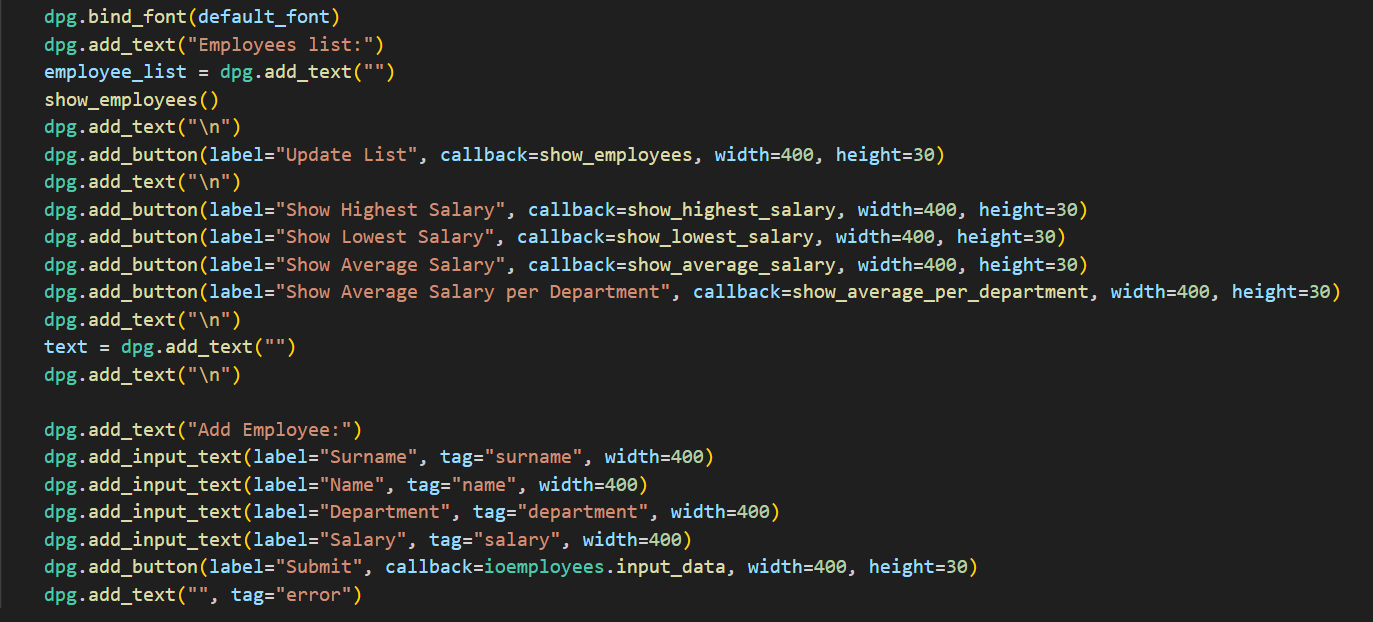
1. **Основная функция main():**
   * Создание контекста и вьюпорта Dear PyGui.
   * Установка шрифта для интерфейса.
   * Создание основного окна управления данными сотрудников.



1. **Функции для отображения данных:**
   * **show\_employees()**: Загружает список сотрудников из модуля **ioemployees** и отображает его в текстовом элементе GUI.
   * **show\_lowest\_salary()**: Показывает сотрудника с самой низкой зарплатой.
   * **show\_highest\_salary()**: Показывает сотрудника с самой высокой зарплатой.
   * **show\_average\_salary()**: Показывает среднюю зарплату всех сотрудников.
   * **show\_average\_per\_department()**: Показывает среднюю зарплату по отделам.



1. **Создание элементов интерфейса:**
   * Связывание шрифта с интерфейсом.
   * Добавление текстовых элементов для отображения списка сотрудников и кнопок для обновления списка и выполнения анализа данных.
   * Добавление полей ввода для добавления нового сотрудника и кнопки для отправки данных.



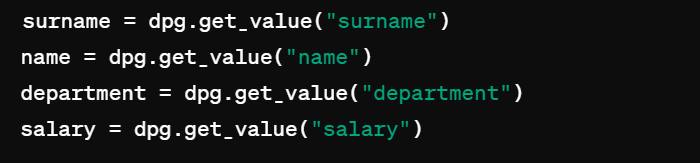
1. **Запуск и завершение GUI:**
   * Показ вьюпорта и запуск основного цикла Dear PyGui.
   * Уничтожение контекста при завершении работы.



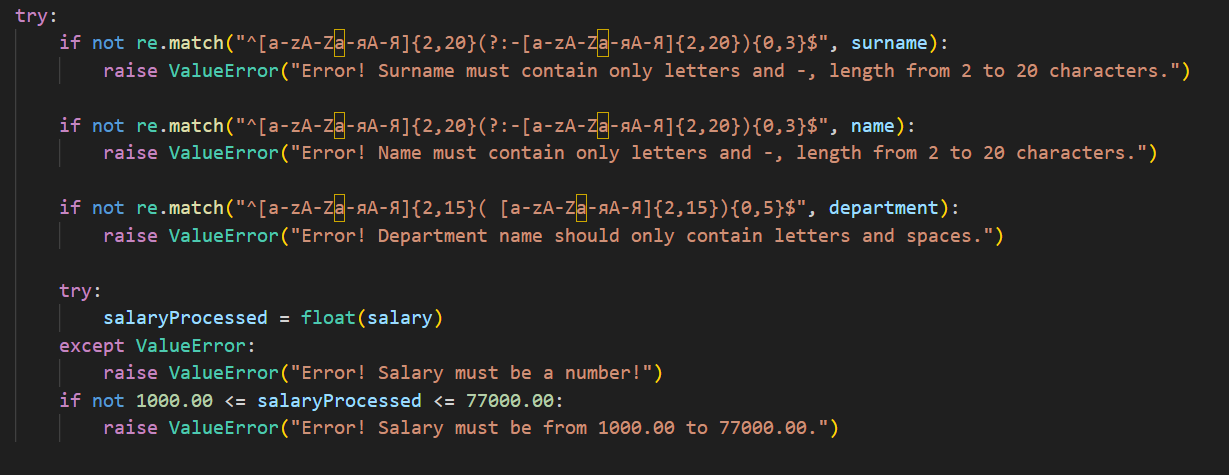
# Ioemployees.py

1. **Получение значений из полей ввода**:
   * **dpg.get\_value("surname")**: получает значение из поля ввода фамилии.
   * **dpg.get\_value("name")**: получает значение из поля ввода имени.
   * **dpg.get\_value("department")**: получает значение из поля ввода отдела.
   * **dpg.get\_value("salary")**: получает значение из поля ввода зарплаты.

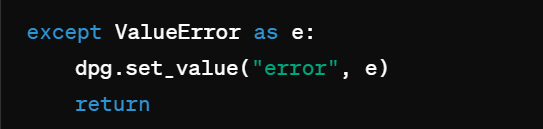
python



1. **Проверка введенных данных**:
   * Проверка фамилии: должна содержать только буквы и дефисы, длина от 2 до 20 символов.
   * Проверка имени: аналогично фамилии.
   * Проверка отдела: должно содержать только буквы и пробелы, каждая часть длиной от 2 до 15 символов.
   * Проверка зарплаты: должно быть числом в диапазоне от 1000.00 до 77000.00.



1. **Обработка ошибок**:
   * Если проверка не пройдена, выбрасывается исключение **ValueError** и сообщение об ошибке отображается в GUI.

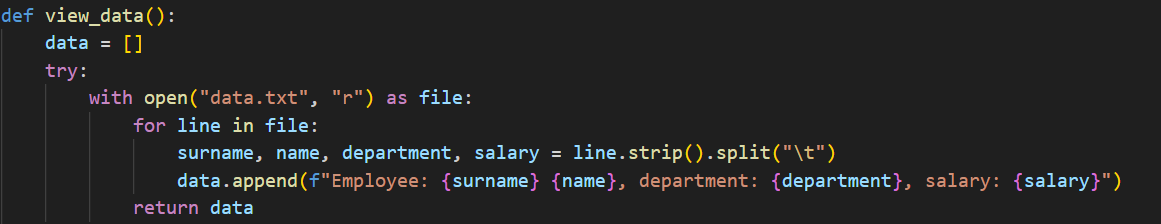


1. **Сохранение данных**:
   * Если все данные валидны, они записываются в файл **data.txt**.
   * Очищаются поля ввода и сообщение об ошибке в GUI.

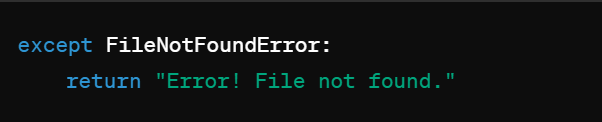


**Функция view\_data**

1. **Чтение данных из файла**:
   * Открывается файл **data.txt** для чтения.
   * Каждая строка файла разбивается на части: фамилия, имя, отдел и зарплата.
   * Формируется список строк для отображения в формате: **Employee: фамилия имя, department: отдел, salary: зарплата**.



1. **Обработка ошибки**:
   * Если файл не найден, возвращается сообщение об ошибке.



# Analysedata.py

Следующий файл это Analysedata.py где код содержит четыре функции для анализа данных о сотрудниках компании.

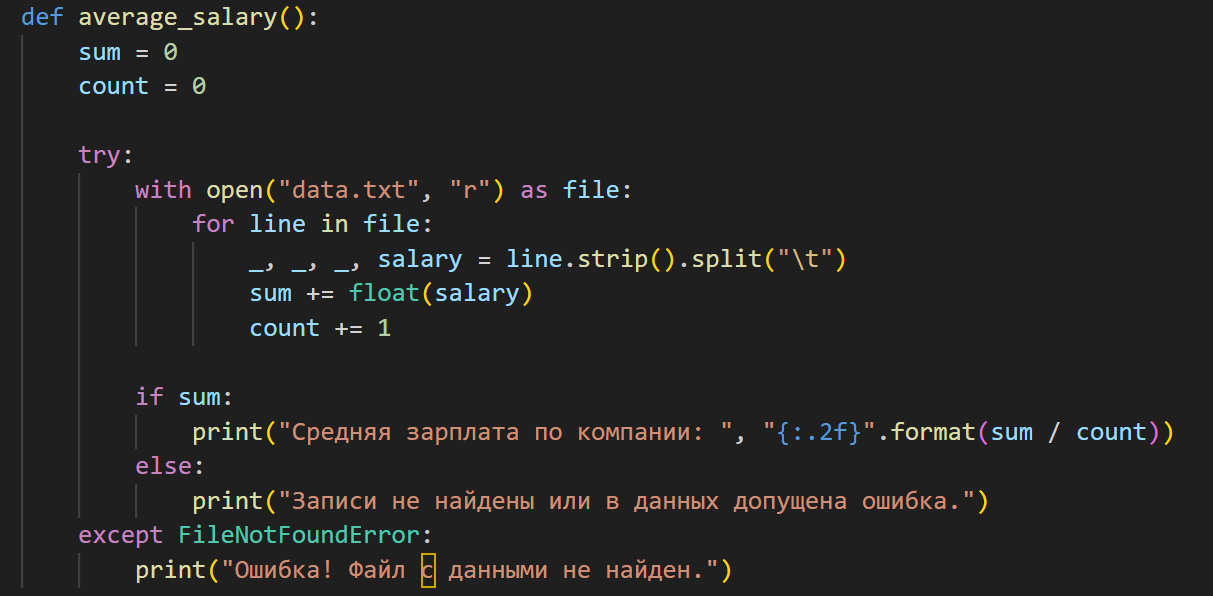


Рис. 6

**Функция average\_salary()** Рассчитывает среднюю зарплату по компании.

В этой функции переменные sum и count используются для расчета общей суммы зарплат и количества сотрудников.

Используется блок try-except для обработки возможных ошибок при чтении файла. Если файл не найден, выводится сообщение об ошибке.

Функция открывает файл "data.txt" и считывает строки. Затем извлекает значение зарплаты из каждого раздела и добавляет его в переменную sum, увеличивая переменную count.

Если данные корректно считаны и есть хотя бы один сотрудник, выводится средняя зарплата, рассчитанная как sum / count. В противном случае, выводится сообщение об отсутствии записей или наличии ошибок.

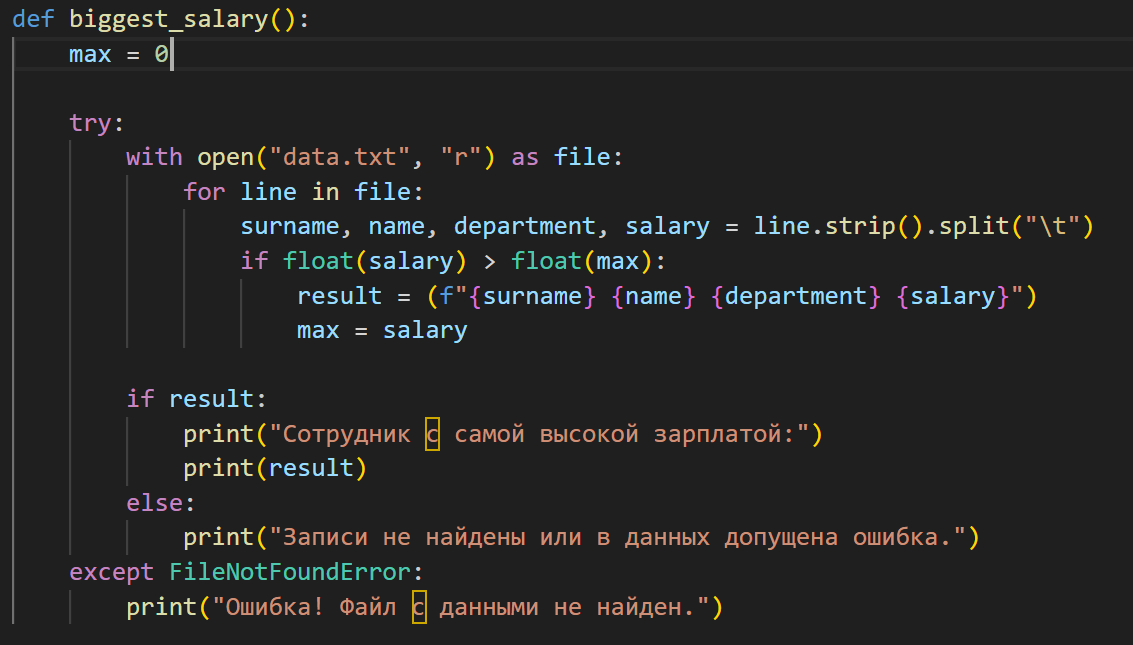


Рис. 7

**Функция biggest\_salary()** Находит сотрудника с самой высокой зарплатой.

Переменная max используется для отслеживания максимальной зарплаты. Изначально установлена в 0.

Функция читает файл "data.txt" и для каждого сотрудника сравнивает его зарплату с переменной max. Если зарплата больше, обновляет max и сохраняет данные сотрудника в переменную result.

Если result не пуст, отображается информация о сотруднике с самой высокой зарплатой.

Если нет данных или возникают ошибки, выводится соответствующее сообщение.

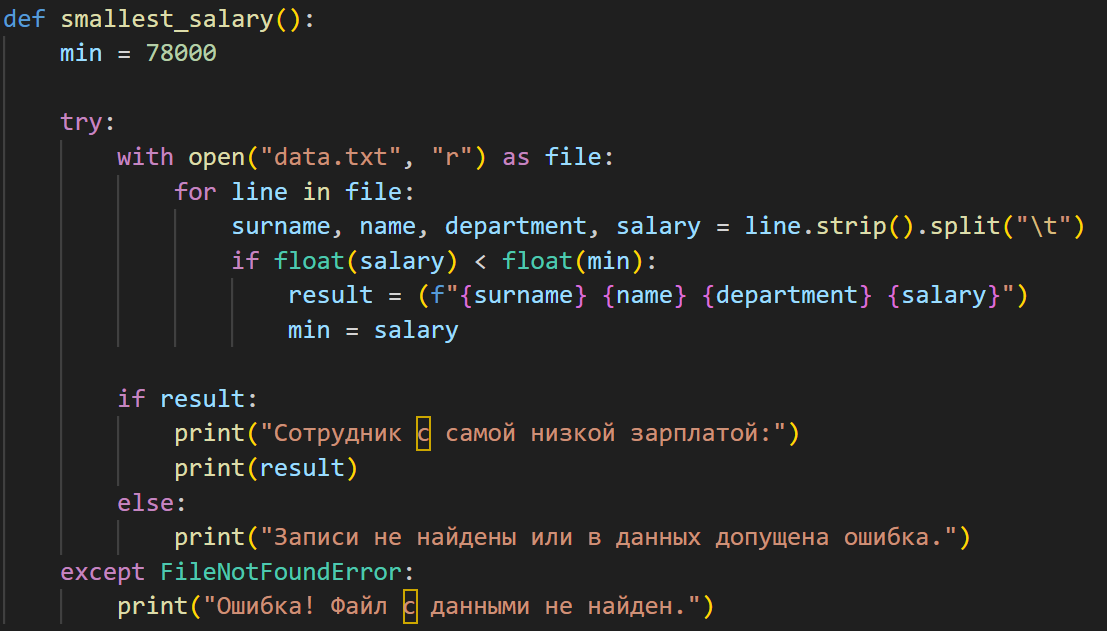


Рис. 8

**Функция smallest\_salary()** Находит сотрудника с самой низкой зарплатой.

Описание:

Переменная min используется для отслеживания минимальной зарплаты. Установлена в 78000 (максимальная допустимая зарплата плюс 1000).

Функция читает файл "data.txt" и ищет зарплату ниже переменной min. Если находит такую, обновляет min и сохраняет данные сотрудника в переменную result.

Если result не пуст, отображается информация о сотруднике с самой низкой зарплатой.

Если возникают ошибки или файл не найден, выводится соответствующее сообщение.

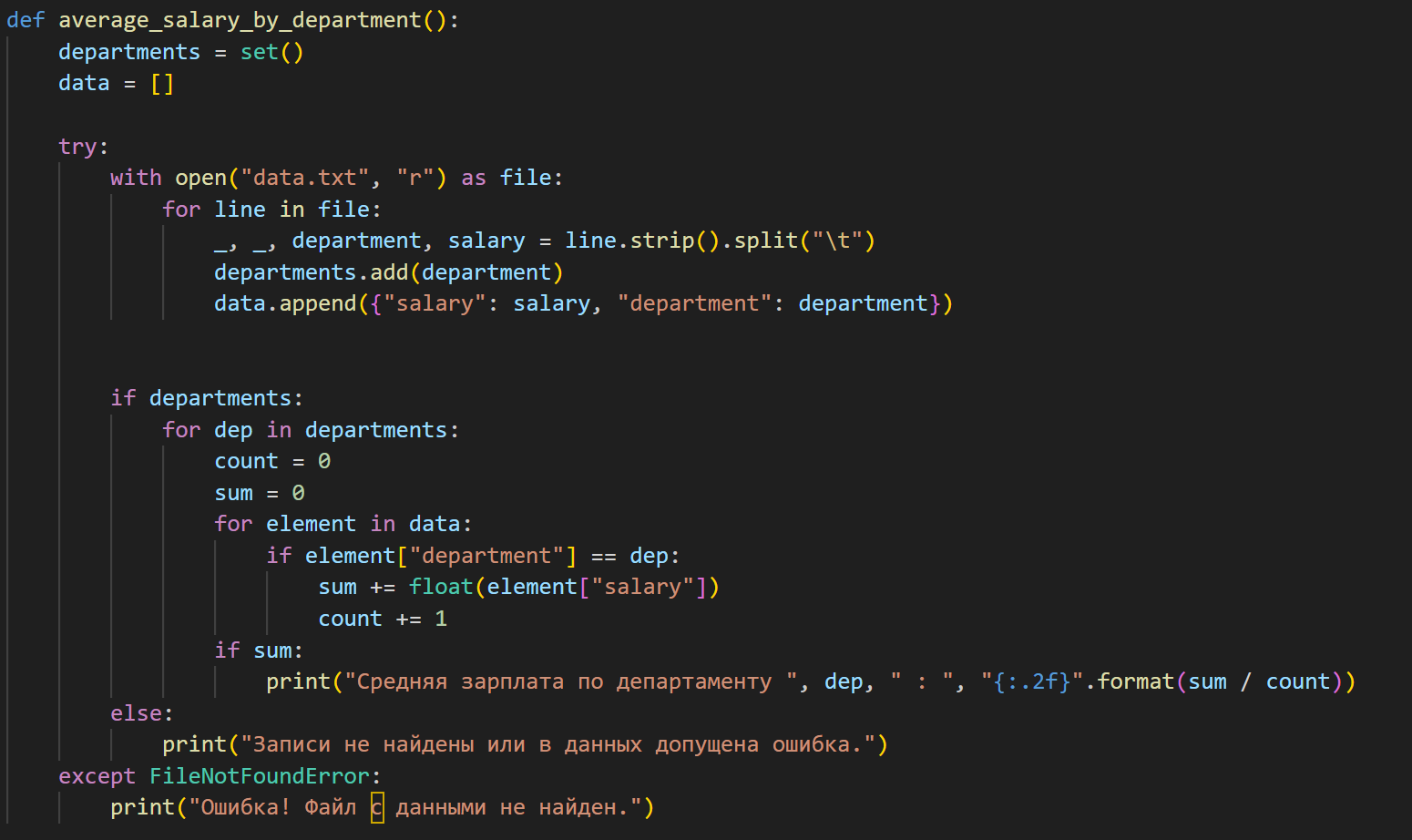


Рис. 9

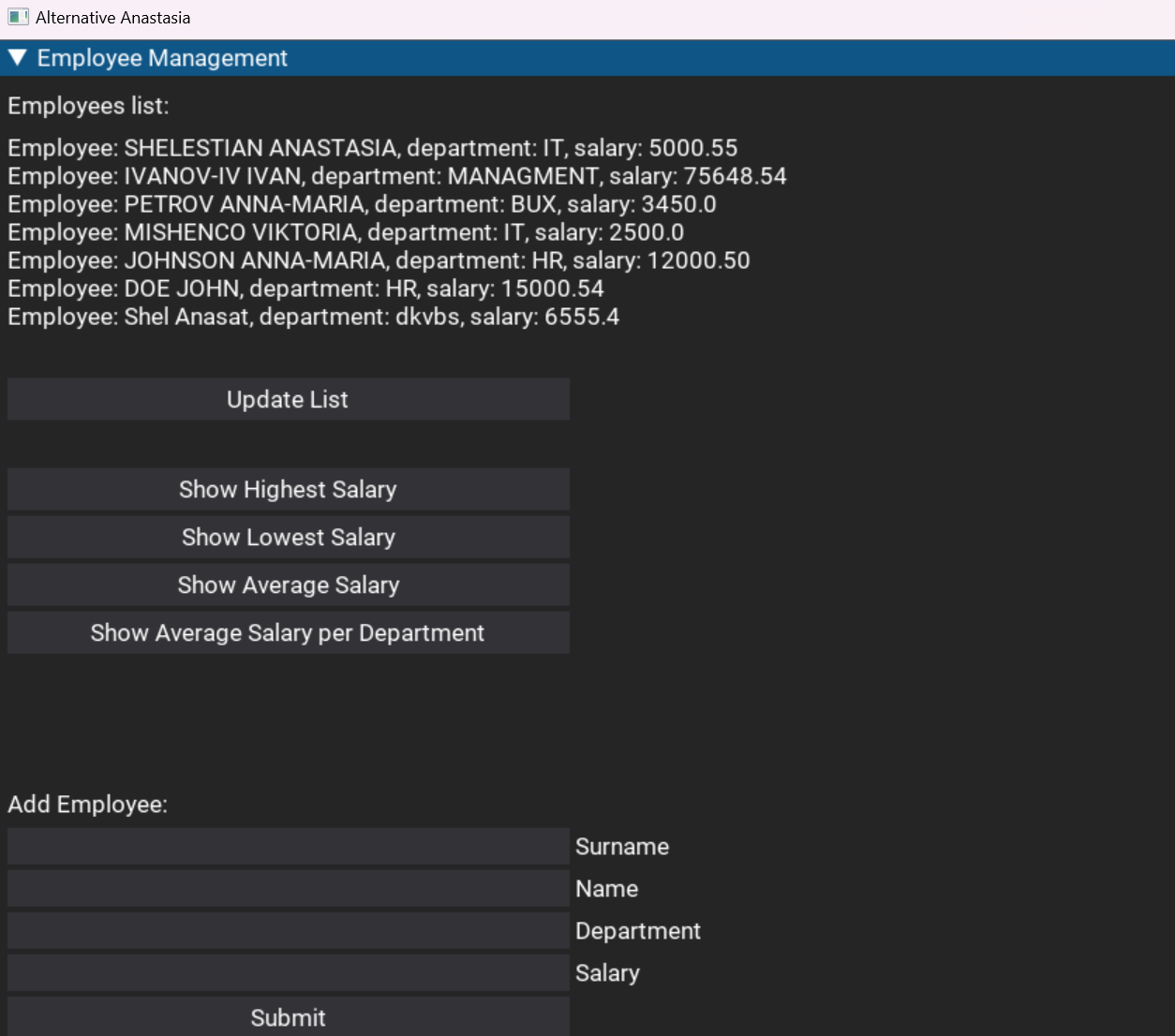
**Функция average\_salary\_by\_department()** Рассчитывает среднюю зарплату по каждому департаменту.

Собирает все уникальные департаменты в departments (используется set для уникальности) и собирает информацию о сотрудниках в список data.

Для каждого департамента dep, функция рассчитывает общую зарплату и количество сотрудников в этом департаменте. Если департаментов не найдено или возникают ошибки, выводится соответствующее сообщение.

Если найдены данные, для каждого департамента отображается средняя зарплата.

# Результат программы:



# Выводы.

Язык Python предоставляет удобные средства для решения задач. Данная программа хорошо структурирована, с разделением на отдельные функции для разных задач. Это упрощает сопровождение и расширение программы, а также повышает читаемость кода. Использование отдельного блока для основного меню позволяет пользователю легко взаимодействовать с программой, выбирая нужные операции из меню.

Функции ввода данных используют регулярные выражения для валидации входных данных, гарантируя, что фамилия, имя и название департамента соответствуют заданным критериям. Это предотвращает ввод некорректных данных и снижает вероятность ошибок при анализе.

Программа предоставляет базовые функции анализа данных, такие как расчёт средней зарплаты, нахождение сотрудников с самой высокой и самой низкой зарплатой, а также расчёт средней зарплаты по департаментам. Эти функции используют простые методы обработки данных, что обеспечивает эффективность при работе с небольшими объёмами данных. Также работает с текстовым файлом для хранения данных о сотрудниках. Используется режим добавления ("append"), что позволяет добавлять новые записи, не перезаписывая существующие. Это обеспечивает сохранность данных между запусками программы.

# Библиография

1. Материалы на сайте курса (https://moodle.usm.md)

2. https://www.python.org/