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 № 14

Выполнил:

студент группы ИВТАПбд-21

Мухаметзянов Т. А.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту**

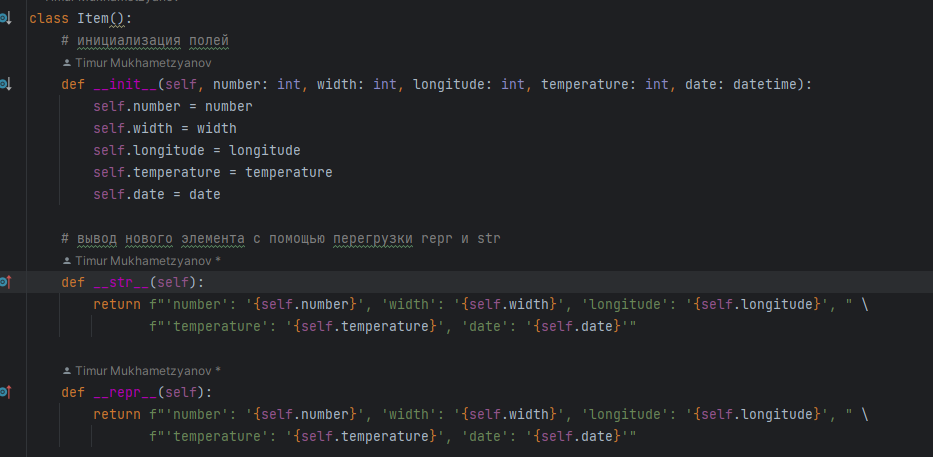
Необходимо переделать лабораторную работу №3 с использованием классов, описывающих предметную область, заданную вариантом, с реализацией следующих особенностей (вполне возможно, что предлагаемое в 3 лабе задание для этого нужно будет расширить):

**Требования к программе:**

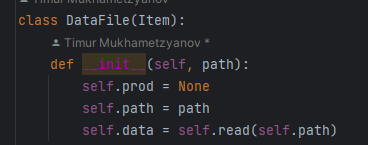
1. Класс должен содержать итератор
2. Должна быть реализована перегрузка стандартных операций (repr, например)
3. Должно быть реализовано наследование
4. Запись значений в свойства - только через \_\_setattr\_\_
5. Возможность доступа к элементам коллекции по индексу (\_\_getitem\_\_)
6. Должны быть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

**Описание реализации**

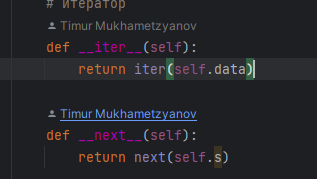
Класс Item для инициализации всех полей и перегрузка методов \_\_repr\_\_ и \_\_str\_\_ для вывода новых (добавленных данных на экран):



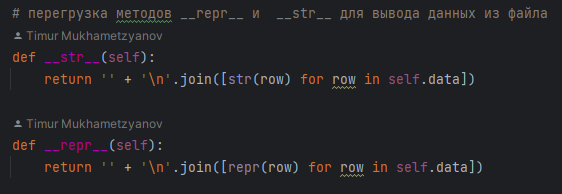
Наследованный от Item класс DataFile(Item):

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


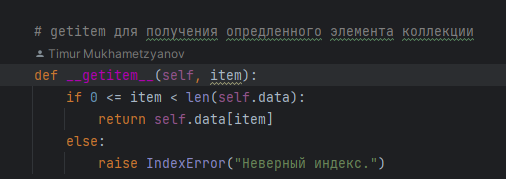
Итератор для вывода всего словаря:



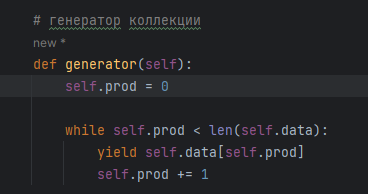
Перегрузка методов:



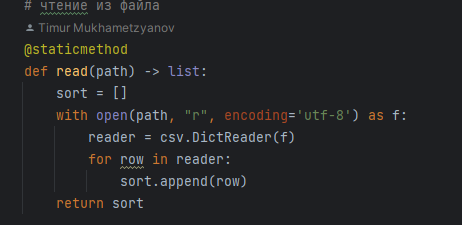
Getitem для получения определенной строки из словаря по индексы в словаре:



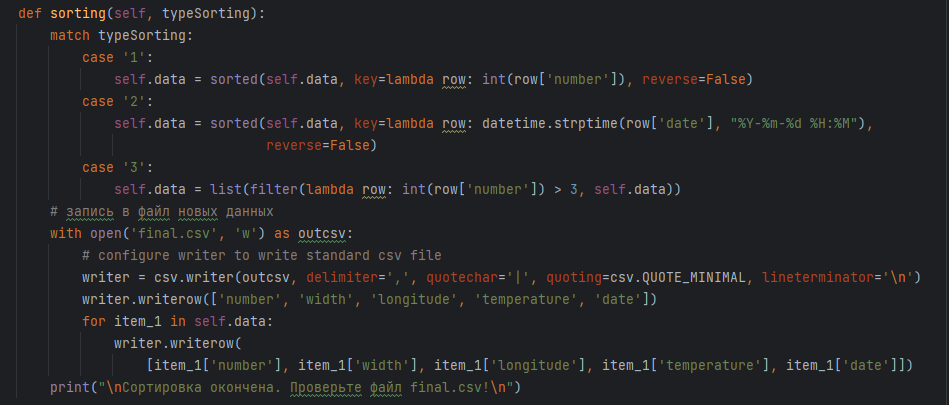
Генератор коллекции:



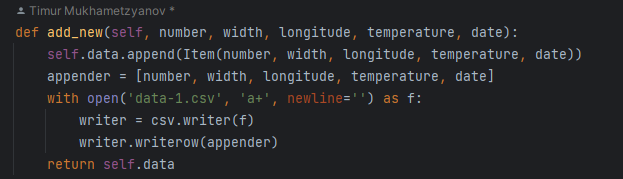
Чтение из файла:



Сортировка словаря по ключам, и сразу запись в новый .csv файл обновленных (сортированных данных):



Добавление новых данных в словарь:



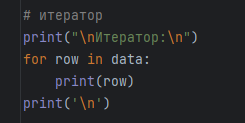
**Описание возникших затруднений**

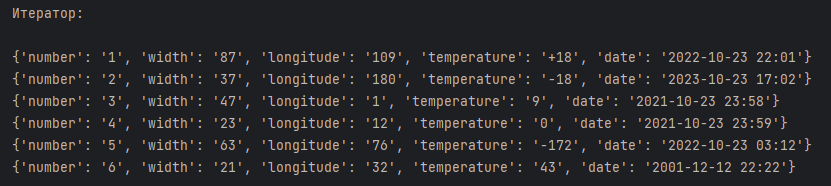
Работа с ООП, классами, методами, перегрузками методов, а также с наследованием была применена впервые на языке Python. ООП в этом языке достаточно сильно отличается от ООП других языков, на работу ушло много времени, тем не менее с выполнением поставленной задачи справился.

**Результаты работы программы**

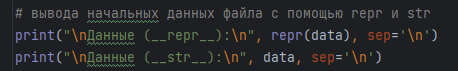


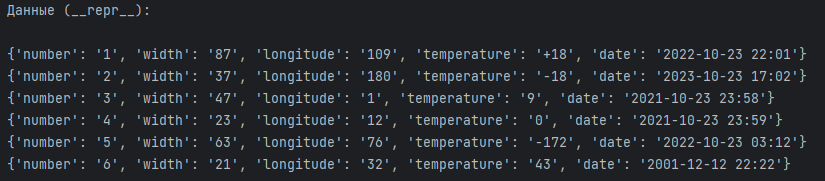
**1 рис. Экземпляр класса**





**2 рис. Вывод данных итератором**



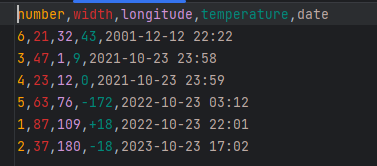


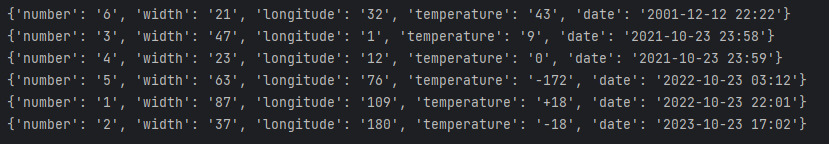
**3 рис. \_\_repr\_\_ и \_\_str\_\_**





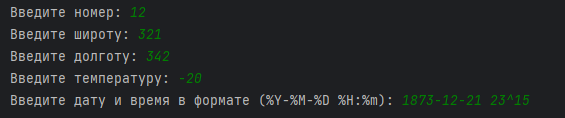
**4 рис. getitem**



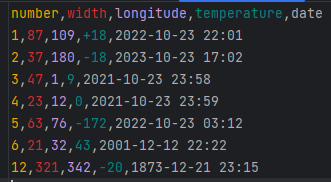


**5 рис. Сортировка по дате**

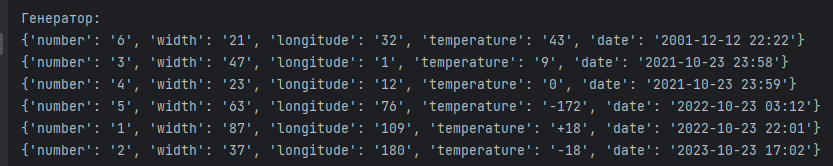


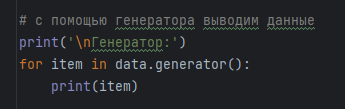






**6 рис. Добавление данных, вывод на экран и обновленный начальный файл с новой строкой**





**7 рис. Генератор**

**Код программы**

import csv  
from datetime import datetime  
  
  
# класс отдельного айтема для его инициализации в данных и выводе на экран при добавлении  
class Item():  
 # инициализация полей  
 def \_\_init\_\_(self, number: int, width: int, longitude: int, temperature: int, date: datetime):  
 self.number = number  
 self.width = width  
 self.longitude = longitude  
 self.temperature = temperature  
 self.date = date  
  
 # вывод нового элемента с помощью перегрузки repr и str  
 def \_\_str\_\_(self):  
 return f"'number': '{self.number}', 'width': '{self.width}', 'longitude': '{self.longitude}', " \  
 f"'temperature': '{self.temperature}', 'date': '{self.date}'"  
  
 def \_\_repr\_\_(self):  
 return f"'number': '{self.number}', 'width': '{self.width}', 'longitude': '{self.longitude}', " \  
 f"'temperature': '{self.temperature}', 'date': '{self.date}'"  
  
  
class DataFile(Item):  
 def \_\_init\_\_(self, path):  
 self.prod = None  
 self.path = path  
 self.data = self.read(self.path)  
  
 # итератор  
 def \_\_iter\_\_(self):  
 return iter(self.data)  
  
 def \_\_next\_\_(self):  
 return next(self.s)  
  
 # перегрузка методов \_\_repr\_\_ и \_\_str\_\_ для вывода данных из файла  
 def \_\_str\_\_(self):  
 return '' + '\n'.join([str(row) for row in self.data])  
  
 def \_\_repr\_\_(self):  
 return '' + '\n'.join([repr(row) for row in self.data])  
  
 def \_\_setattr\_\_(self, key, value):  
 self.\_\_dict\_\_[key] = value  
  
 # getitem для получения опредленного элемента коллекции  
 def \_\_getitem\_\_(self, item):  
 if 0 <= item < len(self.data):  
 return self.data[item]  
 else:  
 raise IndexError("Неверный индекс.")  
  
 # генератор коллекции  
 def generator(self):  
 self.prod = 0  
  
 while self.prod < len(self.data):  
 yield self.data[self.prod]  
 self.prod += 1  
  
 # чтение из файла  
 @staticmethod  
 def read(path) -> list:  
 sort = []  
 with open(path, "r", encoding='utf-8') as f:  
 reader = csv.DictReader(f)  
 for row in reader:  
 sort.append(row)  
 return sort  
  
 # сортировка  
 def sorting(self, typeSorting):  
 match typeSorting:  
 case '1':  
 self.data = sorted(self.data, key=lambda row: int(row['number']), reverse=False)  
 case '2':  
 self.data = sorted(self.data, key=lambda row: datetime.strptime(row['date'], "%Y-%m-%d %H:%M"),  
 reverse=False)  
 case '3':  
 self.data = list(filter(lambda row: int(row['number']) > 3, self.data))  
 # запись в файл новых данных  
 with open('final.csv', 'w') as outcsv:  
 # configure writer to write standard csv file  
 writer = csv.writer(outcsv, delimiter=',', quotechar='|', quoting=csv.QUOTE\_MINIMAL, lineterminator='\n')  
 writer.writerow(['number', 'width', 'longitude', 'temperature', 'date'])  
 for item\_1 in self.data:  
 writer.writerow(  
 [item\_1['number'], item\_1['width'], item\_1['longitude'], item\_1['temperature'], item\_1['date']])  
 print("\nСортировка окончена. Проверьте файл final.csv!\n")  
  
 # добавление новой записи в словарь  
 def add\_new(self, number, width, longitude, temperature, date):  
 self.data.append(Item(number, width, longitude, temperature, date))  
 appender = [number, width, longitude, temperature, date]  
 with open('data-1.csv', 'a+', newline='') as f:  
 writer = csv.writer(f)  
 writer.writerow(appender)  
 return self.data  
  
  
# создаем экземпляр класса  
data = DataFile('data-1.csv')  
  
# итератор  
print("\nИтератор:\n")  
for row in data:  
 print(row)  
print('\n')  
  
# получение элемента через getItem  
print(data.\_\_getitem\_\_(int(input("Enter number of elem: "))))  
  
# вывода начальных данных файла с помощью repr и str  
print("\nДанные (\_\_repr\_\_):\n", repr(data), sep='\n')  
print("\nДанные (\_\_str\_\_):\n", data, sep='\n')  
  
# сортировка  
print('\n')  
data.sorting(input("Выберите тип сортировки:\n1 - Сортировка по номеру записи (integer)"  
 "\n2 - Сортировка по дате\n3 - сортировка по критерию\n"))  
  
# с помощью итератора выводим новые данные  
for row in data:  
 print(row)  
  
data.add\_new(int(input("\nВведите номер: ")), int(input("Введите широту: ")), int(input("Введите долготу: ")),  
 int(input("Введите температуру: ")), input("Введите дату и время в формате (%Y-%M-%D %H:%m): "))  
  
# с помощью генератора выводим данные  
print('\nГенератор:')  
for item in data.generator():  
 print(item)