ORM CSV Saver Documentation

ORM CSV Saver is a Python tool for managing CSV data. In this there are two classes namely CSVSaver (Parent class) and CSVOperation (Child class). These classes provide with various method to perform CRUD operations on the ORM CSV.

Table of Contents

[CSV\_Saver class (Parent Class) 2](#_Toc153554835)

[1) Constructor(‘\_\_init\_\_’): 2](#_Toc153554836)

[2) Reading Method(‘reading’): 2](#_Toc153554837)

[3) Create Data Method(‘create\_data’): 3](#_Toc153554838)

[4) Updating Method(‘updating’): 3](#_Toc153554839)

[5) Deleting Data Rows Method(‘deleting\_data\_rows’): 4](#_Toc153554840)

[6) Create CSV Method(‘create\_CSV’-Static Method): 4](#_Toc153554841)

[7) Get New Data for Update Method(‘get\_new\_data\_for\_update’): 5](#_Toc153554842)

[8) Get New Data Method(‘get\_new\_data – Class Method’): 5](#_Toc153554843)

[CSV\_Operation 6](#_Toc153554844)

[1) Constructor(‘\_\_init\_\_’): 6](#_Toc153554845)

[2) Create Method(‘create’): 6](#_Toc153554846)

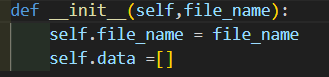
[3) Update Method(‘update’): 6](#_Toc153554847)

[USER INTERFACE 7](#_Toc153554848)

# CSV\_Saver class (Parent Class)

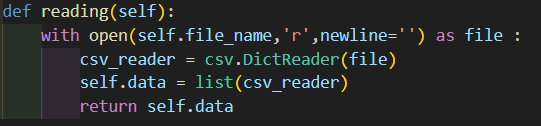
The CSV\_Saver class is designed to facilitate basic CRUD (Create, Read, Update, Delete) operations on a CSV file. It includes methods for reading data from a CSV file, creating new data entries, updating existing data, deleting specific rows, and creating a CSV file with a specified header. Below is the documentation for each method in the class:

## Constructor(‘\_\_init\_\_’):



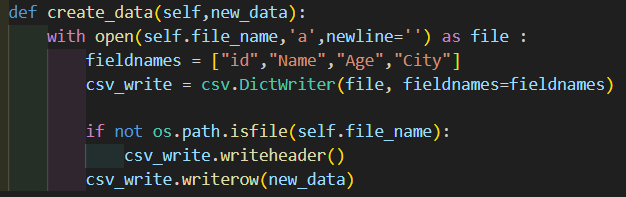
* Initializes an instance of the CSV\_Saver class with the specified CSV file name (file\_name). The data attribute is initialized as an empty list.

## Reading Method(‘reading’):



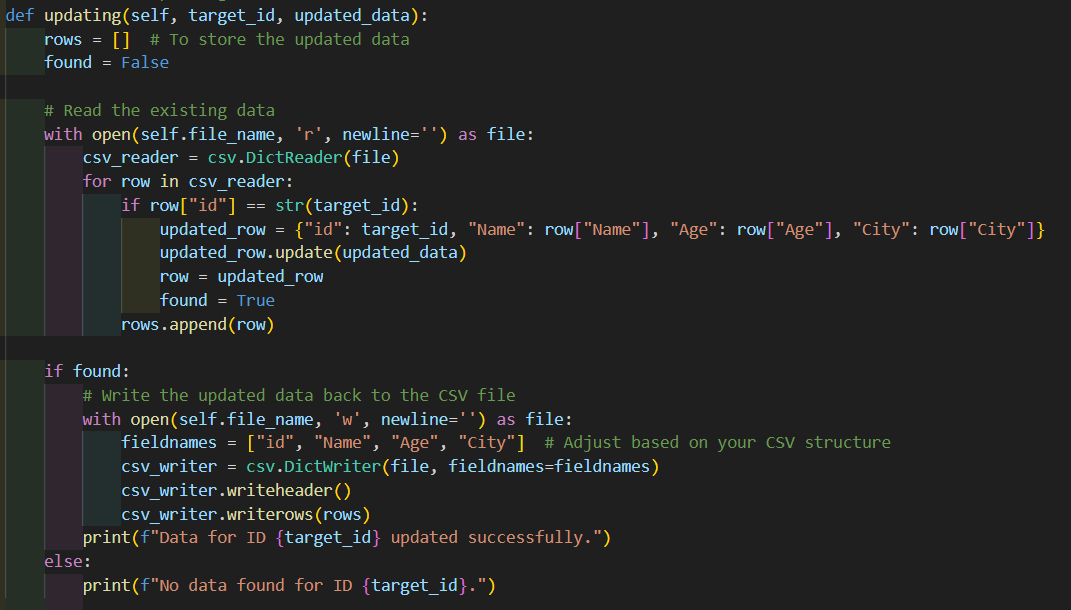
* Reads the data from the CSV file specified during object initialization and returns it as a list of dictionaries.
* Returns a list of dictionaries representing the data in the CSV file.

## Create Data Method(‘create\_data’):



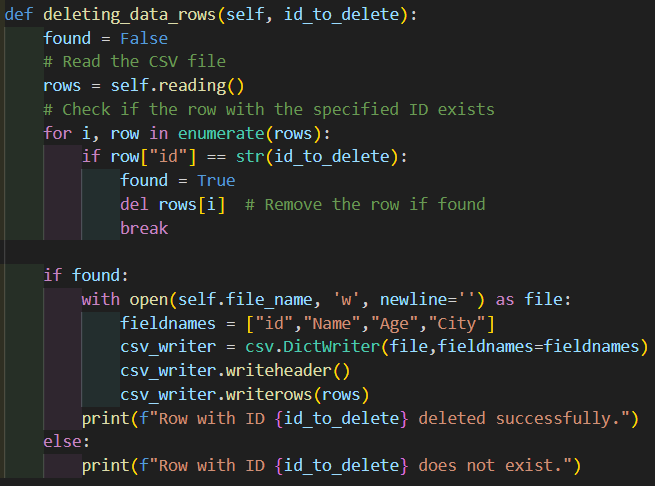
* It takes parameter ‘new\_data’ which is a dictionary representing the new entry to be added to the CSV file.
* Appends a new data entry to the CSV file. If the file does not exist, it creates a new file with the specified header.

## Updating Method(‘updating’):



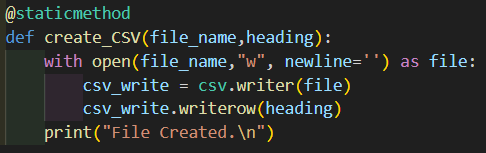
* It takes two arguments ‘target\_id(str)’ which is the unique identifier of the row to be updated and ‘updated\_data(dict)’ which is a dictionary containing the updated values for the specified row.
* It updates the specified row in the CSV file with the provided data.

## Deleting Data Rows Method(‘deleting\_data\_rows’):



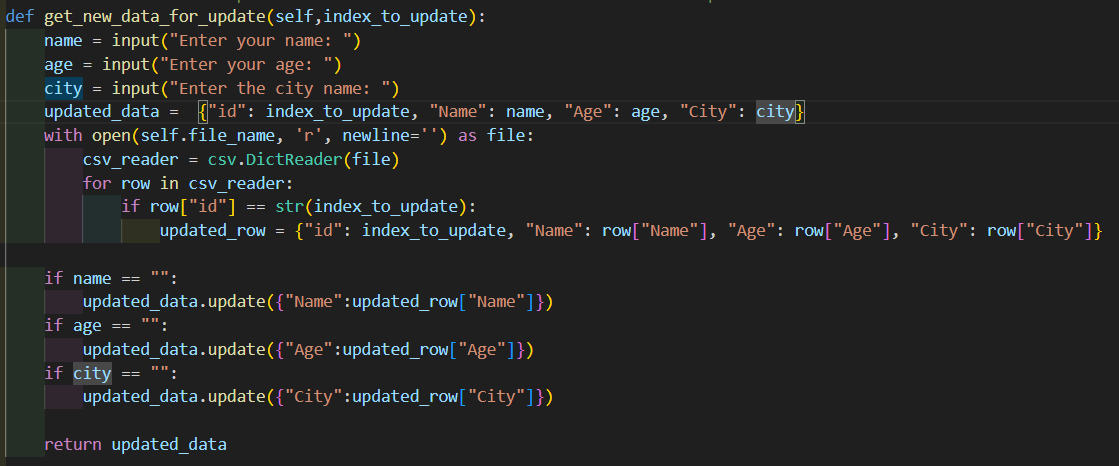
* It takes in the argument ‘id\_to\_delete’ which is a unique identifier of the row to be deleted.
* It deletes the row with the specified ID from the CSV file.

## Create CSV Method(‘create\_CSV’-Static Method):



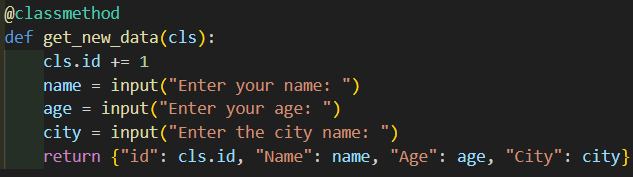
* It takes in two parameters ‘filename(str)’ which is the name of the CSV file to be created and ‘heading(list)’ is a list representing the header of the CSV file.
* This creates a new CSV file with specified header.

## Get New Data for Update Method(‘get\_new\_data\_for\_update’):



* This takes in ‘index\_to\_update(int)’ as an argument which us the index of the row to be updated.
* This method reads the user input to collect updated data for a specific row identified by its index.
* In this the csv file is read and the selected row’s data is put in a variable called updated\_row
* If the name, age or city field is sent as empty then it updates the data stored in updated\_row of specific field to the updated\_data.
* This method returns updated\_data.

## Get New Data Method(‘get\_new\_data – Class Method’):



* This method is a class method which reads user input to collect new data for creating a new entry in the CSV file and increments the ‘id’.
* This returns a dictionary containing the new data.

# CSV\_Operation

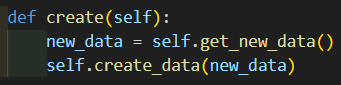
The CSV\_Operation class is a subclass of CSV\_Saver and provides a specialized interface for specific data manipulation tasks. It includes methods for creating and updating data entries.

### Constructor(‘\_\_init\_\_’):



* This takes in the parameter file\_name to be operated on.
* This initializes an instance of ‘CSV\_Operation’ class by calling the constructor of the parent class(‘CSV\_Saver’).

### Create Method(‘create’):



* This method invokes the’get\_new\_data’ method of the parent class to collect new data and then calls the ‘create\_data’ method to add the new entry to the CSV file.

### Update Method(‘update’):



* This method takes in two parameters ‘id(int)’ which is the index of the row to be updated and ‘updated\_data(dict)’ which is a dictionary containing the updated values for the specified row.
* This method invokes the updating method to update the specified row in the CSV file.

# USER INTERFACE

* The program provides a menu-driven interface for users to choose various operations, such as creating a table, writing data, reading data, updating data, deleting data, creating and updating data using the child class (CSV\_Operation), and exiting the program.
* The user is prompted with appropriate messages and instructions for each operation.