Федеральное государственное образовательное бюджетное учреждение высшего образования "Финансовый университет при правительстве Российской Федерации"

**Практическое задание**

Реализация собственного пакета модулей по манипулированию

табличными данными

Выполнил: студент первого курса

Балашкин А.М., группа ПИ21-4

Проверил: доцент Волков А.В.

Преподаватель по дисциплине «Практикум по программированию»

Москва, 2021

Оглавление

[**Постановка задачи** 3](#_Toc91594252)

[**Алгоритм решения** 6](#_Toc91594253)

[**Решение** 8](#_Toc91594254)

[**Основной модуль** 8](#_Toc91594255)

[**Модуль работы с csv файлами** 14](#_Toc91594256)

[**Модуль работы с pickle файлами** 16](#_Toc91594257)

[**Модуль работы с TXT файлами** 17](#_Toc91594258)

[**Модуль получения типа данных столбцов** 19](#_Toc91594259)

[**Результат выполнения программы** 20](#_Toc91594260)

# **Постановка задачи**

На базе модулей: csv, pickle и прямой работы с файлами реализовать следующий базовый функционал:

1. функций load\_table, save\_table по загрузке/сохранению табличных данных во внутреннее представление модуля/из внутреннего представления модуля:

* файла формата csv (отдельный модуль с load\_table, save\_table в рамках общего пакета)
* файла формата pickle (отдельный модуль с load\_table, save\_table в рамках общего пакета), модуль использует структуру данных для представления таблицу, удобную автору работы.
* текстового файла (только функция save\_table сохраняющая в текстовом файле представление таблицы, аналогичное выводу на печать с помощью функции print\_table()).

Примечание: внутреннее представление может базироваться на словаре, где по разным ключам хранятся ключевые «атрибуты» таблицы, а значения таблицы хранятся в виде вложенных списков. Студент может выбрать другое внутреннее представление таблицы (согласовав его с преподавателем), в том числе, студенты знакомые с ООП на Python, могут реализовать собственный класс для таблицы.

При определении api модулей максимально полно использовать возможности сигнатур функций на Python (значения по умолчанию, запаковка/распаковка, в т.ч. именованных параметров, возвращение множественных значений), интенсивно выполнять проверки и возбуждать исключительные ситуации.

1. модуля с базовыми операциями над таблицами:

* get\_rows\_by\_number(start, [stop], copy\_table=False) – получение таблицы из одной строки или из строк из интервала по номеру строки. Функция либо копирует исходные данные, либо создает новое представление таблицы, работающее с исходным набором данных (copy\_table=False), таким образом изменения, внесенные через это представления будут наблюдаться и в исходной таблице.
* get\_rows\_by\_index(val1, … , copy\_table=False) – получение новой таблицы из одной строки или из строк со значениями в первом столбце, совпадающими с переданными аргументами val1, … , valN. Функция либо копирует исходные данные, либо создает новое представление таблицы, работающее с исходным набором данных (copy\_table=False), таким образом изменения, внесенные через это представления будут наблюдаться и в исходной таблице.
* get\_column\_types(by\_number=True) – получение словаря вида столбец: тип\_значений. Тип значения: int, float, bool, str (по умолчанию для всех столбцов). Параметр by\_number определяет вид значения столбец – целочисленный индекс столбца или его строковое представление.
* set\_column\_types(types\_dict, by\_number=True) – задание словаря вида столбец:тип\_значений. Тип значения: int, float, bool, str (по умолчанию для всех столбцов). Параметр by\_number определяет вид значения столбец – целочисленный индекс столбца или его строковое представление.
* get\_values(column=0) – получение списка значений (типизированных согласно типу столбца) таблицы из столбца либо по номеру столбца (целое число, значение по умолчанию 0, либо по имени столбца)
* get\_value(column=0) – аналог get\_values(column=0) для представления таблицы с одной строкой, возвращает не список, а одно значение (типизированное согласно типу столбца).
* set\_values(values, column=0) – задание списка значений values для столбца таблицы (типизированных согласно типу столбца) либо по номеру столбца (целое число, значение по умолчанию 0, либо по имени столбца).
* set\_value(column=0) – аналог set\_values(value, column=0) для представления таблицы с одной строкой, устанавливает не список значений, а одно значение (типизированное согласно типу столбца).
* print\_table() – вывод таблицы на печать.

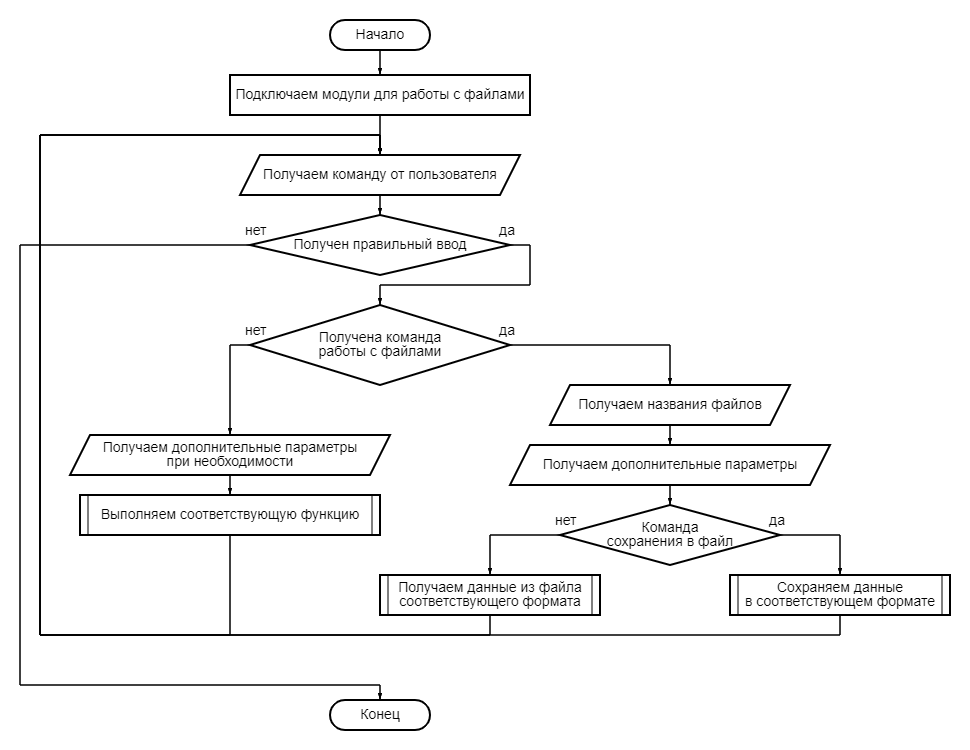
1. Для каждой функции должно быть реализована генерация не менее одного вида исключительных ситуаций.

Дополнительные задания:

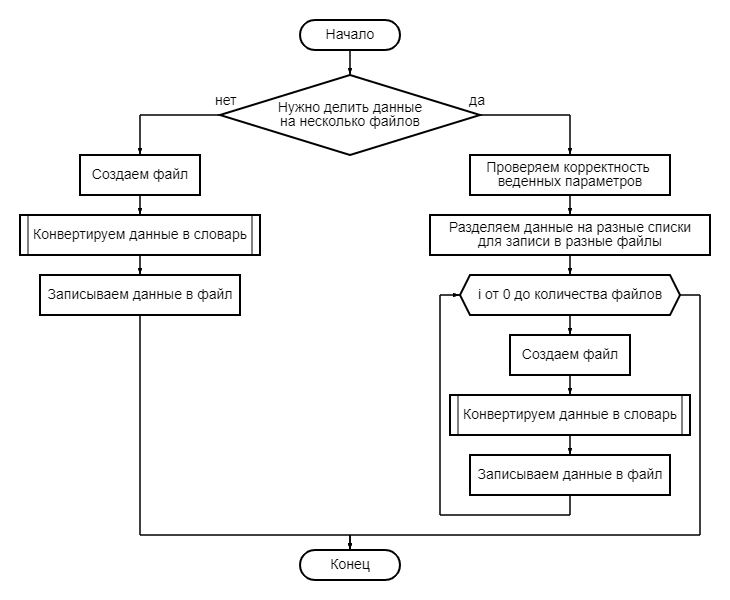
* В load\_table реализовать load\_table(file1, …) – поддержку загрузки таблицы, разбитой на несколько файлов (произвольное количество фйалов) (для форматов csv и pickle). В случае несоответствия структуры столбцов файлов вызывать исключительную ситуацию. **Сложность 1**
* В save\_table реализовать поддержку сохранения таблицы в разбитой на несколько файлов (произвольное количество фйалов) по параметру max\_rows, определяющему максимальное количество строк в файле. Файлы csv и pickle, полученные с помощью save\_table должны быть совместимы с load\_table из задания 1. **Сложность 1**
* Реализовать автоматическое определение типа столбцов по хранящимся в таблице значениям. Оформить как отдельную функцию и встроить этот функционал как опцию работы функции load\_table. **Сложность 2**
* Реализовать поддержку дополнительного типа значений «дата и время» на основе модуля datetime. **Сложность 2**

# **Алгоритм решения**

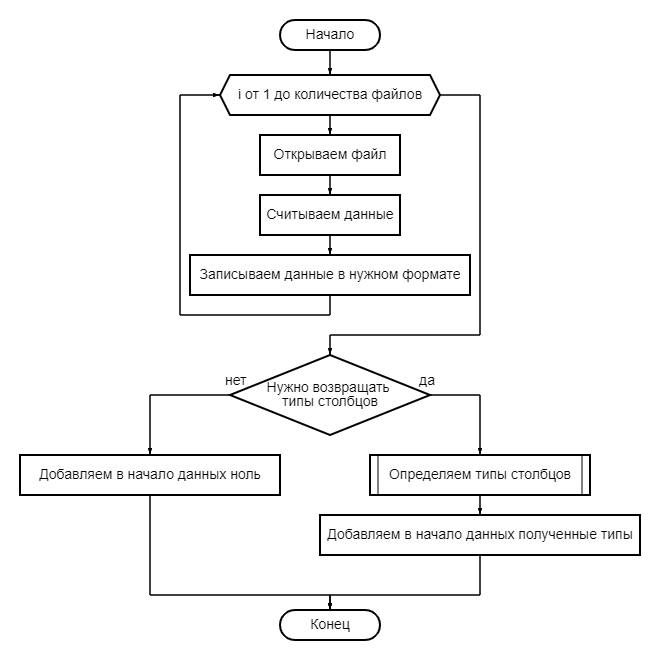
Основной модуль программы:



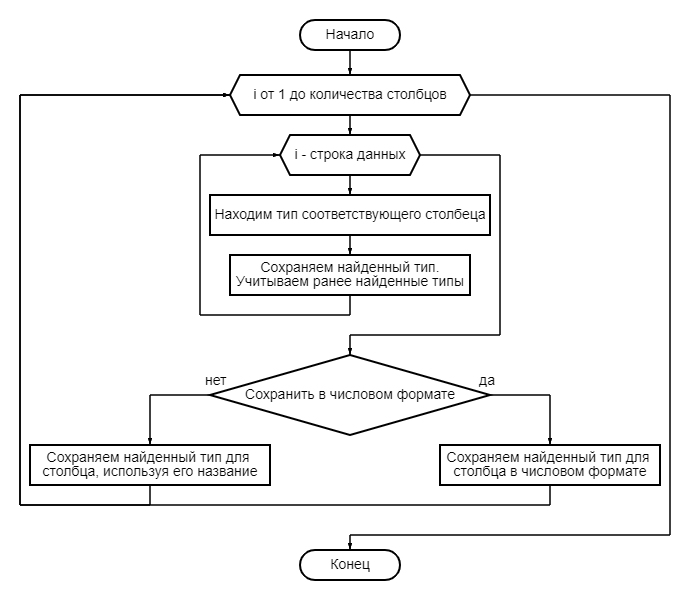
Функция сохранения данных:



Функция загрузки данных:



Модуль получения типа данных столбцов:



# **Решение**

## **Основной модуль**

import CSVModule, PickleModule, TXTModule *# Подключаем модулей для работы с файлами*import GetColumnsTypes *# Подключаем модуль получения типа данных столбцов*data = [] *# Таблица данных*types = [] *# Типы данных столбцов*def get\_values(column=0): *# Функция получение столбца* if isinstance(column, str): *# Если введена строка* if column.isdigit(): *# Если введено целое число* column = int(column) *# Превращаем переменную в целое число* else: *# Если введена строка* column = data[0].index(column) *# Получаем индекс - название столбца* result = [] *# Столбец* for i in data[1:]: *# Цикл по строкам* result.append(i[column]) *# Сохраняем нужный столбец* return result *# Возвращаем найденный столбец*def get\_value(column=0): *# Функция получения значения по столбцу* if isinstance(column, str): *# Если введена строка* if column.isdigit(): *# Если это целое число* column = int(column) *# То превращаем переменную в целое число* else: *# Если введена строка* column = data[0].index(column) *# Получаем индекс - название столбца* return data[1][column] *# Возвращаем нужное значение*def get\_rows\_by\_number(start, stop, copy\_table=False): *# Функция получения строк* global data *# Задаем глобальную переменную* complete\_data = [] *# Полученные строки* for i in range(start + 1, stop + 2): *# Цикл от начальной до конечной строки* complete\_data.append(data[i]) *# Записываем текущую строку* if copy\_table: *# Если нужно скопировать таблицу* data = [data[0]] + complete\_data *# То сохраняем ее с названиями столбцов* return complete\_data *# Возвращаем найденные строки*def get\_rows\_by\_index(copy\_table=False, \*val): *# Функция получения строк по индексам* global data *# Задаем глобальную переменную* complete\_data = [] *# Полученные строки* for i in list(\*val): *# Цикл по индексам* complete\_data.append(data[i + 1]) *# Сохраняем текущую строку* if copy\_table: *# Если нужно скопировать таблицу* data = [data[0]] + complete\_data *# Сохраняем ее с названиями столбцов* return complete\_data *# Возвращаем найденные строки*def set\_values(values, column=0): *# Метод задания столбца* if isinstance(column, str): *# Если столбец - строка* if column.isdigit(): *# Если столбец - число* column = int(column) *# Превращаем его в число* else: *# Если строка* column = data[0].index(column) *# Записываем индекс этого столбца* for i in range(1, len(data)): *# Цикл по строкам* data[i][column] = values[i - 1] *# Заменяем столбец на нужный*def set\_value(value, column=0): *# Метод задания первого элемента столбца* if isinstance(column, str): *# Если столбец - строка* if column.isdigit(): *# Если столбец - число* column = int(column) *# Превращаем его в число* else: *# Если строка* column = data[0].index(column) *# Записываем индекс этого столбца* data[1][column] = value *# Записываем нужное значение*def print\_table(): *# Метод вывода таблицы данных* for i in data: *# Цикл по строкам* print(\*i) *# Выводим текущую строку*operations = {  
 **'0'**: **'импорт данных из файла .csv'**,  
 **'1'**: **'импорт данных из файла .pkl'**,  
 **'2'**: **'импорт данных из файла .txt'**,  
 **'3'**: **'сохранение данных в файл .csv'**,  
 **'4'**: **'сохранение данных в файл .pkl'**,  
 **'5'**: **'сохранение данных в файл .txt'**,  
 **'6'**: **'типы данных столбцов'**,  
 **'7'**: **'вывод столбца'**,  
 **'8'**: **'вывод элемента столбца'**,  
 **'9'**: **'вывод таблицы'**,  
 **'10'**: **'вывод строк'**,  
 **'11'**: **'вывод строк по индексам'**,  
 **'12'**: **'задать столбец'**,  
 **'13'**: **'задать значение'**,  
} *# Возможные операции*while True: *# Бесконечный цикл для получения ввода от пользователя* print(**'Таблица'**, data) *# Выводим текущую таблицу* print(**'Типы данных столбов'**, types) *# Выводим типы данныхх текущей таблицы, если они есть* print(**'**\n**Работа с таблицами'**) *# Название программы* for i in list(operations.keys()): *# Цикл по всем операциям* print(i + **' - '** + operations[i]) *# Выводим все операции* a = input(**'**\n**Номер команды: '**) *# Получаем команду от пользователя* if a not in list(operations.keys()): *# Если введенная команда невозможна* exit() *# Выходим из программы* else: *# Если введена правильная команда* if int(a) < 6: *# Если это импорт или экспорт данных  
 # Получаем названия файлов* b = input(**'Если файлов несколько, то запишите их названия через пробел**\n**Название файла без расширения: '**)  
 while b == **''**:  
 print(**'Неправильный ввод'**)  
 b = input(**'Если файлов несколько, то запишите их названия через пробел**\n**Название файла без расширения: '**)  
  
 *# Получаем дополнительные параметры* c = **''** if int(a) < 3:  
 c = input(**'Получить таблицу с типами дыннах столбцов? (y or n): '**)  
 while c != **'y'** and c != **'n'**:  
 print(**'Неправильный ввод'**)  
 c = input(**'Получить таблицу с типами дыннах столбцов? (y or n): '**)  
 elif int(a) < 5:  
 c = input(**'Разделить таблицу на несколько файлов? Если да, то напишите максимальное кол-во строк'** +  
 **'в файле. Если нет, то напишите 0: '**)  
 while not c.isdigit():  
 print(**'Неправильный ввод'**)  
 c = input(**'Разделить таблицу на несколько файлов? Если да, то напишите максимальное'** +  
 **'кол-во строк в файле. Если нет, то напишите 0: '**)  
  
 *# Конвертируем полученные параметры для дальнейшей работы* if c == **'y'**:  
 c = True  
 elif c == **'n'**:  
 c = False  
 elif c.isdigit():  
 c = int(c)  
  
 if a == **'0'**: *# Если это импорт (0, 1, 2)* a = CSVModule.load\_table(c, b.split()) *# Выполняем загрузку таблицы* if a != False: *# Если удалось получить данные из файла* data = a[1:] *# Записываем данные* types = a[0] *# Записываем типы данных столбцов* else: *# Если не удалось получить данные из файла* print(**'Ошибка'**) *# Выводим сообщение об ошибке* elif a == **'1'**:  
 a = PickleModule.load\_table(c, b.split())  
 if a != False:  
 data = a[1:]  
 types = a[0]  
 else:  
 print(**'Ошибка'**)  
 elif a == **'2'**:  
 a = TXTModule.load\_table(c, b.split()[0])  
 if a != False:  
 data = a[1:]  
 types = a[0]  
 else:  
 print(**'Ошибка'**)  
 elif a == **'3'**: *# Если это экспорт (3, 4, 5)* print(CSVModule.save\_table(data, c, b.split())) *# Выводим результат операции сохранения таблицы* elif a == **'4'**:  
 print(PickleModule.save\_table(data, c, b.split()))  
 elif a == **'5'**:  
 print(TXTModule.save\_table(data, b.split()[0]))  
 elif a == **'6'**: *# Если это получение типа данных столбцов* c = input(**'Вывести названия или номера столбцов? (1 - названия, 2 - номера): '**) *# Получаем  
 # дополнительный параметр* while c != **'1'** and c != **'2'**:  
 print(**'Неправильный ввод'**)  
 c = input(**'Вывести названия или номера столбцов? (1 - названия, 2 - номера): '**)  
  
 *# Конвертируем полученный ввод для дальнейшей работы* if c == **'1'**:  
 c = False  
 else:  
 c = True  
  
 types = GetColumnsTypes.get\_column\_types(data, c) *# Получаем типы данных столбцов* elif a == **'7'**: *# Если нужно вывести столбец* b = input(**'Введите номер или имя столбца: '**) *# Получаем номер или название столбца* while b == **''**:  
 print(**'Неправильный ввод'**)  
 b = input(**'Введите номер или имя столбца: '**)  
  
 try: *# Пробуем выполнить операцию* print(get\_values(b)) *# Выводим результат операции получения столбца* except: *# Если не получилось* print(**'Такого столбца нет'**) *# Выводим сообщение, что такого столбца нет* elif a == **'8'**: *# Если нужно вывести значение в столбце* b = input(**'Введите номер или имя столбца: '**) *# Получаем номер или название столбца* while b == **''**:  
 print(**'Неправильный ввод'**)  
 b = input(**'Введите номер или имя столбца: '**)  
  
 try: *# Пробуем выполнить операцию* print(get\_value(b)) *# Выводим результат операции* except: *# Если не получилось* print(**'Такого столбца нет'**) *# Выводим сообщение об ошибке* elif a == **'9'**: *# Если нужно вывести таблицу данных* print\_table() *# Вызываем метод вывода таблицы данных* elif a == **'10'**: *# Если нужно вывести строки* b = input(**'Введите начальную строку: '**) *# Получаем номер начальной строки* while b == **''** or not b.isdigit():  
 print(**'Неправильный ввод'**)  
 b = input(**'Введите начальную строку: '**)  
  
 c = input(**'Введите конечную строку: '**) *# Получаем номер конечной строки* while c == **''** or not c.isdigit():  
 print(**'Неправильный ввод'**)  
 c = input(**'Введите конечную строку: '**)  
  
 d = input(**'Копировать таблицу? (y or n): '**) *# Узнаем нужно ли копировать таблицу* while d != **'y'** and d != **'n'**:  
 print(**'Неправильный ввод'**)  
 d = input(**'Копировать таблицу? (y or n): '**)  
  
 *# Конвертируем полученные данные* if d == **'y'**:  
 d = True  
 else:  
 d = False  
  
 try: *# Пробуем выполнить операцию* f = get\_rows\_by\_number(int(b), int(c), d) *# Получаем строки* print(\*data[0]) *# Выводим названия столбцов* for i in f: *# Цикл по полученный строкам* print(\*i) *# Выводим текущую строку* except: *# Если не получилось* print(**'Ошибка'**) *# Выводим сообщение об ошибке* elif a == **'11'**: *# Если нужно вывести строки по индексам* b = input(**'Введите индексы через пробел: '**) *# Получаем индексы строк* try: *# Проверяем корректность введенных данных* list(map(int, b.split()))  
 except: *# Если неправильный ввод* while True: *# Пробуем получить данные заново* print(**'Неправильный ввод'**) *# Выводим сообщение об ошибке* b = input(**'Введите индексы через пробел: '**) *# Получаем индексы строк* try: *# Проверяем корректность введенных данных* list(map(int, b.split()))  
 break *# Если получилось, останавливаем получение данных* except: *# Если не получилось* pass *# Заново получаем данные* d = input(**'Копировать таблицу? (y or n): '**) *# Узнаем нужно ли копировать таблицу* while d != **'y'** and d != **'n'**: *# Проверяем корректность введенных данных* print(**'Неправильный ввод'**) *# Если неправильный ввод, выводим сообщение об ошибке* d = input(**'Копировать таблицу? (y or n): '**) *# Заново получаем данные  
  
 # Конвертируем полученные данные* if d == **'y'**:  
 d = True  
 else:  
 d = False  
  
 try: *# Пробуем выполнить операцию* f = get\_rows\_by\_index(d, list(map(int, b.split()))) *# Получаем строки* print(\*data[0]) *# Выводим названия столбцов* for i in f: *# Цикл по полученным строкам* print(\*i) *# Выводим текущую строку* except: *# Если не получилось* print(**'Ошибка'**) *# Выводим сообщение об ошибке* elif a == **'12'**: *# Если нужно задать столбец* b = input(**'Номер или название столбца: '**) *# Получаем номер или название столбца* while b == **''**:  
 print(**'Неправильный ввод'**)  
 b = input(**'Номер или название столбца: '**)  
  
 c = input(**'Значения через пробел: '**) *# Получаем нужные значения* while c == **''**:  
 print(**'Неправильный ввод'**)  
 c = input(**'Значения через пробел: '**)  
  
 try: *# Пробуем выполнить операцию* set\_values(c.split(), b)  
 except: *# Если не получилось* print(**'Ошибка'**) *# Выводим сообщение об ошибке* elif a == **'13'**: *# Если нужно задать значение* b = input(**'Номер или название столбца: '**) *# Получаем номер или название столбца* while b == **''**:  
 print(**'Неправильный ввод'**)  
 b = input(**'Номер или название столбца: '**)  
  
 c = input(**'Значение: '**) *# Получаем нужное значение* while c == **''**:  
 print(**'Неправильный ввод'**)  
 c = input(**'Значение: '**)  
  
 try: *# Пробуем выполнить операцию* set\_value(c, b)  
 except: *# Если не получилось* print(**'Ошибка'**) *# Выводим сообщение об ошибке*

## **Модуль работы с csv файлами**

import csv *# Подключение библиотеки для работы с csv файлами*import GetColumnsTypes *# Подключаем модуль получения типа данных столбцов*def convert\_to\_dict(raw\_data): *# Функция конвертации данных из списка в словарь* complete\_data = [] *# Полученные словари в списке* for value in raw\_data[1:]: *# Цикл по строкам данных* complete\_dict = dict(zip(raw\_data[0], value)) *# Создаем словарь* complete\_data.append(complete\_dict) *# Записываем словарь* return complete\_data *# Возвращаем полученный словарь*def save\_table(data, max\_rows=0, \*files): *# Функция сохранения данных в файл* try: *# Пробуем выполнить сохранение данных* if max\_rows != 0: *# Если нужно разбивать на файлы* files = list(\*files) *# Превращаем полученные файлы в список для работы с ними* if max\_rows < 0 or len(data[1:]) // max\_rows > len(files): *# Если не получится разбить таблицу на файлы* 0 / 0 *# Вызываем ошибку для выхода* names = data[0] *# Названия столбцов* del data[0] *# Удаляем названия столбцов* complete\_data = [] *# Отформатированные дынные* for i in files: *# Цикл по кол-ву файлов* a = [] *# Данные для текущего файла* for j in range(min(max\_rows, len(data))): *# Цикл от 0 до длины списка данных или до максимального числа строк в файле* a.append(data[0]) *# Сохраняем первый элемент* del data[0] *# Удаляем первый элемент в списке данных* complete\_data.append(a) *# Сохраняем данные для текущего файла* if [] in complete\_data or len(data) > 0: *# Если есть файл с пустыми данными или остались данные* 0 / 0 *# Вызываем ошибку для выхода* for i in range(len(complete\_data)): *# Цикл по кол-ву файлов* file = open(files[i] + **'.csv'**, **'w'**) *# Создаем или открываем файл* data = convert\_to\_dict([names] + complete\_data[i]) *# Конвертируем данные в словарь* writer = csv.DictWriter(file, delimiter=**';'**,  
 fieldnames=list(data[0].keys())) *# Создаем объект для записи данных в файл* writer.writeheader() *# Записываем названия столбцов* for row in data: *# Цикл по данным* writer.writerow(row) *# Записываем текущую строку* file.close() *# Закрываем файл* else:  
 file = open(files[0][0] + **'.csv'**, **'w'**) *# Создаем или открываем файл* data = convert\_to\_dict(data) *# Конвертируем данные в словарь* writer = csv.DictWriter(file, delimiter=**';'**,  
 fieldnames=list(data[0].keys())) *# Создаем объект для записи данных в файл* writer.writeheader() *# Записываем названия столбцов* for row in data: *# Цикл по данным* writer.writerow(row) *# Записываем текущую строку* file.close() *# Закрываем файл* return True *# Если все получилось, возвращаем True* except: *# Если не удалось выполнить сохранение данных* return False *# Возвращаем False*def load\_table(types=False, \*filenames): *# Функция импорта данных из файла* try: *# Пробуем получить данные из файла* filenames = list(\*filenames) *# Превращаем полученные файлы в список для работы с ними* complete\_data = [] *# Полученные данные* for i in filenames: *# Цикл по файлам* file = open(i + **'.csv'**, **'r'**) *# Открываем файл* reader = csv.DictReader(file, delimiter=**';'**) *# Создаем объект для чтения данных из файла* for i, line in enumerate(reader): *# Цикл по строкам файла* if i == 0: *# Если список полученных данных пуст, то есть получаем первую строку* names = list(line.keys()) *# Получаем первую строку* if complete\_data == []: *# Если список полученных данных пуст* complete\_data.append(names) *# Записываем название столбцов, так как это первая строка* elif complete\_data[0] != names: *# Если в списке уже есть значения и  
 # текущие названия столбцов не равны уже сохраненным* 0 / 0 *# Вызываем ошибку для выхода* value = list(line.values()) *# Текущая строка* if value not in complete\_data: *# Если такой строкаи нет в полученных данных* complete\_data.append(value) *# Сохраняем текущую строку* file.close() *# Закрываем файл* if types: *# Если нужно вернуть данные с типами данных столбцов* a = GetColumnsTypes.get\_column\_types(complete\_data, False) *# Получаем типы данных столбцов* complete\_data = [a] + complete\_data *# Записываем их в начало списка* else: *# Иначе* complete\_data = [0] + complete\_data *# Добавляем ноль - разделитель* return complete\_data *# Возвращаем полученные данные* except: *# Если не получилось импортировать данные из файла* return False *# Возвращаем False*

## **Модуль работы с pickle файлами**

import GetColumnsTypes *# Подключаем модуль получения типа данных столбцов*pickle.DEFAULT\_PROTOCOL *# Устанавливаем стандартный протокол сериализации данных*def convert\_to\_dict(raw\_data): *# Функция конвертации данных из списка в словарь  
 # В этом методе сохранения файлов надо использовать словарь вида:  
 # ключ: [все значения для этого ключа]* complete\_data = {} *# Полученный словарь* for i in range(len(raw\_data[0])): *# Цикл по столбцам* value = [] *# Текущая строка* for j in raw\_data[1:]: *# Цикл по строкам* value.append(j[i]) *# Добавляем нужный столбец* complete\_data[raw\_data[0][i]] = value *# Сохраняем строку* return complete\_data *# Возвращаем полученный словарь*def save\_table(data, max\_rows, \*filename): *# Функция сохранения данных в файл* try: *# Пробуем выполнить сохранение данных* filename = list(\*filename) *# Превращаем полученные файлы в список для работы с ними* if max\_rows != 0: *# Если нужно разбивать на файлы* if max\_rows < 0 or len(data) // max\_rows > len(filename): *# Если не получится разбить таблицу на файлы* 0 / 0 *# Вызываем ошибку для выхода* names = data[0] *# Названия столбцов* del data[0] *# Удаляем названия столбцов* complete\_data = [] *# Отформатированные дынные* for i in filename: *# Цикл по кол-ву файлов* a = [] *# Данные для текущего файла* for j in range(min(max\_rows, len(data))): *# Цикл от 0 до длины списка данных или до максимального числа строк в файле* a.append(data[0]) *# Сохраняем первый элемент* del data[0] *# Удаляем первый элемент в списке данных* complete\_data.append(a) *# Сохраняем данные для текущего файла* if [] in complete\_data or len(data) > 0: *# Если есть файл с пустыми данными или остались данные* 0 / 0 *# Вызываем ошибку для выхода* for i in range(len(complete\_data)): *# Цикл по кол-ву файлов (через данные в файлах)* file = open(filename[i] + **'.pkl'**, **'wb'**) *# Открываем или создаем файл* data = convert\_to\_dict([names] + complete\_data[i]) *# Конвертируем данные в словарь* pickle.dump(data, file) *# Записываем данные в файл* file.close() *# Закрываем файл* else:  
 file = open(filename[0] + **'.pkl'**, **'wb'**) *# Открываем или создаем файл* data = convert\_to\_dict(data) *# Конвертируем данные в словарь* pickle.dump(data, file) *# Записываем данные в файл* file.close() *# Закрываем файл* return True *# Если все получилось, возвращаем True* except: *# Если не удалось выполнить сохранение данных в файл* return False *# Возвращаем False*def load\_table(types=False, \*filename): *# Функция импорта данных из файла* try: *# Пробуем получить данные из файла* filename = list(\*filename) *# Превращаем полученные файлы в список для работы с ними* complete\_data = [] *# Полученные данные* for i in filename: *# Цикл по файлам* file = open(i + **'.pkl'**, **'rb'**) *# Открываем файл* raw\_data = pickle.load(file) *# Получаем данные из файла* values = list(raw\_data.values()) *# Сохраняем значения в словаре данных* if complete\_data == []: *# Если еще не получены данные* complete\_data = [list(raw\_data.keys())] *# Сначала сохраняем названия столбцов* elif list(raw\_data.keys()) != complete\_data[0]: *# Если уже есть данные, то мы переходим к новому файлу  
 # Если стобцы в новом файле отличаются от начального* 0 / 0 *# То вызываем ошибку для выхода* for i in range(len(values[0])): *# Цикл от 0 до кол-ва столбцов* a = [] *# Текущая строка* for j in values: *# Цикл по значениям словаря данных* a.append(j[i]) *# Добавляем нужный столбец строки* complete\_data.append(a) *# Сохраняем текущую строку* file.close() *# Закрываем файл* if types: *# Если нужно вернуть данные с типами данных столбцов* a = GetColumnsTypes.get\_column\_types(complete\_data, False) *# Получаем типы данных столбцов* complete\_data = [a] + complete\_data *# Записываем их в начало списка* else: *# Иначе* complete\_data = [0] + complete\_data *# Добавляем ноль - разделитель* return complete\_data *# Возвращаем полученные данные* except: *# Если не получилось импортировать данные из файла* return False *# Возвращаем False*

## **Модуль работы с TXT файлами**

import GetColumnsTypes *# Подключаем модуль получения типа данных столбцов*def save\_table(data, filename): *# Функция сохранения данных в файл* try: *# Пробуем выполнить сохранение данных* file = open(filename + **'.txt'**, **'w+'**) *# Открываем или создаем файл* file.write(**''**) *# Очищаем файл* file.close() *# Закрываем файл* file = open(filename + **'.txt'**, **'a'**) *# Открываем файл для добавления строк* max\_len = 0 *# Максимальная длина столбца.* for i in data: *# Цикл по данным* for j in i: *# Цикл по столбцам данных* if len(j) > max\_len: *# Если найдена запись, которая больше предыдущей найденной* max\_len = len(j) *# Сохраняем ее длину* first = **''** *# Названия столбцов* for i in data[0]: *# Цикл по названиям столбцов* first += i + **' '** \* (max\_len - len(i)) + **'**\t**'** *# Сохраняем текущее название и отступ* first = first[:-1] + **'**\n**'** *# Убираем последний отступ и переносим на следующую строку* file.write(first) *# Записываем названия столбцов* for i in data[1:]: *# Цикл по строкам данных* complete\_data = **''** *# Отформатированные данные* for j in i: *# Цикл по элементам строки данных* complete\_data += j + **' '** \* (max\_len - len(j)) + **'**\t**'** *# Сохраняем элемент и отступ* complete\_data = complete\_data[:-1] + **'**\n**'** *# Убираем последний отступ и переносим на следующую строку* file.write(complete\_data) *# Записываем строку* file.close() *# Закрываем файл* return True *# Если получилось сохранить данные, возвращаем True* except: *# Если не удалось выполнить сохранение данных в файл* return False *# Возвращает False*def load\_table(types, filename): *# Функция импорта данных из файла* try: *# Пробуем получить данные из файла* file = open(filename + **'.txt'**, **'r'**) *# Открываем файл* raw\_data = file.read().split(**'**\n**'**) *# Получаем строки из файла* complete\_data = [] *# Полученные данные* for i in raw\_data: *# Цикл по считанным строкам* data = i.replace(**'**\t**'**, **''**).split() *# Превращаем строку в список, убирая отступы* complete\_data.append(data) *# Сохраняем отформатированную строку* while [] in complete\_data: *# Если в полученных данных содержится пустой список* del complete\_data[complete\_data.index([])] *# Удаляем его* file.close() *# Закрываем файл* if types: *# Если нужно вернуть данные с типами данных столбцов* a = GetColumnsTypes.get\_column\_types(complete\_data, False) *# Получаем типы данных столбцов* complete\_data = [a] + complete\_data *# Записываем их в начало списка* else: *# Иначе* complete\_data = [0] + complete\_data *# Добавляем ноль - разделитель* return complete\_data *# Возвращаем полученные данные* except: *# Если не удалось получить данные из файла* return False *# Возвращаем False*

## **Модуль получения типа данных столбцов**

from datetime import datetime *# Подключаем модуль для работы с датами*def get\_column\_types(data, by\_number=True): *# Функция получения типа данных столбцов* result = {} *# Словарь с полученными типами данных* for i in range(len(data[0])): *# Цикл по кол-ву столбцов* type = **''** *# Найденный тип* for j in data[1:]: *# Цикл по строкам* try: *# Пробуем конвертировать в целое число* int(j[i]) *# Попытка конвертации в целое число* try: *# Пробуем конвертировать в дробь* float(j[i]) *# Попытка конвертации в дробь* if float(j[i]) == int(j[i]): *# Если дробь и целое число равны* if type != **'str'** and type != **'float'**: *# И не найдены ранее строка или дробь* type = **'int'** *# То тип - целое число* else: *# Если это дробь* if type != **'str'**: *# И не найдена строка ранее* type = **'float'** *# То тип - дробь* except: *# Если не получилось получить дробь* if type != **'str'** and type != **'float'**: *# И не найдены ранее строка или дробь* type = **'int'** *# То тип - целое число* except: *# Если не получилось конвертировать в целое число* if j[i] in [**'True'**, **'False'**]: *# Если значение - логическая переменная* type = **'bool'** *# То тип - логическая переменная* else: *# Если не логическая переменная* get\_date = lambda d: datetime.strptime(d, **'%d.%m.%Y'**).date() <= datetime.today().date() *# Создаем  
 # функцию для проверки корректности даты* try: *# Пробуем конвертировать в дату* get\_date(j[i]) *# Попытка конвертации даты* type = **'datetime'** *# Если получилось, то тип - дата* except: *# Если не получилось* type = **'str'** *# То это строка* if by\_number: *# Если столбцы в цифрах* result[i] = type *# Добавляем ключ-значение (номер столбца-тип)* else: *# Если в названиях столбцов* result[data[0][i]] = type *# Добавляем ключ-значение (название столбца-тип)* return result *# Возвращаем найденные типы*

# **Результат выполнения программы**

