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 №3**

Дисциплина: «Разработка профессиональных приложений»

Файлы и словари

Вариант № 14

Выполнил:

студент группы ИВТАПбд-21

Мухаметзянов Т. А.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту**

Необходимо реализовать 14 вариант задания, а именно: из исходного .csv файла с начальными данными Временная шкала температурных показаний: №, широта, долгота, показание температуры, дата и время.

**Требования к программе:**

1.    Пусть дана некоторая директория (папка). Посчитайте количество файлов в данной директории (папке) и выведите на экран.

2.    Пусть   дан   файл   data.csv,  в   котором   содержится информация в соответствии с вариантом:

Считайте информацию из файла в соответствующую структуру (словарь):

2.1. Выведите информацию об объектах, отсортировав их по одному полю (строковому).

2.2. Выведите информацию об объектах, отсортировав их по одному полю (числовому).

2.3. Выведите информацию, соответствующую какому-либо критерию (например, для студентов - тех, у кого возраст больше какого-либо значения)

3.    Добавьте к программе возможность сохранения новых данных обратно в файл.

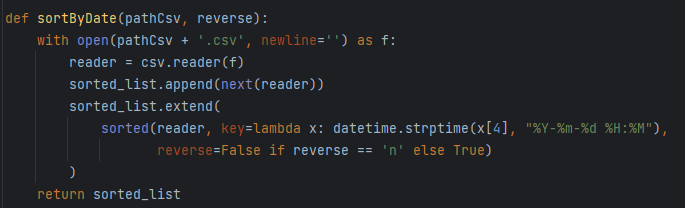
**Описание реализации**

Созданы 5 функций:

1. Функция вывода кол-ва файлов в заданной директории

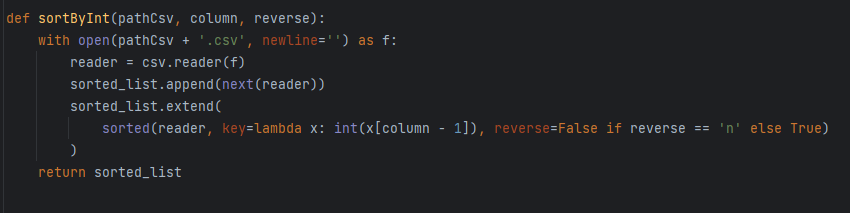
В которой вы считаем кол-во файлов в заданной папке при помощи os.walk (создает кортеж всех файлов в папке и возвращает генератор)

def countFilesInDir(path):  
 return sum(len(files) for \_, \_, files in os.walk(path))

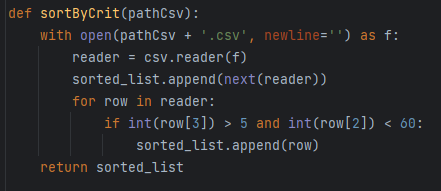
2. Функция Сортировки по Дате и времени

Происходит открытие файла формата .csv и считывание, дальше создаем пустой список, в который сначала при помощи next(reader) добавляем название столбцов, не учитывая их в сортировке. Дальше при помощи метода extend() и функции sorted по ключу = номеру колонки наших date данных сортируем наши записи и добавляем их в список.

3. Функция сортировки по целочисленному значению:

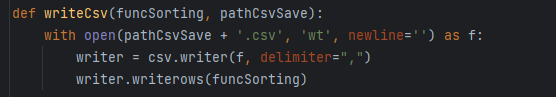


Происходит открытие файла формата .csv и считывание, дальше создаем пустой список, в который сначала при помощи next(reader) добавляем название столбцов, не учитывая их в сортировке. Дальше при помощи метода extend() и функции sorted по ключу = номеру колонки наших int данных сортируем наши записи и добавляем их в список. Номер колонки для сортировки пользователь вводит с клавиатуры.

4. Сортировка по критерию 

Происходит открытие файла формата .csv и считывание, дальше создаем пустой список, в который сначала при помощи next(reader) добавляем название столбцов, не учитывая их в сортировке. Далее пробегаемся по всем строчкам данных, если они подходят под условия, то добавляем их в список.

5. Запись в файл



Создается новый файл для записи данных с названием, который введет пользователь. В него записывается тот параметр (данные после функция сортировки), который укажет пользователь. Delimiter - разделитель данных.

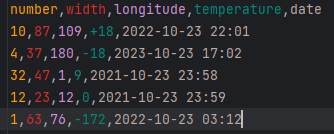
**Описание возникших затруднений**

Работа с чтением и записями в .csv файл была произведена впервые поэтому потребовалось время чтобы разобраться в процессе сохранения и чтения файлов в Python.

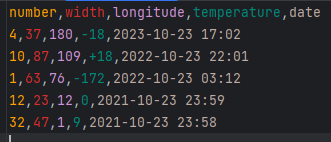
**Результаты работы программы**



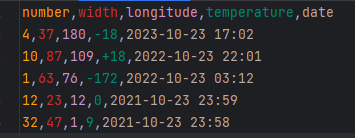
**1 рис. Кол-во файлов в папке Загрузки**



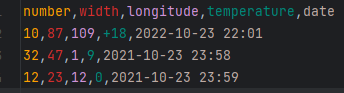
**2 рис. Исходный файл**



**3 рис. Сортировка по Дате и времени**



**4 рис. Сортировка по убыванию целоч. данных (3 столбец)**



**5 рис. Сортировка по критерию (данные с температурой больше или равной 0)**

**Код программы**

import csv  
from datetime import datetime  
import os  
  
  
def countFilesInDir(path):  
 return sum(len(files) for \_, \_, files in os.walk(path))  
  
  
print(countFilesInDir(input("path for scanning with format(disc:\ folder) ")), 'files in this dir')  
  
sorted\_list = []  
  
  
def sortByDate(pathCsv, reverse):  
 with open(pathCsv + '.csv', newline='') as f:  
 reader = csv.reader(f)  
 sorted\_list.append(next(reader))  
 sorted\_list.extend(  
 sorted(reader, key=lambda x: datetime.strptime(x[4], "%Y-%m-%d %H:%M"),  
 reverse=False if reverse == 'n' else True)  
 )  
 return sorted\_list  
  
  
def sortByInt(pathCsv, column, reverse):  
 with open(pathCsv + '.csv', newline='') as f:  
 reader = csv.reader(f)  
 sorted\_list.append(next(reader))  
 sorted\_list.extend(  
 sorted(reader, key=lambda x: int(x[column - 1]), reverse=False if reverse == 'n' else True)  
 )  
 return sorted\_list  
  
  
def sortByCrit(pathCsv):  
 with open(pathCsv + '.csv', newline='') as f:  
 reader = csv.reader(f)  
 sorted\_list.append(next(reader))  
 for row in reader:  
 if int(row[3]) > 5 and int(row[2]) < 60:  
 sorted\_list.append(row)  
 return sorted\_list  
  
  
def writeCsv(funcSorting, pathCsvSave):  
 with open(pathCsvSave + '.csv', 'wt', newline='') as f:  
 writer = csv.writer(f, delimiter=",")  
 writer.writerows(funcSorting)  
  
  
try:  
 match input("Enter type by sort - Date or number or crit format(d or n or c): "):  
 case 'd':  
 writeCsv(sortByDate(input('enter name file .csv for read: '), input("want reversed? (y, n) ")),  
 input("enter file name for save final file .csv: "))  
 case 'n':  
 writeCsv(sortByInt(input('enter name file .csv for read: '),  
 int(input('enter number of column for sorting 1 - 4: ')),  
 input("want reversed? (y, n) ")),  
 input("enter file name for save final file .csv: "))  
 case 'c':  
 writeCsv(sortByCrit(input('enter name file .csv for read: ')),  
 input("enter file name for save final file .csv: "))  
 case \_:  
 print("Error")  
except Exception:  
 print("Error")