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

Выполнила:

студентка группы ИВТАПбд-22

Архипова А.С.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

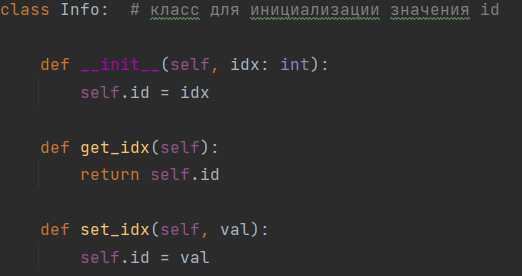
**Задание по варианту**

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):

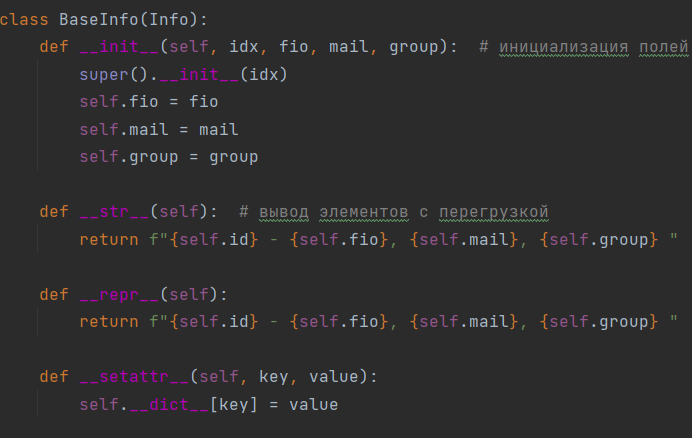
**Требования к программе:**

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**

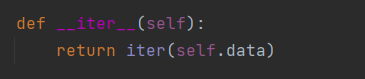
Класс Info для инициализации поля id   


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

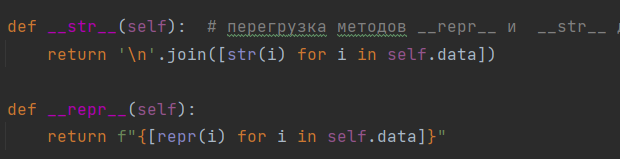


Инициализация наших данных, а именно конструктор, в который вписываем путь до нашего файла при создании экземпляра класса

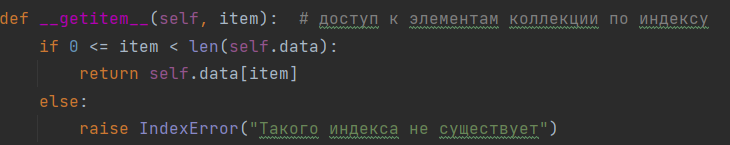
Итератор для вывода всего словаря:



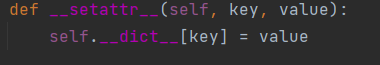
Перегрузка методов:



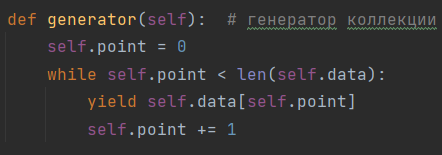
\_\_getitem\_\_ для получения определенной строки из словаря по индексу в словаре:



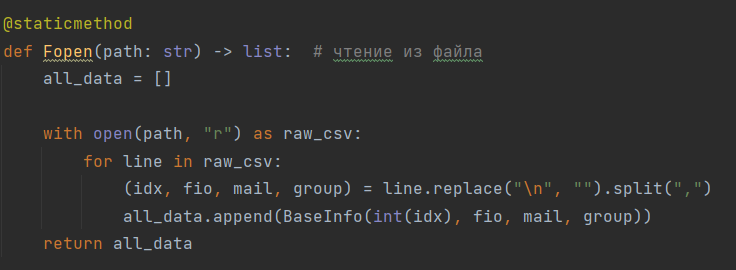
\_\_setattr\_ для записи значений в словарь:



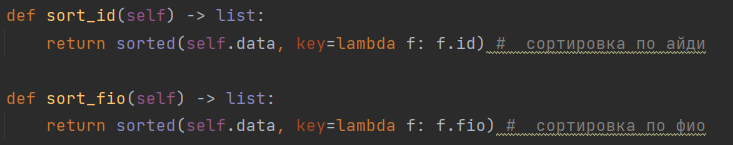
Генератор коллекции:



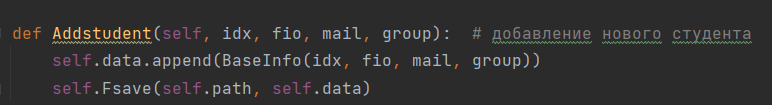
Чтение из файла:



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



Добавление новых данных в словарь:

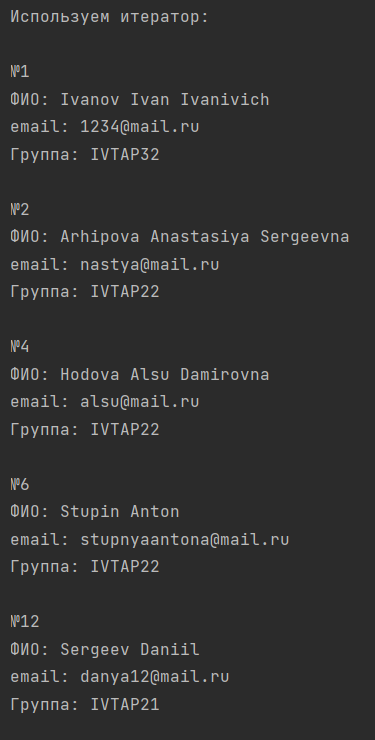


**Описание возникших затруднений**

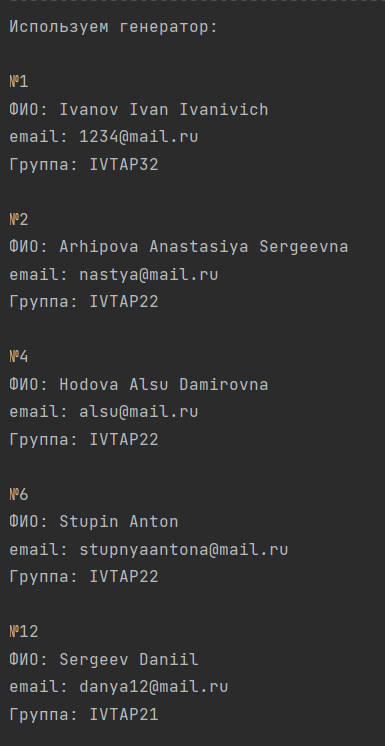
Работа с ООП, классами, методами, перегрузками методов, а также с наследованием была применена впервые на языке Python. ООП в этом языке достаточно сильно отличается от ООП других языков, на работу ушло много времени, тем не менее с выполнением поставленной задачи справился.

**Результаты работы программы**

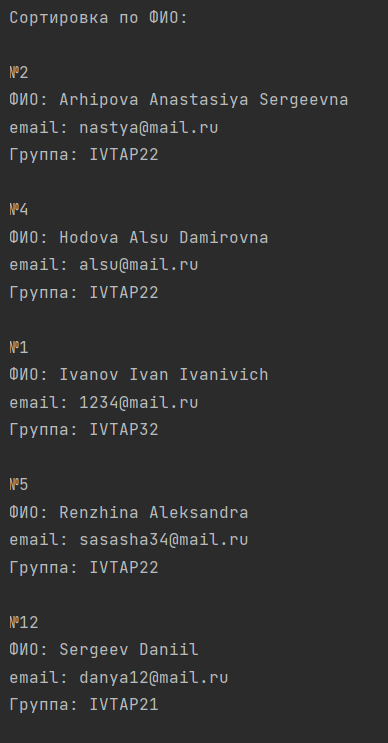
Тест 1. Вывод данных итератором

****

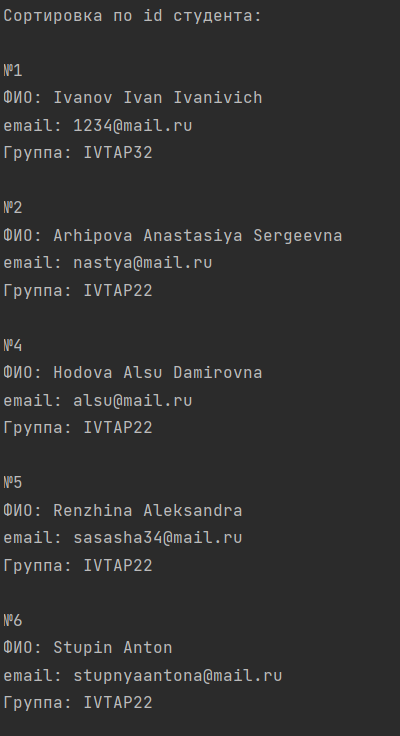
Тест 2. Вывод данных генератором

****

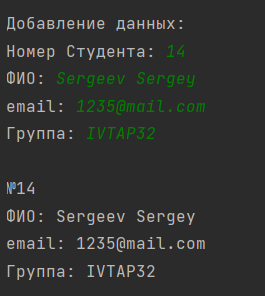
Тест 3. Сортировка по ФИО

****

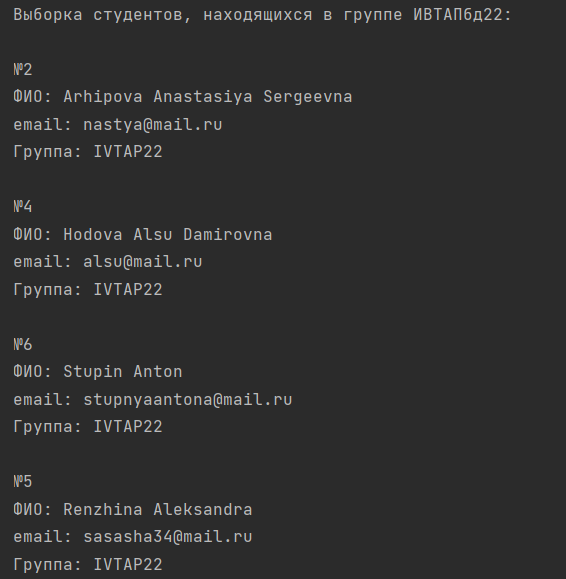
Тест 4. Сортировка по id



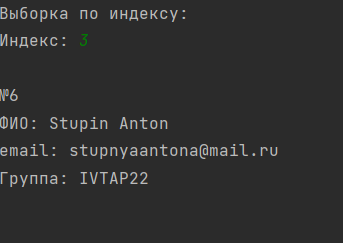
Тест 5. Добавление данных и вывод

****

Тест 6. Вывод студентов группы ИВТАП22

****

Тест 7. Выборка по индексу

****

**Код программы**

ALPHABET = "qwertyuiopasdfghjklzxcvbnm1234567890\_"  
  
  
# класс хранит номер студента отдельно  
class Row():  
 id = 0  
  
 def \_\_init\_\_(self, id: int):  
 self.id = id  
  
 def get\_id(self):  
 return self.id  
  
 def set\_id(self, val):  
 self.id = val  
  
  
# класс хранит все данные списка  
class RowModel(Row):  
 id = 0  
 name = ""  
 email = ""  
 group = ""  
  
 def \_\_init\_\_(self, id: int, name: str, email: str, group: str):  
 super().\_\_init\_\_(id)  
 self.name = name  
 self.email = email  
 self.group = group  
  
 # вызывается при команде print элемента списка  
 def \_\_str\_\_(self):  
 return f"№{self.id}\nФИО: {self.name}\nemail: {self.email}\nГруппа: {self.group}\n"  
  
 def \_\_repr\_\_(self):  
 return f"RowModel(id={self.id},name={self.name},email={self.email},group={self.group})"  
  
 def \_\_setattr\_\_(self, \_\_name, \_\_value):  
 self.\_\_dict\_\_[\_\_name] = \_\_value  
  
  
class Data():  
 file\_path = ""  
 data = []  
 pointer = 0  
  
 def \_\_init\_\_(self, path: str):  
 self.file\_path = path  
 self.data = self.parse(self.file\_path)  
  
 def \_\_str\_\_(self):  
 d\_str = '\n'.join([str(rm) for rm in self.data])  
 return f"Контейнер хранит в себе следующее:\n{d\_str}"  
  
 def \_\_repr\_\_(self):  
 return f"Data({[repr(rm) for rm in self.data]})"  
  
 def \_\_iter\_\_(self):  
 return self  
  
 def \_\_next\_\_(self):  
 if self.pointer >= len(self.data):  
 self.pointer = 0  
 raise StopIteration  
 else:  
 self.pointer += 1  
 return self.data[self.pointer - 1]  
  
 def \_\_getitem\_\_(self, item):  
 if not isinstance(item, int):  
 raise TypeError("Индекс должен быть целым числом.")  
  
 if 0 <= item < len(self.data):  
 return self.data[item]  
 else:  
 raise IndexError("Неверный индекс.")  
  
 def as\_generator(self):  
 self.pointer = 0  
  
 while self.pointer < len(self.data):  
 yield self.data[self.pointer]  
 self.pointer += 1  
  
 def sorted\_by\_name(self) -> list:  
 return sorted(self.data, key=lambda f: f.name)  
  
 def sorted\_by\_number(self) -> list:  
 return sorted(self.data, key=lambda f: f.id)  
  
 def select\_rows\_with\_likes\_count\_more\_than(self, value) -> list:  
 return [rm for rm in self.data if rm.group >= value]  
  
 def add\_new(self, id, name, email, group):  
 self.data.append(RowModel(id, name, email, group))  
 self.save(self.file\_path, self.data)  
  
 @staticmethod  
 def parse(path: str) -> list:  
 parsed = []  
  
 with open(path, "r") as raw\_csv:  
 for line in raw\_csv:  
 (id, name, email, group) = line.replace("\n", "").split(",")  
 parsed.append(RowModel(int(id), name, email, group))  
  
 return parsed  
  
 @staticmethod  
 def save(path, new\_data):  
 with open(path, "w") as f:  
 for rm in new\_data:  
 f.write(f"{rm.id},{rm.name},{rm.email},{rm.group}\n")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
  
 #init класса d, использующий методы RowModel, наследующийся от класса Row  
 d = Data("data-1.csv")  
  
 print("Используем итератор:\n")  
  
 for item in iter(d):  
 print(item)  
  
 print("-" \* 64)  
  
 print("Используем генератор:\n")  
  
 for item in d.as\_generator():  
 print(item)  
  
 print("=" \* 64)  
  
 print("Сортировка по ФИО:\n")  
  
 for item in d.sorted\_by\_name():  
 print(item)  
  
 print("-" \* 64)  
  
 print("Сортировка по id студента:\n")  
  
 for item in d.sorted\_by\_number():  
 print(item)  
  
 print("-" \* 64)  
  
 print("Выборка студентов, находящихся в группе ИВТАПбд22:\n")  
  
 for item in d.select\_rows\_with\_likes\_count\_more\_than("IVTAP"):  
 print(item)  
  
 print("=" \* 64)  
  
 # Добавим элемент (addnew и save)  
 print("Добавление данных:")  
 d.add\_new(input("Номер Студента: "), input("ФИО: "), input("email: "), input("Группа: "))  
 print()  
  
 for item in iter(d):  
 print(item)  
  
 print("=" \* 64)  
  
 # Вводишь индекс списка - получаешь его элемент (значение), вызваается в getitem  
 print("Выборка по индексу:")  
 id = int(input("Индекс: "))  
 print(f"\n{d[id]}")