ReadSQL.py [-h] [--interactive | --no-interactive] [--multiprocessing | --no-multiprocessing]

[SQL\_filenames ...]

## Command “folder”

\folder (f) [foldername (f / fn) =] …;

Relative or absolute folder where sqlite3 files, inputs and outputs are placed.

As path separator, use “\” (e.g. “c:\”) or “/” (e.g. “~/..”) in Windows or Linux.

* Examples:

\f ~; \# go to home directory;

\f %TEMP%; \# go to temp directory;

\f ..; \# go to parent directory;

* Result:

Using folder 'C:\#Git'.

* Error:

Folder '~/' does not exist. Using current folder 'C:\#Git\ReadSQL'.

\load filename = …;

Load file with sqls and commands. Two object (data and columns) are created in memory.

\connect sqlite3 (,) "filename" (:memory:), "parse\_formats" (False)

"connect sqlite3", "connect sqlite", "connect sql3", "connect sql", "connect s", "c sqlite3",

"c sqlite", "c sql3", "c sql", "c s", "csqlite3", "csqlite", "csql3", "csql", "cs"

\ connect sqlite3 filename = …;

Create or use existing database in specified folder (using folder:foldername command)

\read filename = …, delimiter = …, text\_qualifier = …, read\_columns = True, strip\_columns = True, lines = 0;

Read file with data. Two object (data and columns) are created in memory.

\export filename = …, delimiter = …, text\_qualifier = …, write\_columns = True, strip\_columns = True, lines = 0;

Export data to a file. Use objects (data and columns) from memory.

\insert tablename = …;

Insert data in memory (data object) into table in database (---sqlite3,---mysql,---postgre commands) using column names. Table MUST be created in advance.

\print what = columns, columns = A B C (or columns = “A a” “B b” “C c” or columns = [A a, B b, C c] or columns = [“ A a ”, “ B b “, “ C c “]);

\pc from = …, to = …, step = …, list = 1 2 3 4;

Print columns in column object.

\print what = data, from = …, to = …, step = …, list = 1 2 3 (or list = [1,2,3]), columns = A B C (or columns = “A a” “B b” “C c” or columns = [A a, B b, C c] or columns = [“ A a ”, “ B b “, “ C c “]);

Print data in data object.

default = '/default/path/'

dir = raw\_input('Folder [%s]' % default)

dir = dir or default

7.1 Connector/Python Connection Arguments

A connection with the MySQL server can be established using either the mysql.connector.connect() function or the mysql.connector.MySQLConnection() class:

cnx = mysql.connector.connect(user='joe', database='test')

cnx = MySQLConnection(user='joe', database='test')

The following table describes the arguments that can be used to initiate a connection. An asterisk (\*) following an argument indicates a synonymous argument name, available only for compatibility with other Python MySQL drivers. Oracle recommends not to use these alternative names.

**Table 7.1 Connection Arguments for Connector/Python**

| **Argument Name** | **Default** | **Description** |
| --- | --- | --- |
| user**(**username**\*)** |  | The user name used to authenticate with the MySQL server. |
| password**(**passwd**\*)** |  | The password to authenticate the user with the MySQL server. |
| password1**,**password2**, and**password3 |  | For Multi-Factor Authentication (MFA) / FIDO; password1 is an alias for password. Added in 8.0.28. |
| database**(**db**\*)** |  | The database name to use when connecting with the MySQL server. |
| host | 127.0.0.1 | The host name or IP address of the MySQL server. |
| port | 3306 | The TCP/IP port of the MySQL server. Must be an integer. |