Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторной работе №4**

Дисциплина: «Разработка профессиональных приложений»

Классы

Вариант № 5

Выполнил:

студент группы ИВТАПбд-21

Гаязов Т. Р.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту**

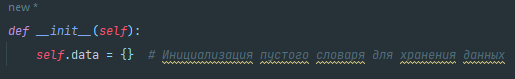
Необходимо переделать лабораторную работу №3 с использованием классов, описывающих предметную область, заданную вариантом, с реализацией следующих особенностей (вполне возможно, что предлагаемое в 3 лабе задание для этого нужно будет расширить):

**Требования к программе:**

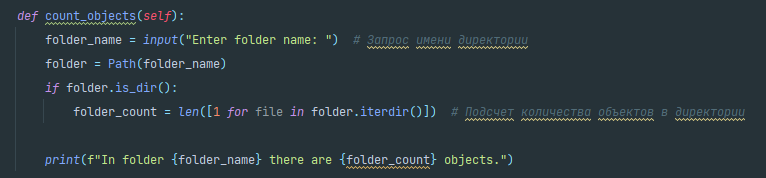
1. Класс должен содержать итератор
2. Должна быть реализована перегрузка стандартных операций (repr, например)
3. Должно быть реализовано наследование
4. Запись значений в свойства - только через \_\_setattr\_\_
5. Возможность доступа к элементам коллекции по индексу (\_\_getitem\_\_)
6. Должны быть реализованы статические методы
7. Должны быть реализованы генераторы

**Описание реализации**

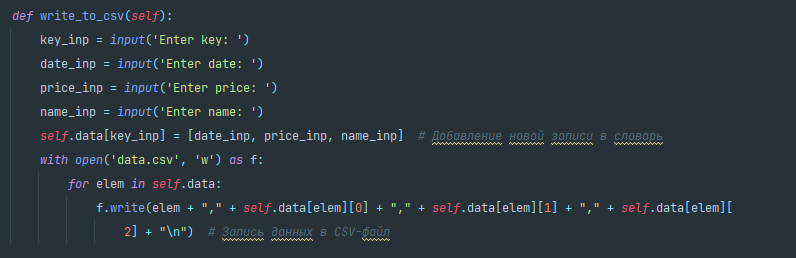
\_\_init\_\_: Инициализирует объект класса DataCollection и создает пустой словарь data для хранения данных.



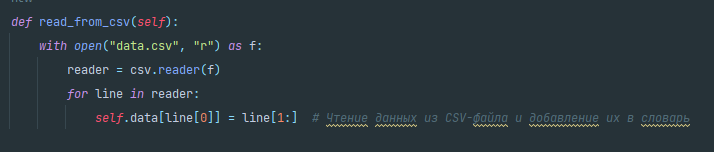
count\_objects: Запрашивает у пользователя имя директории, подсчитывает количество объектов в указанной директории и выводит результат.



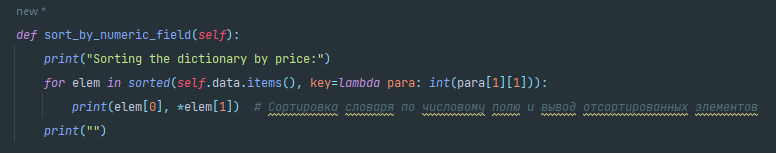
write\_to\_csv: Запрашивает у пользователя ключ, дату, цену и название товара, затем добавляет новую запись в словарь data. Затем записывает содержимое словаря в CSV-файл.



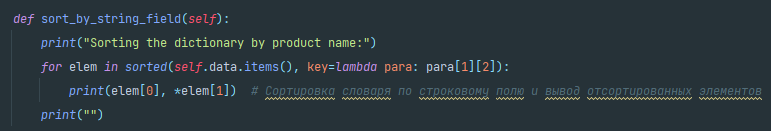
read\_from\_csv: Считывает данные из CSV-файла и добавляет их в словарь data.



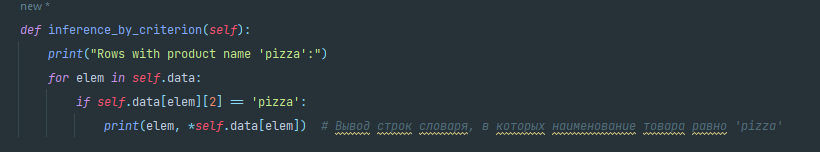
sort\_by\_numeric\_field: Сортирует словарь data по числовому полю "цена" и выводит отсортированные записи.



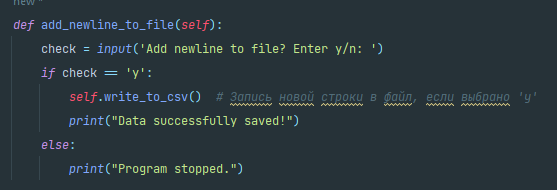
sort\_by\_string\_field: Сортирует словарь data по строковому полю "название товара" и выводит отсортированные записи.



inference\_by\_criterion: Выводит записи из словаря data, в которых название товара равно "pizza".



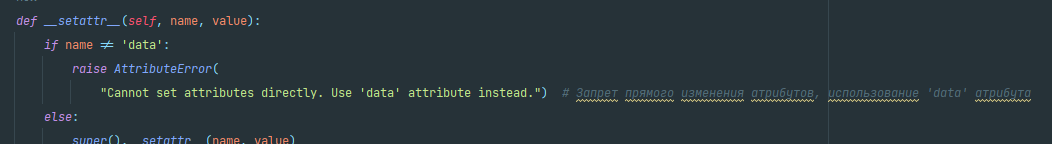
add\_newline\_to\_file: Запрашивает у пользователя ответ на вопрос о добавлении новой строки в файл. Если ответ "y", вызывает метод write\_to\_csv для добавления новой строки в файл.



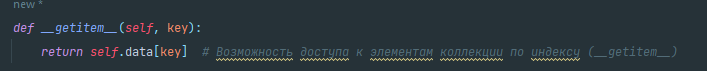
\_\_repr\_\_: Возвращает строковое представление словаря data.



\_\_setattr\_\_: Переопределяет метод \_\_setattr\_\_ для запрета прямого изменения атрибутов класса, кроме атрибута data.



\_\_getitem\_\_: Позволяет получать доступ к элементам коллекции по индексу, используя квадратные скобки.



Функция main: Создает экземпляр класса DataCollection, вызывает различные методы объекта collection для выполнения задач: подсчета объектов, чтения данных из файла, вывода данных, сортировки и фильтрации, а также добавления новой записи в файл.



Класс DataCollection представляет коллекцию данных, содержащую методы для работы с данными, а также включает в себя реализацию перегрузки стандартных операций (\_\_repr\_\_, \_\_setattr\_\_, \_\_getitem\_\_) и итератора (\_\_iter\_\_), а также статический метод и классовый метод. Он обеспечивает возможность работы с данными, их чтения и записи, сортировки, фильтрации и других операций, связанных с коллекцией данных.



Функция main является точкой входа в программу. Она создает объект DataCollection, выполняет различные операции с данными, в соответствии с логикой каждого метода, и выводит результаты на экран.

**Описание возникших затруднений**

Затруднений не возникло

**Результаты работы программы**



Рис. 1 «Поиск количества файлов в директории»

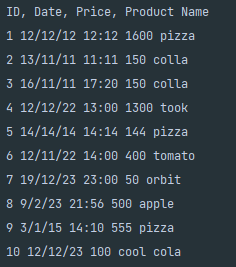


Рис. 2 «Вывод csv файла»

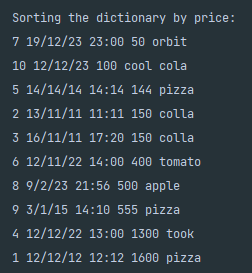


Рис. 3 «Вывод отсортированного csv файла по числовому полю»

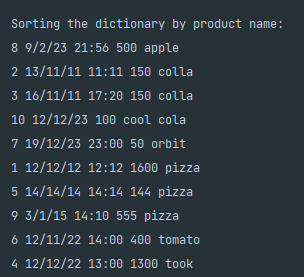


Рис. 4 «Вывод отсортированного csv файла по строковому полю»

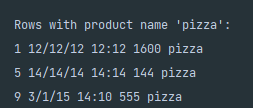


Рис. 5 «Вывод отсортированного csv файла по названию товара «Pizza» »

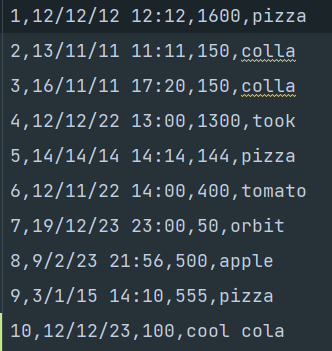


Рис. 6 «csv файл»

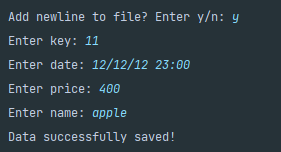


Рис. 7 «Добавление новой строки в csv файл»

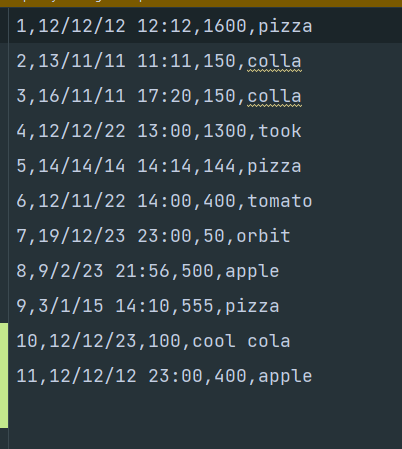


Рис. 8 «csv файл с добавленной строкой»

**Код программы**

*from* pathlib *import* Path  
*import* csv  
  
  
*class* DataCollection:  
 *def \_\_init\_\_*(*self*):  
 *self*.data = {} *# Инициализация пустого словаря для хранения данных  
  
 def* count\_objects(*self*):  
 folder\_name = *input*("Enter folder name: ") *# Запрос имени директории* folder = Path(folder\_name)  
 *if* folder.is\_dir():  
 folder\_count = *len*([1 *for* file *in* folder.iterdir()]) *# Подсчет количества объектов в директории  
  
 print*(f"In folder {folder\_name} there are {folder\_count} objects.")  
  
 *def* write\_to\_csv(*self*):  
 key\_inp = *input*('Enter key: ')  
 date\_inp = *input*('Enter date: ')  
 price\_inp = *input*('Enter price: ')  
 name\_inp = *input*('Enter name: ')  
 *self*.data[key\_inp] = [date\_inp, price\_inp, name\_inp] *# Добавление новой записи в словарь  
 with open*('data.csv', 'w') *as* f:  
 *for* elem *in self*.data:  
 f.write(elem + "," + *self*.data[elem][0] + "," + *self*.data[elem][1] + "," + *self*.data[elem][  
 2] + "\n") *# Запись данных в CSV-файл  
  
 def* read\_from\_csv(*self*):  
 *with open*("data.csv", "r") *as* f:  
 reader = csv.reader(f)  
 *for* line *in* reader:  
 *self*.data[line[0]] = line[1:] *# Чтение данных из CSV-файла и добавление их в словарь  
  
 def* sort\_by\_numeric\_field(*self*):  
 *print*("Sorting the dictionary by price:")  
 *for* elem *in sorted*(*self*.data.items(), key=*lambda* para: *int*(para[1][1])):  
 *print*(elem[0], \*elem[1]) *# Сортировка словаря по числовому полю и вывод отсортированных элементов  
 print*("")  
  
 *def* sort\_by\_string\_field(*self*):  
 *print*("Sorting the dictionary by product name:")  
 *for* elem *in sorted*(*self*.data.items(), key=*lambda* para: para[1][2]):  
 *print*(elem[0], \*elem[1]) *# Сортировка словаря по строковому полю и вывод отсортированных элементов  
 print*("")  
  
 *def* inference\_by\_criterion(*self*):  
 *print*("Rows with product name 'pizza':")  
 *for* elem *in self*.data:  
 *if self*.data[elem][2] == 'pizza':  
 *print*(elem, \**self*.data[elem]) *# Вывод строк словаря, в которых наименование товара равно 'pizza'  
  
 def* add\_newline\_to\_file(*self*):  
 check = *input*('Add newline to file? Enter y/n: ')  
 *if* check == 'y':  
 *self*.write\_to\_csv() *# Запись новой строки в файл, если выбрано 'y'  
 print*("Data successfully saved!")  
 *else*:  
 *print*("Program stopped.")  
  
 *def \_\_repr\_\_*(*self*):  
 *return repr*(*self*.data) *# Представление объекта в виде строки (repr)  
  
 def \_\_setattr\_\_*(*self*, name, value):  
 *if* name != 'data':  
 *raise AttributeError*(  
 "Cannot set attributes directly. Use 'data' attribute instead.") *# Запрет прямого изменения атрибутов, использование 'data' атрибута  
 else*:  
 *super*().*\_\_setattr\_\_*(name, value)  
  
 *def \_\_getitem\_\_*(*self*, key):  
 *return self*.data[key] *# Возможность доступа к элементам коллекции по индексу (\_\_getitem\_\_)* @staticmethod  
 *def* static\_method():  
 *# Implement static method logic here  
 pass* @classmethod  
 *def* class\_method(*cls*):  
 *# Implement class method logic here  
 pass  
  
 def \_\_iter\_\_*(*self*):  
 *return iter*(*self*.data) *# Реализация итератора (\_\_iter\_\_)  
  
  
def* main():  
 collection = DataCollection()  
 collection.count\_objects()  
 collection.read\_from\_csv()  
  
 *print*("ID, Date, Price, Product Name")  
 *for* elem *in* collection:  
 *print*(elem, \*collection[elem]) *# Вывод данных из коллекции  
 print*("\n")  
  
 collection.sort\_by\_numeric\_field()  
 collection.sort\_by\_string\_field()  
 collection.inference\_by\_criterion()  
 *print*("")  
  
 collection.add\_newline\_to\_file()  
  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 main()