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 № 1

Выполнил:

студент группы ИВТАПбд-21

Аронов В. В.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту**

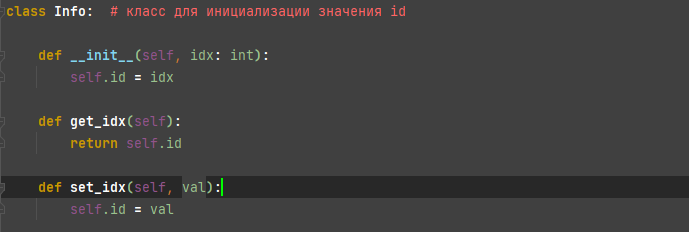
Необходимо переделать лабораторную работу №3 с использованием классов, описывающих предметную область, заданную вариантом, с реализацией следующих особенностей (вполне возможно, что предлагаемое в 3 лабе задание для этого нужно будет расширить):

**Требования к программе:**

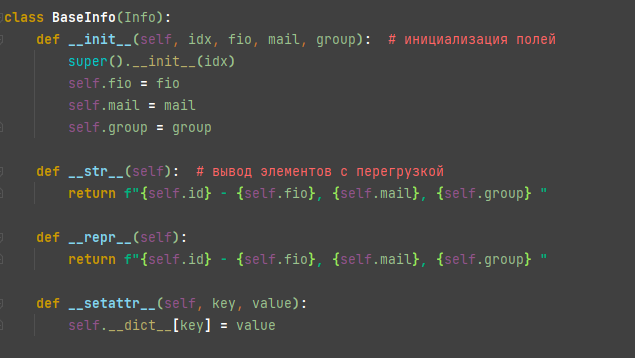
1. Класс должен содержать итератор
2. Должна быть реализована перегрузка стандартных операций (repr, например)
3. Должно быть реализовано наследование
4. Запись значений в свойства - только через \_\_setattr\_\_
5. Возможность доступа к элементам коллекции по индексу (\_\_getitem\_\_)
6. Должны быть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

**Описание реализации**

Класс Info для инициализации поля id

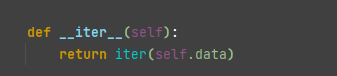


Класс BaseInfo с наследованием от класса Info, перегрузка методов \_\_repr\_\_ и \_\_str\_\_ для вывода новых (добавленных данных на экран):

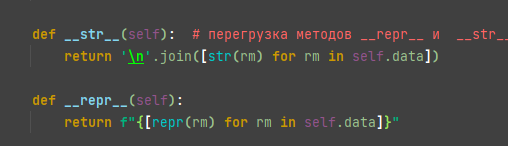


Инициализация наших данных, а именно конструктор, в который вписываем путь до нашего файла при создании экземпляра класса

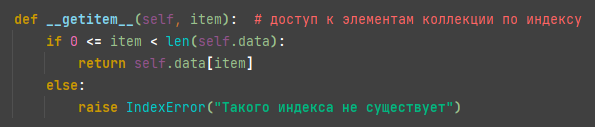
Итератор для вывода всего словаря:



Перегрузка методов:



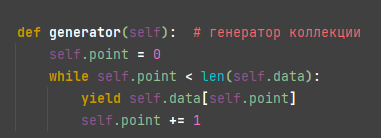
\_\_getitem\_\_ для получения определенной строки из словаря по индексу в словаре:



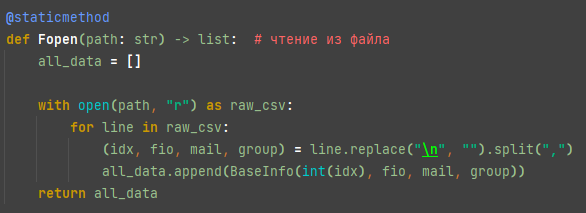
\_\_setattr\_ для записи значений в словарь:



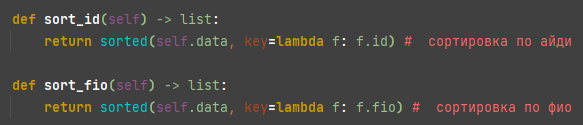
Генератор коллекции:



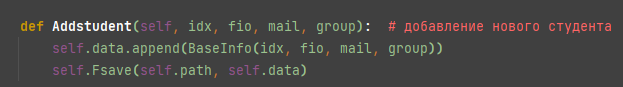
Чтение из файла:



Сортировка словаря построчно (ФИО) и численно(айди):



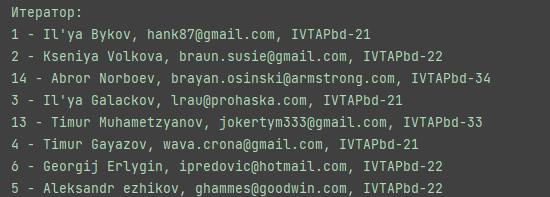
Добавление новых данных в словарь:



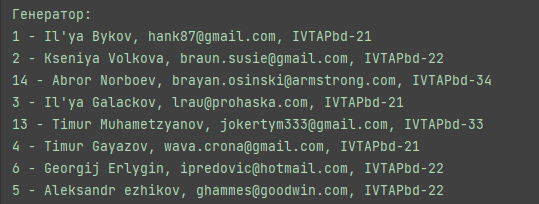
**Описание возникших затруднений**

Работа с ООП, классами, методами, перегрузками методов, а также с наследованием была применена впервые на языке Python. ООП в этом языке достаточно сильно отличается от ООП других языков, на работу ушло много времени, тем не менее с выполнением поставленной задачи справился.

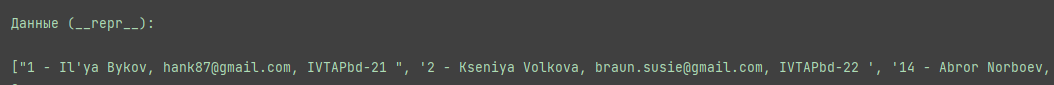
**Результаты работы программы**

****

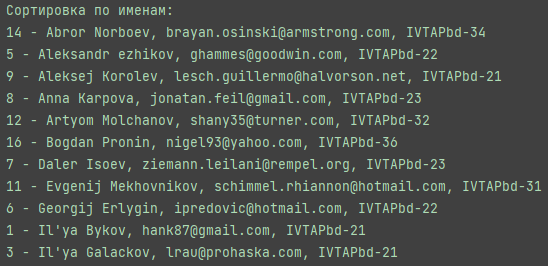
**1 рис. Вывод данных итератором**

****

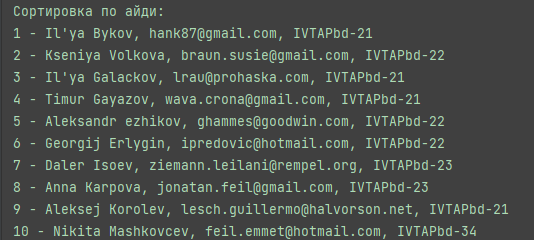
**2 рис. Вывод данных генератором**

****

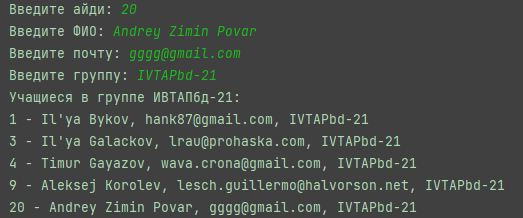
**3 рис. Вывод данных через перегрузку (\_\_repr\_\_)**

****

**4 рис. Сортировка по ФИО**

****

**5 рис. Сортировка по айди**

****

**6 рис. Добавление данных и вывод учащихся в группе ИВТАпбд-21**

****

**7 рис. Получение доступа к элементу коллекции через \_\_getitem\_\_**

**Код программы**

**class Info:** # класс для инициализации значения id  
  
 **def \_\_init\_\_(**self, idx**:** int**):** self.id **=** idx  
  
 **def get\_idx(**self**):  
 return** self.id  
  
 **def set\_idx(**self, val**):** self.id **=** val  
  
  
**class BaseInfo(**Info**):  
 def \_\_init\_\_(**self, idx, fio, mail, group**):** # инициализация полей  
 super**()**.\_\_init\_\_**(**idx**)** self.fio **=** fio  
 self.mail **=** mail  
 self.group **=** group  
  
 **def \_\_str\_\_(**self**):** # вывод элементов с перегрузкой  
 **return f"{**self.id**} - {**self.fio**}, {**self.mail**}, {**self.group**} "  
  
 def \_\_repr\_\_(**self**):  
 return f"{**self.id**} - {**self.fio**}, {**self.mail**}, {**self.group**} "  
  
 def \_\_setattr\_\_(**self, key, value**):** self.\_\_dict\_\_**[**key**] =** value  
  
  
**class File:  
  
 def \_\_init\_\_(**self, path**):** self.point **=** 0  
 self.path **=** path  
 self.data **=** self.Fopen**(**self.path**)  
  
 def \_\_str\_\_(**self**):** # перегрузка методов \_\_repr\_\_ и \_\_str\_\_ для вывода данных из файла  
 **return '\n'**.join**([**str**(**i**) for** i **in** self.data**])  
  
 def \_\_repr\_\_(**self**):  
 return f"{[**repr**(**i**) for** i **in** self.data**]}"  
  
 def \_\_iter\_\_(**self**):  
 return** iter**(**self.data**)  
  
 def \_\_next\_\_(**self**):  
 if** self.point **>=** len**(**self.data**):** self.point **=** 0  
 **raise** StopIteration  
 **else:** self.point **+=** 1  
 **return** self.data**[**self.point **-** 1**]  
  
 def generator(**self**):** # генератор коллекции  
 self.point **=** 0  
 **while** self.point **<** len**(**self.data**):  
 yield** self.data**[**self.point**]** self.point **+=** 1  
  
 **def sort\_id(**self**)** -> list**:  
 return** sorted**(**self.data, key**=lambda** f**:** f.id**)** # сортировка по айди  
  
 **def sort\_fio(**self**)** -> list**:  
 return** sorted**(**self.data, key**=lambda** f**:** f.fio**)** # сортировка по фио  
  
 **def \_\_getitem\_\_(**self, item**):** # доступ к элементам коллекции по индексу  
 **if** 0 **<=** item **<** len**(**self.data**):  
 return** self.data**[**item**]  
 else:  
 raise** IndexError**("Такого индекса не существует")  
  
 @**staticmethod  
 **def Fopen(**path**:** str**)** -> list**:** # чтение из файла  
 all\_data **= []  
  
 with** open**(**path, **"r") as** raw\_csv**:  
 for** line **in** raw\_csv**:  
 (**idx, fio, mail, group**) =** line.replace**("\n"**, **"")**.split**(",")** all\_data.append**(**BaseInfo**(**int**(**idx**)**, fio, mail, group**))  
 return** all\_data  
  
 **def Addstudent(**self, idx, fio, mail, group**):** # добавление нового студента  
 self.data.append**(**BaseInfo**(**idx, fio, mail, group**))** self.Fsave**(**self.path, self.data**)  
  
 def find(**self**)** -> list**:** # поиск учащихся в группе ИВТАПбд-21  
 **return [**i **for** i **in** self.data **if** i.group **== "IVTAPbd-21"]  
  
 @**staticmethod  
 **def Fsave(**f, new\_data**):** # сохранение в файл  
 **with** open**(**f, **"w") as** f**:  
 for** element **in** new\_data**:** f.write**(f"{**element.id**},{**element.fio**},{**element.mail**},{**element.group**}\n")**d **=** File**('data-1.csv')  
  
  
def Mainsub():** print**("Итератор:")  
 for** i **in** iter**(**d**):** print**(**i**)** print**("Генератор:")  
 for** i **in** d.generator**():** print**(**i**)** print**("\nДанные (\_\_repr\_\_):\n"**, repr**(**d**)**, sep**='\n')** print**("Сортировка по именам: ")  
 for** i **in** d.sort\_fio**():** print**(**i**)** print**("Сортировка по айди: ")  
 for** i **in** d.sort\_id**():** print**(**i**)** d.Addstudent**(**input**('Введите айди: ')**, input**('Введите ФИО: ')**,  
 input**('Введите почту: ')**, input**('Введите группу: '))** print**("Учащиеся в группе ИВТАПбд-21: ")  
 for** i **in** d.find**():** print**(**i**)** a **=** int**(**input**("Введите индекс:"))** print**("Строчка под индексом"**, a, **"->"**, d.\_\_getitem\_\_**(**a**))**Mainsub**()**