| МИНОБРНАУКИ РОССИИ | | |
| --- | --- | --- |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №1  по дисциплине «Языки программирования»  Вариант 10 | | |
|  | | Работу выполнил  студент группы ПМИ-9,10-2023 1 курса  Румянцев В.И.  «18» Январь 2024 г. |
| Работу проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И.О.  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г. |
| Пермь 2024 | | |

**СОДЕРЖАНИЕ**

[Постановка задачи](#_heading=h.30j0zll) 3

[Алгоритм решения](#_heading=h.3znysh7) 4

[Тестирование задачи](#_heading=h.1bo9hebu7opy) 6

[Код программы](#_heading=h.tyjcwt) 14

# Постановка задачи

Создать файл записей, в котором хранится информация о дисциплинах, читаемых на факультете: название, с какого семестра читается, продолжительность курса (в семестрах), общее количество часов, вид отчётности (зачёт, экзамен), читающая курс кафедра Разработать и реализовать программу "Учебный план", которая считывает исходную информацию и позволяет на основе неё создавать следующие отчёты:

1. Полный список всех дисциплин, который будет отсортирован следующему ключу: с какого семестра читается (по возрастанию) + читающая кафедра (по возрастанию) + общее количество часов (по убыванию).

2. Список всех дисциплин с заданным видом отчётности (вводится с клавиатуры), отсортированный по следующему ключу: продолжительность курса (по возрастанию) + общее количество часов (по убыванию).

3. Список всех дисциплин с общим количеством часов от N1 до N2 (вводится с клавиатуры), отсортированный по следующему ключу: читающая кафедра (по возрастанию) + общее количество часов (по убыванию).

Создать базу дисциплин, включающую не менее 25 записей и на основе неё сформировать все указанные списки (можно использовать вложенные списки и словари) и на основе неё сформировать все указанные списки). Взаимодействие с базой данных обеспечивается отдельными функциями – добавление/удаление/изменение записи, вывода на экран всей базы и отдельно сформированных списков. Интерфейс пользователя удобнее организовать в виде меню с выбором соответствующего действия. В программе обязательно наличие дружественного интерфейса и защиты от некорректного пользовательского ввода. База должна содержать такие записи, чтобы во всех списках явно прослеживался заданный вид сортировки по всем ключам. Для сортировки записей использовать сортировку бинарными вставками.

# Алгоритм решения

1. Создание класса Discipline

- name - название дисциплины

- semester - номер семестра, с которого начинается чтение дисциплины

- duration - продолжительность курса (в семестрах)

- total\_hours - общее количество часов

- assessment - вид отчётности (зачёт, экзамен)

- department - кафедра, читающая дисциплину

2. Создание и реализация функции read\_data\_from\_file()

- Считывает данные из файла (базы данных)

- Создает объекты типа Discipline на основе полученных данных

- Возвращает считанные данные в виде массива объектов типа Discipline

3. Создание и реализация функции write\_data\_to\_file(data)

- Записывает данные в файл (базу данных), согласно заранее определённому формату ведения синтаксиса в базе данных

4. Создание и реализация функции binary\_insertion\_sort(arr, lam=lambda x: x)

- Алгоритм сортировки бинарными вставками, использующий заданную пользователем функцию lam для определения ключа сортировки

5. Создание и реализация функций добавления, удаления и редактирования дисциплин

- add\_discipline(database), remove\_discipline(database), edit\_discipline(database)

- Взаимодействуют с пользователем, запрашивая информацию о новой дисциплине, удаляя или редактируя существующую

6. Создание и реализация функции display\_database(database)

- Отображает список всех дисциплин в базе данных

- Позволяет пользователю получить подробную информацию о выбранной дисциплине

7. Создание и реализация функции sort(database)

- Реализует функциональность "Учебного плана"

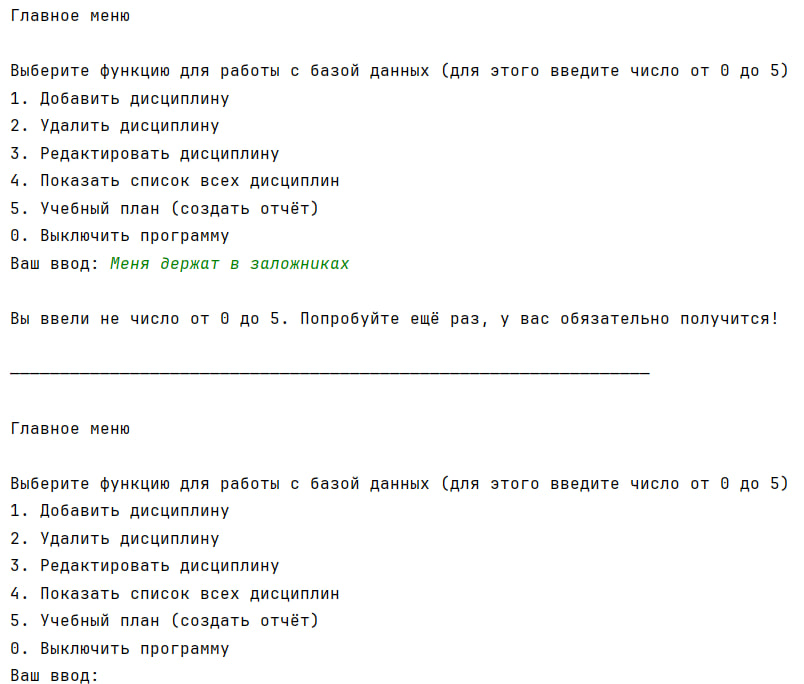
- Запрашивает у пользователя тип отчета и соответствующие параметры для создания отчетов

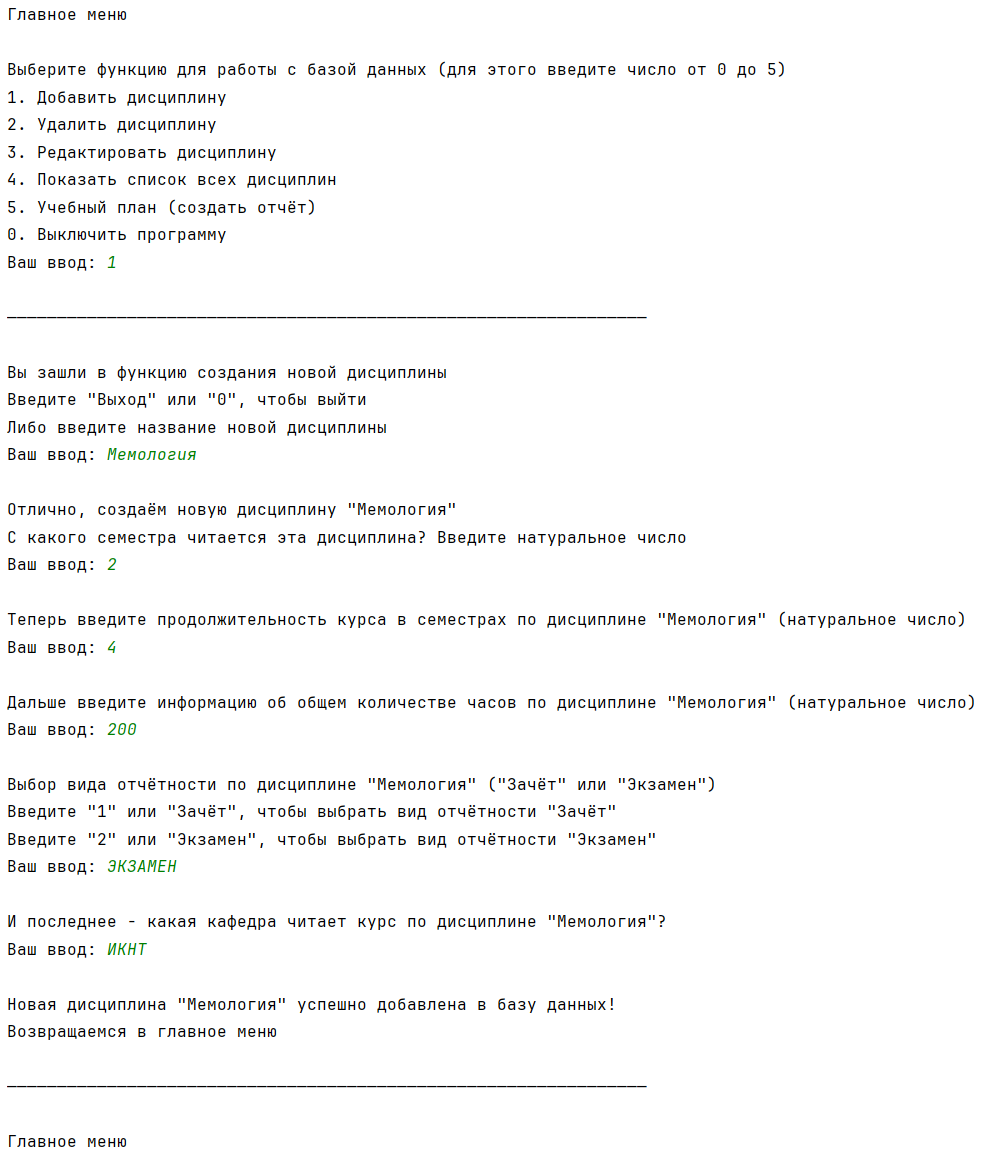
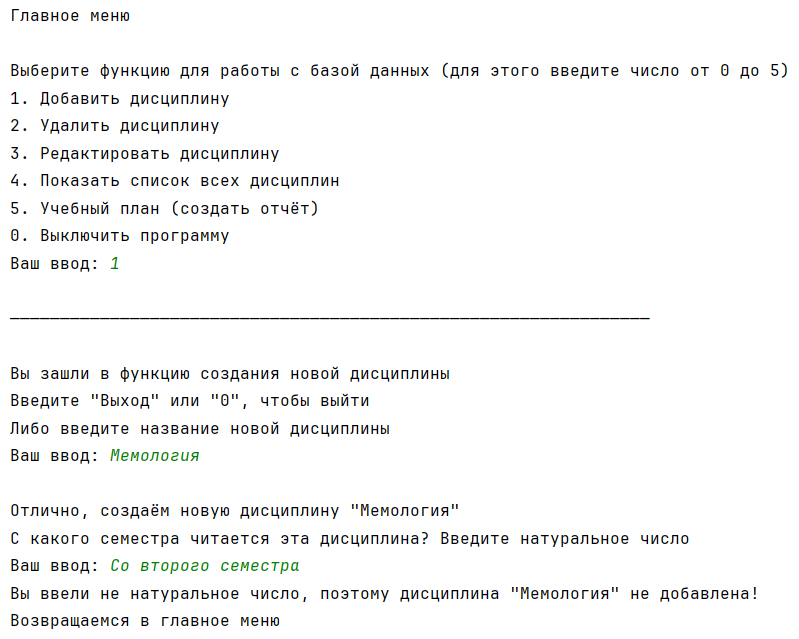
- На основе введённой пользователем информации проводит сортировку базы данных и выводит на экран запрашиваемую информацию

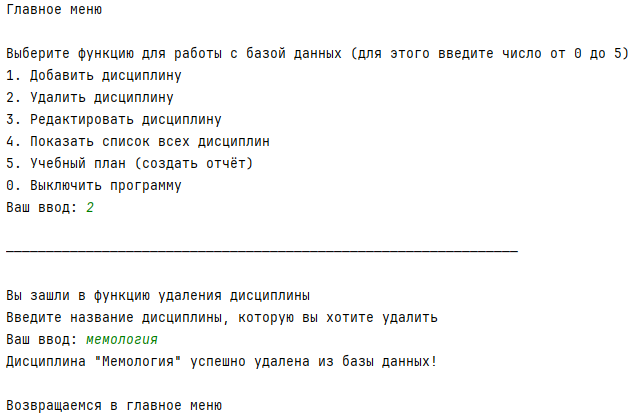
8. Обработка ошибок

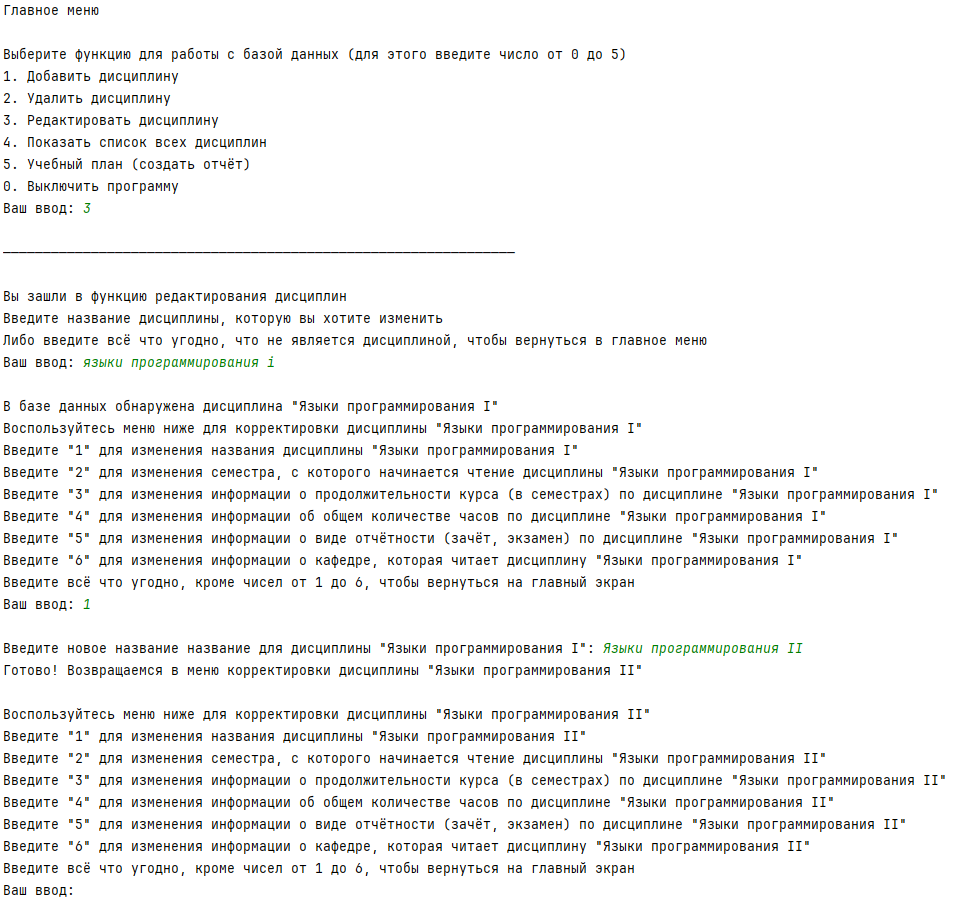
- Программа обрабатывает возможные ошибки при чтении и записи данных, а также при выполнении пользовательских команд

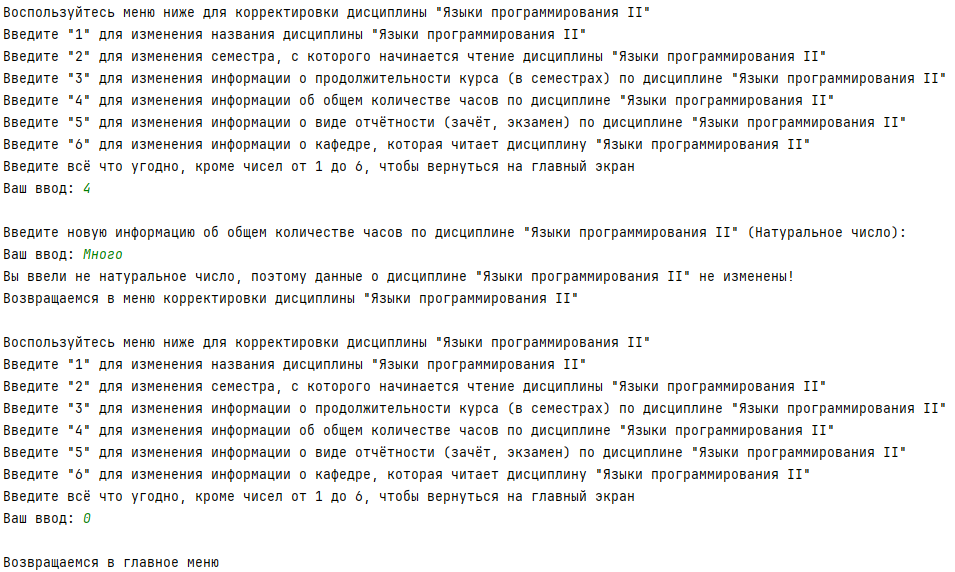
# Тестирование задачи

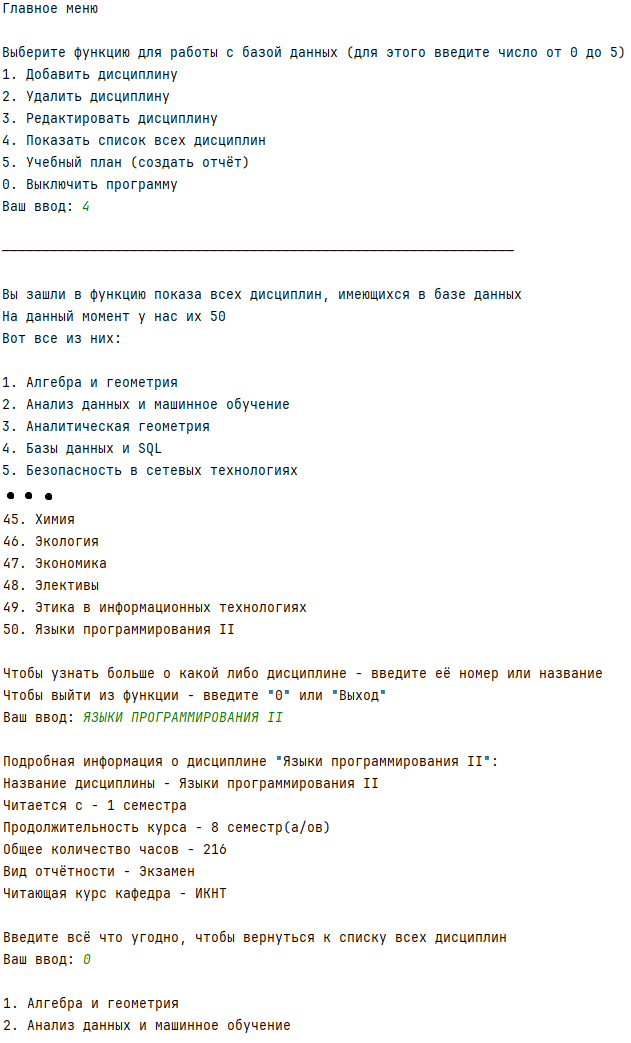


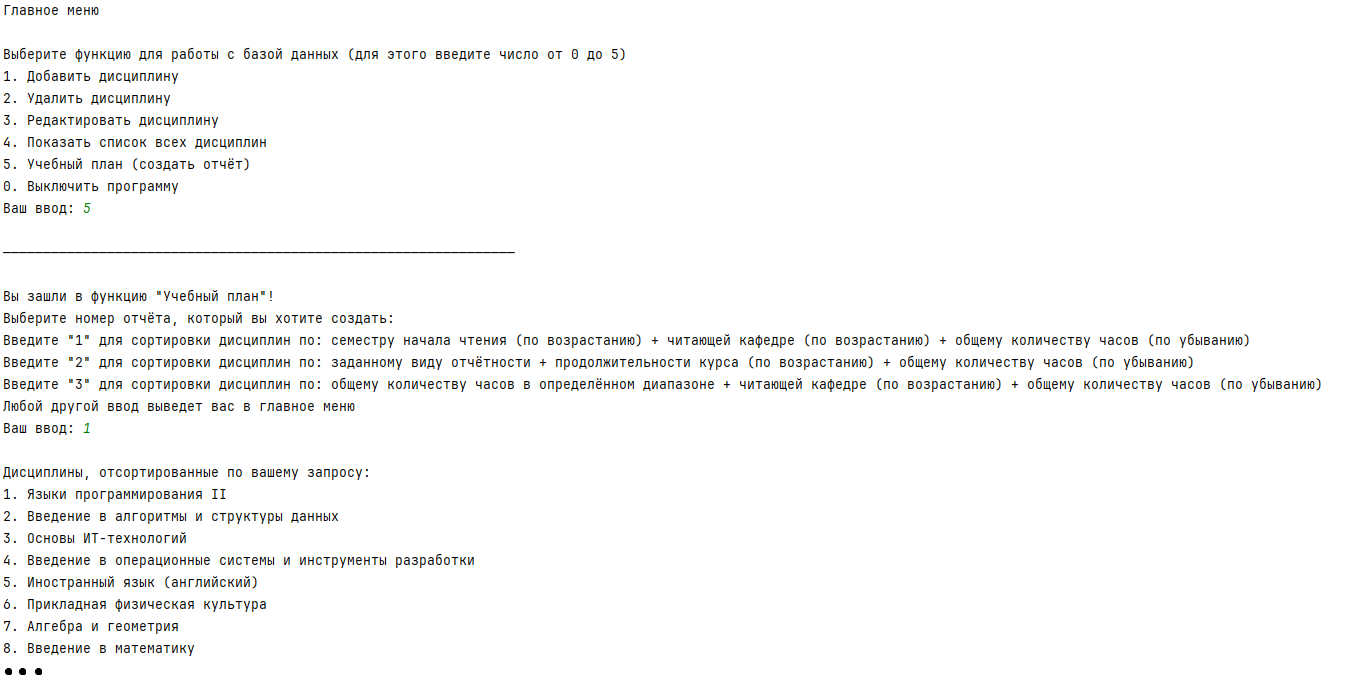














# Код программы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class Discipline: *# Создание класса Discipline и описание её свойств* def \_\_init\_\_(self, name, semester, duration, total\_hours, assessment, department):  
 self.name = name  
 self.semester = semester  
 self.duration = duration  
 self.total\_hours = total\_hours  
 self.assessment = assessment  
 self.department = department  
  
  
def read\_data\_from\_file(): *# Функция считывания данных с БД* result = [] *# Создать временный массив* with open(file, 'r') as f: *# Открыть БД, с которой работаем* arr = f.readlines() *# Каждую строку БД записываем как элемент во временный массив arr* for i in arr: *# Для каждой строки в массиве arr* data = i.strip().split(', ') *# В массив data записать элементы строки, разделённые ", "* name = data[0][1:-1] *# Первый элемент записать в name, но без его начального и конечного символов* semester = int(data[1]) *# Второй элемент записать как число* duration = int(data[2]) *# Третий элемент записать как число* total\_hours = int(data[3]) *# Четвёртый элемент записать как число* assessment = data[4][1:-1] *# Пятый элемент записать без начального и конечного символа (без кавычек)* department = data[5][1:-1] *# Шестой элемент аналогично* result.append(Discipline(name, semester, duration, total\_hours, assessment, department))  
 return result *# Все элементы записать в формате элемента класса Discipline в БД программы*def write\_data\_to\_file(data): *# Функция записи данных в БД* with open(file, 'w') as f: *# Открыть БД, с которой работаем* for discipline in data: *# Для каждого элемента БД* line = (f'"{discipline.name}", {discipline.semester}, {discipline.duration}, {discipline.total\_hours}, '  
 f'"{discipline.assessment}", "{discipline.department}"\n')  
 f.write(line) *# Записать в строчку по шаблону файла БД*def binary\_insertion\_sort(arr, lam=lambda x: x): *# Сортировка массива по какому-то свойству его элементов* for i in range(1, len(arr)): *# Сортировка бинарными вставками* key = arr[i] *# Запоминаем элемент с которым работаем сейчас* left, right = 0, i *# Границы отсортированной части массива* while left < right: *# Пока левый указатель и правый не на одном элементе* mid = (left + right) // 2 *# Создаём центральный указатель* if lam(arr[mid]) <= lam(key): *# Сравниваем с ключом элемент на центральном указателе по свойству* left = mid + 1 *# Если ключ больше - сдвигаем левый и центральный указатель вправо* else: *# Иначе (если ключ меньше центрального указателя), значит нужная ячейка левее* right = mid *# центрального указателя, значит правому указателю даём значение центрального* for j in range(i, left, -1): *# Все указатели смотрят на одну ячейку, всё что правее неё сдвигаем* arr[j] = arr[j - 1] *# на один элемент вправо, чтобы освободить место для ключа* arr[left] = key *# В освобождённую ячейку записываем значение ключа* return arr *# Вернуть отсортированный массив*def add\_discipline(database): *# Функция создания новой дисциплины* name = str(input('\n————————————————————————————————————————————————————————————————\n\n'  
 'Вы зашли в функцию создания новой дисциплины\n'  
 'Введите "Выход или "0", чтобы выйти\n'  
 'Либо введите название новой дисциплины\nВаш ввод: '))  
 if (name == '0') or (name.lower() == 'выход'): *# Если пользователь попросился на выход, вместо ввода названия* print(f'Возвращаемся в главное меню\n\n————————————————————————————————————————————————————————————————\n')  
 return  
 try: *# Вводим информацию (с какого семестра), должно быть число* semester = int(input(f'\nОтлично, создаём новую дисциплину "{name}"\n'  
 f'С какого семестра читается эта дисциплина? Введите натуральное число\nВаш ввод: '))  
 except: *# Если пользователь ввёл не число* print(f'Вы ввели не натуральное число, поэтому дисциплина "{name}" не добавлена!\nВозвращаемся в главное меню'  
 f'\n\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть его в главное меню* try: *# Информация с продолжительности курса (число)* duration = int(input(f'\nТеперь введите продолжительность курса в семестрах по дисциплине "{name}" '  
 f'(натуральное число)\nВаш ввод: '))  
 except: *# Если пользователь ввёл не число* print(f'Вы ввели не натуральное число, поэтому дисциплина "{name}" не добавлена!\nВозвращаемся в главное меню'  
 f'\n\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть его в главное меню* try: *# Информация об общем количестве часов (число)* hours = int(input(f'\nДальше введите информацию об общем количестве часов по дисциплине "{name}" '  
 f'(натуральное число)\nВаш ввод: '))  
 except: *# Если пользователь ввёл не число* print(f'Вы ввели не натуральное число, поэтому дисциплина "{name}" не добавлена!\nВозвращаемся в главное меню'  
 f'\n\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть его домой* assessment = str(input(f'\nВыбор вида отчётности по дисциплине "{name}" ("Зачёт" или "Экзамен")\n'  
 f'Введите "1" или "Зачёт", чтобы выбрать вид отчётности "Зачёт"\n'  
 f'Введите "2" или "Экзамен", чтобы выбрать вид отчётности "Экзамен"\nВаш ввод: '))  
 if (assessment == '1') or (assessment.lower() == 'зачёт'): *# Пользователь выбрал "зачёт" для новой дисциплины* if file == 'data.txt': *# Если работаем с английской БД* assessment = 'Credit'  
 else: *# Если работаем с русской БД* assessment = 'Зачёт'  
 elif (assessment == '2') or (assessment.lower() == 'экзамен'): *# Если пользователь выбрал "экзамен"* if file == 'data.txt': *# Если работаем с английской БД* assessment = 'Exam'  
 else: *# Если работаем с русской БД* assessment = 'Экзамен'  
 else: *# Если пользователь вводит ерунду* print(f'Вы ввели некорректную информацию, поэтому дисциплина "{name}" не добавлена!\nВозвращаемся в главное '  
 f'меню\n\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть его в главное меню, новую дисциплину не создавать* department = str(input(f'\nИ последнее - какая кафедра читает курс по дисциплине "{name}"?\nВаш ввод: '))  
 database.append(Discipline(name, semester, duration, hours, assessment, department)) *# Внести информацию* print(f'\nНовая дисциплина "{name}" успешно добавлена в базу данных!\n' *# Дисциплина создана* f'Возвращаемся в главное меню\n\n————————————————————————————————————————————————————————————————\n')  
  
  
def remove\_discipline(database): *# Функция удаления дисциплины* ln = len(database) *# Количество дисциплин на данный момент* if ln == 0: *# Если их ноль* print('\n————————————————————————————————————————————————————————————————\n\n'  
 'Вы зашли в функцию удаления дисциплины, но на данный момент в базе данных нет ни одной дисциплины\n'  
 'Возвращаемся в главное меню\n'  
 '\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть в главное меню* name = str(input('\n————————————————————————————————————————————————————————————————\n\n'  
 'Вы зашли в функцию удаления дисциплины\n' *# Ввести название дисциплины на удаление* 'Введите название дисциплины, которую вы хотите удалить\nВаш ввод: '))  
 chek = 0 *# Находили ли мы нужную дисциплину?* for i in range(ln): *# Пройтись по всем дисциплинам* if name.lower() == str(database[i].name).lower(): *# Если нашли нужную дисциплину* print(f'Дисциплина "{database[i].name}" успешно удалена из базы данных!')  
 database.pop(i) *# Удалить эту дисциплину из БД* chek += 1 *# Мы находили нужную дисциплину* break *# Выйти из цикла* if chek == 0: *# Если дисциплину не нашли* print('Данная дисциплина не была обнаружена в базе данных. Возможно вы ошиблись при вводе её названия')  
 print('\nВозвращаемся в главное меню\n\n————————————————————————————————————————————————————————————————\n')  
  
  
def edit\_discipline(database): *# Функция редактирования дисциплин* ln = len(database) *# Количество дисциплин в БД* if ln == 0: *# Если их ноль* print('\n————————————————————————————————————————————————————————————————\n\n'  
 'Вы зашли в функцию редактирования дисциплин, но на данный момент в базе данных нет ни одной дисциплины'  
 '\nВозвращаемся в главное меню\n'  
 '\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть пользователя в главное меню* name = str(input('\n————————————————————————————————————————————————————————————————\n\n'  
 'Вы зашли в функцию редактирования дисциплин\n'  
 'Введите название дисциплины, которую вы хотите изменить\n'  
 'Либо введите всё что угодно, что не является дисциплиной, чтобы вернуться в главное меню'  
 '\nВаш ввод: '))  
 for i in range(ln): *# Пройтись по всей БД в поиске нужного названия* if name.lower() == str(database[i].name).lower(): *# Если нужная дисциплина найдена* print(f'\nВ базе данных обнаружена дисциплина "{database[i].name}"')  
 while True: *# Меню редактирования найденной дисциплины (выйдем когда пользователь попросит)* a = str(input(f'Воспользуйтесь меню ниже для корректировки дисциплины "{database[i].name}"\n'  
 f'Введите "1" для изменения названия дисциплины "{database[i].name}"\n'  
 f'Введите "2" для изменения семестра, с которого начинается чтение '  
 f'дисциплины "{database[i].name}"\n'  
 f'Введите "3" для изменения информации о продолжительности курса (в семестрах) '  
 f'по дисциплине "{database[i].name}"\n'  
 f'Введите "4" для изменения информации об общем количестве часов '  
 f'по дисциплине "{database[i].name}"\n'  
 f'Введите "5" для изменения информации о виде отчётности (зачёт, экзамен) '  
 f'по дисциплине "{database[i].name}"\n'  
 f'Введите "6" для изменения информации о кафедре, которая читает '  
 f'дисциплину "{database[i].name}"\n'  
 f'Введите всё что угодно, кроме чисел от 1 до 6, чтобы вернуться на главный экран\n'  
 f'Ваш ввод: '))  
 if a == '1': *# Пользователь хочет сменить название дисциплины* database[i].name = str(input(f'\nВведите новое название название для '  
 f'дисциплины "{database[i].name}": '))  
 print(f'Готово! Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 elif a == '2': *# Пользователь хочет сменить данные о семестре начала чтения дисциплины* try: *# Пытаемся получить от него число* semester = int(input(f'\nВведите новую информацию о том, к какого семестра читается '  
 f'дисциплина "{database[i].name}" (Натуральное число):\nВаш ввод: '))  
 database[i].semester = semester *# Обновить данные в БД об этой дисциплине* print(f'Готово! Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 except: *# Пользователь ввёл не число, вернуть его в меню корректировки дисциплины* print(f'Вы ввели не натуральное число, поэтому данные о дисциплине '  
 f'"{database[i].name}" не изменены!\n'  
 f'Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 elif a == '3': *# Пользователь меняет информацию о количестве семестров* try: *# Пробуем получить от него число* duration = int(input(f'\nВведите новую информацию о количестве семестров по дисциплине '  
 f'"{database[i].name}" (Натуральное число):\nВаш ввод: '))  
 database[i].duration = duration *# Обновляем данные в БД* print(f'Готово! Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 except: *# Пользователь ввёл не число, возвращаем его в меню редактирования* print(f'Вы ввели не натуральное число, поэтому данные о дисциплине '  
 f'"{database[i].name}" не изменены!\n'  
 f'Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 elif a == '4': *# Новая информация об общем количестве часов по дисциплине* try: *# Пытаемся получить число* hours = int(input(f'\nВведите новую информацию об общем количестве часов по дисциплине '  
 f'"{database[i].name}" (Натуральное число):\nВаш ввод: '))  
 database[i].total\_hours = hours *# Обновить данные в БД* print(f'Готово! Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 except: *# Пользователь ввёл не число* print(f'Вы ввели не натуральное число, поэтому данные о дисциплине '  
 f'"{database[i].name}" не изменены!\n'  
 f'Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 elif a == '5': *# Пользователь меняет информацию об виде отчётности* assessment = str(input(f'\nВведите новую информацию о виде отчётности по дисциплине '  
 f'"{database[i].name}" ("Зачёт" или "Экзамен")\n'  
 f'Введите "1" или "Зачёт", чтобы выбрать вид отчётности "Зачёт"\n'  
 f'Введите "2" или "Экзамен", чтобы выбрать вид отчётности "Экзамен"\n'  
 f'Ваш ввод: '))  
 if (assessment == '1') or (assessment.lower() == 'зачёт'): *# Пользователь просит "зачёт"* if file == 'data.txt': *# Если работаем с англ БД* database[i].assessment = 'Credit'  
 else: *# Если работаем с ру БД* database[i].assessment = 'Зачёт'  
 print(f'Готово! Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 elif (assessment == '2') or (assessment.lower() == 'экзамен'): *# Пользователь просит "экзамен"* if file == 'data.txt': *# Если работаем с англ БД* database[i].assessment = 'Exam'  
 else: *# Если работаем с ру БД* database[i].assessment = 'Экзамен'  
 print(f'Готово! Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 else: *# Пользователь ввёл ерунду* print(f'Вы ввели некорректную информацию, поэтому данные о дисциплине '  
 f'"{database[i].name}" не изменены!\n'  
 f'Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 elif a == '6': *# Пользователь желает изменить кафедру* database[i].department = str(input(f'Введите новую информацию о кафедре, которая читает '  
 f'дисциплину "{database[i].name}"\nВаш ввод: '))  
 print(f'Готово! Возвращаемся в меню корректировки дисциплины "{database[i].name}"\n')  
 else: *# Пользователь просится в главное меню* print(f'\nВозвращаемся в главное меню'  
 f'\n\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть его в главное меню* print(f'Данная дисциплина не была обнаружена в базе данных\nВозвращаемся в главное меню'  
 f'\n\n————————————————————————————————————————————————————————————————\n')  
  
  
def display\_database(database): *# Функция показа всех дисциплин* ln = len(database) *# Количество дисциплин в БД* if ln == 0: *# Если их сейчас ноль* print('\n————————————————————————————————————————————————————————————————\n\n'  
 'Вы зашли в функцию показа всех дисциплин, имеющихся в базе данных, '  
 'но в данный момент в базе данных нет ни одной дисциплины\nПеревожу вас в главное меню\n'  
 '\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть в главное меню* print('\n————————————————————————————————————————————————————————————————\n\n'  
 'Вы зашли в функцию показа всех дисциплин, имеющихся в базе данных\n'  
 f'На данный момент у нас их {ln}\nВот все из них:\n')  
 while True:  
 for i in range(ln): *# Пройтись по всем дисциплинам* print(f'{i + 1}. {database[i].name}') *# Вывести их названия* a = str(input('\nЧтобы узнать больше о какой либо дисциплине - введите её номер или название'  
 '\nЧтобы выйти из функции - введите "0" или "Выход"\nВаш ввод: ')) *# Подробности о дисциплине* if (a == '0') or (a.lower() == 'выход'): *# Если пользователь просится в главное меню* print(f'\nВозвращаемся в главное меню\n'  
 f'\n————————————————————————————————————————————————————————————————\n')  
 return *# Отправить пользователя в главное меню* chek = 0 *# Находили дисциплину по требованию пользователя?* for i in range(ln): *# Пройтись по всей БД в поисках дисциплины по требованию пользователя* if (a.lower() == (str(database[i].name)).lower()) or (a == str(i + 1)): *# Если совпадает название или номер* print(f'\nПодробная информация о дисциплине "{database[i].name}":\n'  
 f'Название дисциплины - {database[i].name}\n'  
 f'Читается с - {database[i].semester} семестра\n'  
 f'Продолжительность курса - {database[i].duration} семестр(а/ов)\n'  
 f'Общее количество часов - {database[i].total\_hours}\n'  
 f'Вид отчётности - {database[i].assessment}\n'  
 f'Читающая курс кафедра - {database[i].department}\n')  
 chek += 1 *# Нужная дисциплина была найдена* if chek == 0: *# Если она не была найдена* print(f'\nТакая дисциплина не найдена, возможно вы ошиблись в её названии')  
 input('Введите всё что угодно, чтобы вернуться к списку всех дисциплин\nВаш ввод: ')  
 print()  
  
  
def sort(database): *# Функция учебный план* print('\n————————————————————————————————————————————————————————————————\n\nВы зашли в функцию "Учебный план"!')  
 while True:  
 try: *# Попытаться получить от пользователя число,* srt = int(input('Выберите номер отчёта, который вы хотите создать:\n'  
 'Введите "1" для сортировки дисциплин по: семестру начала чтения (по возрастанию) + '  
 'читающей кафедре (по возрастанию) + общему количеству часов (по убыванию)\n'  
 'Введите "2" для сортировки дисциплин по: заданному виду отчётности + '  
 'продолжительности курса (по возрастанию) + общему количеству часов (по убыванию)\n'  
 'Введите "3" для сортировки дисциплин по: общему количеству часов в определённом диапазоне '  
 '+ читающей кафедре (по возрастанию) + общему количеству часов (по убыванию)\n'  
 'Любой другой ввод выведет вас в главное меню\nВаш ввод: '))  
 if not (1 <= srt <= 3): *# Если число нас не устраивает - вернуть в главное меню* print(f'\nВозвращаемся в главное меню\n'  
 f'\n————————————————————————————————————————————————————————————————\n')  
 return  
 except: *# Если получили не число - вернуть в главное меню* print(f'\nВозвращаемся в главное меню\n'  
 f'\n————————————————————————————————————————————————————————————————\n')  
 return  
  
 if srt == 1: *# Сортировка номер 1* arr = (binary\_insertion\_sort *# Взять копию базы данных, отправить её в сортировку по уменьшению* (binary\_insertion\_sort *# общего количества чтения дисциплины, полученный после сортировки* ((binary\_insertion\_sort *# массив отправить в сортировку по читающей кафедре (по возрастанию),* (database.copy(), *# полученный массив отсортировать по семестрам начала чтения (возраст)* lambda x: -x.total\_hours)), lambda x: x.department), lambda x: x.semester))  
 if len(arr) > 0: *# Полученный массив arr уже отсортирован так как нам нужно. Если в нём больше нуля элем...* print(f'\nДисциплины, отсортированные по вашему запросу:')  
 else: *# Если в нём нет элементов, то пишем...* print('\nНа данный момент в базе нет ни одной дисциплины, удовлетворяющей вашим требованиям')  
 for i in range(len(arr)): *# Проходим по всем элементам* print(f'{i + 1}. {arr[i].name}') *# Пишем все элементы массива arr по порядку* srt = str(input('\nВведите "0" или "Выход", чтобы вернуться в главное меню'  
 '\nВведите всё что угодно другое, чтобы перейти к меню функции "Учебный план"'  
 '\nВаш ввод: '))  
 if srt == '0' or srt.lower() == 'выход': *# Если пользователь хочет в главное меню* print(f'\nВозвращаемся в главное меню\n'  
 f'\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть его в главное меню* print() *# Иначе - оставить в меню функции "Учебный план"* elif srt == 2: *# Сортировка номер 2* srt = input('\nВыберите вид отчётности, по которому нужно выдать выборку дисциплин\n'  
 'Введите "1" или "Зачёт", чтобы выбрать вид отчётности "Зачёт"\n'  
 'Введите "2" или "Экзамен", чтобы выбрать вид отчётности "Экзамен"\n'  
 'Ваш ввод: ') *# Выбор "Зачёт" или "Экзамен"* arr = [] *# Создать массив, в котором будут нужные нам элементы* if srt == '1' or srt.lower() == 'зачёт': *# Если пользователь выбрал "Зачёт"* for i in range(len(database)): *# Пройтись по всем элементам БД* if ((str(database[i].assessment).lower() == 'зачёт') or  
 (str(database[i].assessment).lower() == 'credit')):  
 arr.append(database[i]) *# Добавить в arr те, которые обладают свойством "Зачёт"* elif srt == '2' or srt.lower() == 'экзамен': *# Если пользователь выбрал "Экзамен"* for i in range(len(database)): *# Пройтись по БД* if ((str(database[i].assessment).lower() == 'экзамен') or  
 (str(database[i].assessment).lower() == 'exam')):  
 arr.append(database[i]) *# Добавить в arr элементы со свойством "Экзамен"* arr = (binary\_insertion\_sort *# Массив arr отсортировать по уменьшению общего количества часов,* ((binary\_insertion\_sort *# полученное отсортировать по кафедрам, полученное записать в arr* (arr, lambda x: -x.total\_hours)), lambda x: x.duration))  
 if len(arr) > 0: *# Если в arr есть элементы...* print(f'\nДисциплины, отсортированные по вашему запросу:')  
 else: *# Если в arr нет элементов...* print('\nНа данный момент в базе нет ни одной дисциплины, удовлетворяющей вашим требованиям')  
 for i in range(len(arr)): *# Пройтись по массиву arr и выписать все элементы* print(f'{i + 1}. {arr[i].name}')  
 srt = str(input('\nВведите "0" или "Выход", чтобы вернуться в главное меню'  
 '\nВведите всё что угодно другое, чтобы перейти к меню функции "Учебный план"'  
 '\nВаш ввод: '))  
 if srt == '0' or srt.lower() == 'выход': *# Если пользователь хочет в главное меню* print(f'\nВозвращаемся в главное меню\n'  
 f'\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть в главное меню* print()  
 elif srt == 3: *# Сортировка номер 2* try: *# Попытаться получить от пользователя два числа* n1, n2 = map(int, input('\nЧтобы подготовить выборку дисциплин, в которой общее количество часов '  
 'чтения находится в заданном диапазоне, введите два числа через пробел: '  
 'Минимальное и Максимальное количество часов чтения дисциплин\n'  
 'Чтобы выйти в главное меню - введите что угодно другое\n'  
 'Ваш ввод: ').split(" ", 1))  
 except: *# Если не получилось - вернуть его в главное меню* print(f'\nВозвращаемся в главное меню\n'  
 f'\n————————————————————————————————————————————————————————————————\n')  
 return *# Вернуть в главное меню* arr = [] *# Создать массив arr, который в последствии будем пополнять* for i in range(len(database)): *# Пройтись по всей БД* if n1 <= int(database[i].total\_hours) <= n2: *# Если общее количество часов элемента в диапазоне,* arr.append(database[i]) *# указанном пользователем, то добавить элемент в arr* arr = (binary\_insertion\_sort *# Сортируем массив arr по уменьшению* ((binary\_insertion\_sort *# общего количества часов, полученный массив* (arr, lambda x: -x.total\_hours *# сортируем по читающей дисциплину кафедре,* )), lambda x: x.department)) *# полученный массив записываем в переменную arr* if len(arr) > 0: *# Если в массиве есть дисциплины* print(f'\nДисциплины, отсортированные по вашему запросу:')  
 else: *# Если в массиве нет дисциплин* print('\nНа данный момент в базе нет ни одной дисциплины, удовлетворяющей вашим требованиям')  
 for i in range(len(arr)): *# Проходимся по всему массиву arr* print(f'{i + 1}. {arr[i].name}') *# Выписываем названия дисциплин* srt = str(input('\nВведите "0" или "Выход", чтобы вернуться в главное меню'  
 '\nВведите всё что угодно другое, чтобы перейти к меню функции "Учебный план"'  
 '\nВаш ввод: ')) *# Спрашиваем у пользователя, куда он хочет попасть дальше* if srt == '0' or srt.lower() == 'выход': *# Если он хочет в главное меню* print(f'\nВозвращаемся в главное меню\n'  
 f'\n————————————————————————————————————————————————————————————————\n')  
 return *# Возвращаем его в главное меню* print() *# Иначе возвращаемся к меню функции "Учебный план"*file = 'data.txt' *# Файл с которым будем работать*a = input('\nДобро пожаловать в программу для работы с базой данных по дисциплинам!\n'  
 'Надеюсь, опыт пользования программой вам понравится!\n\n'  
 'P.S. Программа плохо работает с данными на русском языке, поэтому сначала выберите базу данных\n'  
 'Введите "1", чтобы использовать базу данных на русском языке (на свой страх и риск)\n'  
 'Введите "2", чтобы продолжить использовать базу данных на английском языке\n'  
 'Ваш ввод: ')  
if a == '1': *# Пользователь будет работать с базой данных на русском* file = 'data\_ru.txt'  
error\_ru = 0 *# Для обработок ошибок в будущем*error\_en = 0  
ex = '' *# Переменная для текста ошибки*try:  
 A = read\_data\_from\_file() *# Пробуем скачать базу данных*except Exception as exception:  
 if file == 'data\_ru.txt': *# Если не удалось скачать базу данных на русском - пробуем скачать на английском* file = 'data.txt'  
 error\_ru = 1 *# Запоминаем что была ошибка с русской базой данных* try:  
 A = read\_data\_from\_file() *# Пробуем скачать базу данных на английском* except:  
 error\_en = 1 *# И русская и английская базы данных не загрузились* else:  
 error\_en = 1 *# Не загрузилась английская база данных, а русскую не пытались загрузить* ex = str(exception) *# В переменную ex записать текст ошибки*if a == '1': *# Пользователь просил базу данных на русском* if error\_ru == 0: *# Её удалось скачать* print('\nЗаписал, приступаем к работе с базой данных на русском и переходим в главное меню программы!\n'  
 '\n————————————————————————————————————————————————————————————————\n')  
 else: *# Её не удалось скачать, поэтому скачивали английскую* if error\_en == 0: *# Удалось скачать БД на английском* print(f'\n{ex}\n\nВо время загрузки базы данных на русском произошла ошибка, '  
 'поэтому была загружена база данных на английском!\n'  
 'Приступаем к работе с базой данных на английском и переходим в главное меню программы!\n'  
 '\n————————————————————————————————————————————————————————————————\n')  
 else: *# Не удалось скачать ни БД на русском, ни на английском* print(f'\n{ex}\n\nВо время загрузки базы данных на русском произошла ошибка!\n'  
 f'Поэтому я попытался загрузить базу данных на английском, но и с её загрузкой проблема повторилась\n'  
 f'Программа не смогла считать ни одну базу данных, поэтому завершает свою работу')  
elif a == '2': *# Пользователь просил БД на английском* if error\_en == 0: *# Ошибок не произошло* print('\nОтлично, приступаем к работе с базой данных на английском и переходим в главное меню программы!\n'  
 '\n————————————————————————————————————————————————————————————————\n')  
 else: *# Скачать БД на английском не удалось* print(f'\n{ex}\n\nВо время загрузки базы данных на английском произошла ошибка!\n'  
 f'Программа не смогла считать базу данных, поэтому завершает свою работу')  
else: *# Пользователь в начале программы повёл себя неожиданно* if error\_en == 0: *# Скачалась БД на английском* print('\nЯсно... Тогда я сделаю выбор за вас\nБудем работать с базой данных на английском\n'  
 'Переходим в главное меню программы!\n'  
 '\n————————————————————————————————————————————————————————————————\n')  
 else: *# Произошла ошибка при установке БД на английском* print(f'\nЯсно... Тогда я сделаю выбор за вас\nСкачиваю базу данных на английском...\n\n{ex}\n\n'  
 f'Ой, программа не смогла считать базу данных на английском, поэтому завершает свою работу')  
  
if error\_en == 0: *# Если при установке БД из файла data.txt ошибок не было,* binary\_insertion\_sort(A, lambda x: x.name) *# отсортировать БД по названию дисциплин*while error\_en == 0: *# Если ошибок при установке БД не было* try: *# Тело программы и главное меню запускается* a = str(input('Главное меню\n\n'  
 'Выберите функцию для работы с базой данных (для этого введите число от 0 до 5)\n'  
 '1. Добавить дисциплину\n'  
 '2. Удалить дисциплину\n'  
 '3. Редактировать дисциплину\n'  
 '4. Показать список всех дисциплин\n'  
 '5. Учебный план (создать отчёт)\n'  
 '0. Выключить программу\n'  
 'Ваш ввод: '))  
 if a == '1': *# Если пользователь хочет добавить дисциплину* add\_discipline(A) *# Функция добавления новой дисциплины* binary\_insertion\_sort(A, lambda x: x.name) *# Отсортировать дисциплины по названию* write\_data\_to\_file(A) *# Внести данные в БД* elif a == '2': *# Если пользователь хочет удалить дисциплину* remove\_discipline(A) *# Функция удаления* write\_data\_to\_file(A) *# Записать новые данные в БД* elif a == '3': *# Если пользователь редактирует дисциплины* edit\_discipline(A) *# Функция редактирования* binary\_insertion\_sort(A, lambda x: x.name) *# Сортировка по названию* write\_data\_to\_file(A) *# Внести в БД новую информацию* elif a == '4': *# Пользователь просит показать дисциплины* display\_database(A) *# Функция показа дисциплин* elif a == '5': *# Пользователь просит "Учебный план"* sort(A) *# Функция для учебного плана* elif a == '0': *# Пользователь выключает программу* print('\nПрограмма корректно завершила свою работу!')  
 error\_en = 1 *# Чтобы выйти из цикла while* else: *# Пользователь вводит ерунду* print('\nВы ввели не число от 0 до 5. Попробуйте ещё раз, у вас обязательно получится!\n'  
 '\n————————————————————————————————————————————————————————————————\n')  
 except Exception as ex: *# По ходу работы всей программы произошла неожиданная ошибка* print(f'\nК сожалению, во время работы программы произошла ошибка...\n{ex}\n'  
 f'\n————————————————————————————————————————————————————————————————\n')