Circuitpython CircuitPython CSV Library Documentation

Release 1.0

Alec Delaney

Apr 10, 2023

CONTENTS

CircuitPython helper library for working with CSV files

CHAPTER

ONE

DEPENDENCIES

This driver depends on:

- Adafruit CircuitPython
- MicroPython's regular expression library (re)

You can find which Adafruit boards have the `re` library here.

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle or individual libraries can be installed using circup.

CHAPTER

TWO

INSTALLING TO A CONNECTED CIRCUITPYTHON DEVICE WITH CIRCUP

Make sure that you have `circup` installed in your Python environment. Install it with the following command if necessary:

```
pip3 install circup
```

With `circup` installed and your CircuitPython device connected use the following command to install:

```
circup install circuitpython-csv
```

Or the following command to update an existing version:

```
circup update
```

CHAPTER

THREE

INSTALLING FROM PYPI

Note: This library is provided on PyPI so that code developed for microcontrollers with this library will also run on computers like the Raspberry Pi. If you just need a package for working with CSV files on a computer or SBC only, consider using the Python standard library's `csv` module instead.

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install circuitpython-csv
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install circuitpython-csv
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .venv
source .venv/bin/activate
pip3 install circuitpython-csv
```

CHAPTER

FOUR

USAGE EXAMPLE

```
import board
import sdcardio
import storage
import circuitpython_csv as csv

# Initialize SD card
spi = board.SPI()
sdcard = sdcardio.SDCard(spi, board.D10)
vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd")

# Write the CSV file!
with open("/sd/testwrite.csv", mode="w", encoding="utf-8") as writablefile:
    csvwriter = csv.writer(writablefile)
    csvwriter.writerow(["I", "love", "CircuitPython", "!"])
    csvwriter.writerow(["Spam"] * 3)
```

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER

SIX

DOCUMENTATION

For information on building library documentation, please check out this guide.

CHAPTER

SEVEN

ATTRIBUTION

Some code contained here is ported from CPython, dual licensed by the Python Software Foundation under the PSF License verion 2 and the Zero-Clause BSD license.

CHAPTER

EIGHT

TABLE OF CONTENTS

8.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/csv_simpletest.py

```
# SPDX-FileCopyrightText: 2017 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Alec Delaney
#
# SPDX-License-Identifier: MIT

import board
import sdcardio
import storage
import circuitpython_csv as csv

# Initialize SD card
spi = board.SPI()
sdcard = sdcardio.SDCard(spi, board.D10)
vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd")

# Write the CSV file!
with open("/sd/testwrite.csv", mode="w", encoding="utf-8") as writablefile:
    csvwriter = csv.writer(writablefile)
    csvwriter.writerow(["I", "love", "CircuitPython", "!"])
    csvwriter.writerow(["Spam"] * 3)
```

8.2 Disklogger

Logging data to .CSV file on CircuitPython Disk

Listing 2: examples/csv_disklogger.py

```
# SPDX-FileCopyrightText: 2022 @Skicka for Adafruit Industries / Hakcat
# Logging data to .CSV file on CircuitPython Disk
# SPDX-License-Identifier: MIT

```

(continues on next page)

(continued from previous page)

```
# If you get a read-only filesystem error, add "storage.remount('/', False)" in boot.py
# Make sure you add a way to reverse this in boot.py or your CP device won't show up via USB
# See example below:
# https://learn.adafruit.com/getting-started-with-raspberry-pi-pico-circuitpython/data-logger

import os
import random
import circuitpython_csv as csv

# Check if .CSV file is already present. If not, we write CSV headers.
all_files = os.listdir()  ## List all files in directory
if "datelog.csv" not in all_files:
    with open("datelog.csv", mode="w", encoding="utf-8") as writablefile:
        csvwriter = csv.writer(writablefile)
        csvwriter.writerow(["Year", "Month", "Day", "Hour", "Minute"])

# Now that the file exists (or already did) we make a random date
year = random.randint(1999, 2022)
month = random.randint(1, 12)
day = random.randint(1, 30)
hour = random.randint(0, 23)
minute = random.randint(0, 60)

# We append this to the .CSV file
with open("datelog.csv", mode="a", encoding="utf-8") as writablefile:
    csvwriter = csv.writer(writablefile)
    csvwriter.writerow([year, month, day, hour, minute])

# Finally, we try to read back the last line in the CSV file to make sure it wrote.
with open("datelog.csv", "r", encoding="utf-8") as file:
    data = file.readlines()
    print(data[-1])
```

8.3 DictWriter test

Illustrate an example of the `DictWriter` class

Listing 3: examples/csv_dictwritertest.py

```
# SPDX-FileCopyrightText: 2017 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Alec Delaney
#
# SPDX-License-Identifier: MIT

import board
import sdcardio
import storage
import circuitpython_csv as csv
```

(continues on next page)

(continued from previous page)

```
10
11  # Initialize SD card
12  spi = board.SPI()
13  sdcard = sdcardio.SDCard(spi, board.D10)
14  vfs = storage.VfsFat(sdcard)
15  storage.mount(vfs, "/sd")
16
17  header = ["name", "fav-board", "fav-wing"]
18
19  my_info = {
20      "name": "Blinka",
21      "fav-board": "Feather M4 Express",
22      "fav-wing": "Adalogger FeatherWing",
23  }
24
25  with open("/sd/testwrite.csv", mode="w", encoding="utf-8") as writablefile:
26      csvwriter = csv.DictWriter(writablefile, header)
27      csvwriter.writeheader()
28      csvwriter.writerow(my_info)
```

8.4 circuitpython_csv

CircuitPython helper library for working with CSV files

- Author(s): Alec Delaney

8.4.1 Implementation Notes

Hardware:

None

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases

class circuitpython_csv.**DictReader**(*f: TextIOWrapper*, *fieldnames: Optional[Sequence[str]] = None*, *restkey: Optional[str] = None*, *restval: Optional[Any] = None*, ***kwargs*)

CSV reader that maps rows to a dict according to given or inferred fieldnames, it also accepts the delimiter and quotechar keywords

Parameters

- **f** (*io.TextIOWrapper*) – The open file to read from
- **fieldnames** (*Sequence[str]*) – (Optional) The fieldnames for each of the columns, if none is given, it will default to the whatever is in the first row of the CSV file
- **restkey** (*str*) – (Optional) A key name for values that have no key (row is larger than the length of fieldnames), default is None
- **restval** (*Any*) – (Optional) A default value for keys that have no values (row is small than the length of fieldnames, default is None

class circuitpython_csv.**DictWriter**(*f: TextIOWrapper*, *fieldnames: Sequence[str]*, *restval: str = ''*, *extrasaction: str = 'raise'*, ***kwargs*)

CSV writer that uses a dict to write the rows according fieldnames, it also accepts the delimiter and quotechar keywords

Parameters

- **f** (*io.TextIOWrapper*) – The open file to write to
- **fieldnames** (*Sequence[str]*) – The fieldnames for each of the comlumns
- **restval** (*str*) – A default value for keys that have no values
- **extrasaction** (*str*) – The action to perform if a key is encountered when parsing the dict that is not included in the fieldnames parameter, either "raise" or "ignore". Ignore raises a ValueError, and "ignore" simply ignore that key/value pair. Default behavior is "raise"

writeheader() → None

Writes the header row to the CSV file

writerow(*rowdict: Dict[str, Any]*) → None

Writes a row to the CSV file

Parameters

rowdict (*Dict[str, Any]*) – The row to write as a dict, with keys of the DictWriter's fieldnames parameter; values must be str or be able to be cast to str

writerows(*rowdicts: Iterable[Dict[str, Any]]*) → None

Writes multiple rows to the CSV files

Parameters

rowdicts (*Iterable[Dict[str, Any]]*) – An iterable item that yields multiple rows to write; values in those rows must be str or be able to be cast to str

class circuitpython_csv.**reader**(*csvfile: TextIOWrapper*, *delimiter: str = ','*, *quotechar: str = '"'*)

Basic CSV reader class that behaves like CPython's `csv.reader()`

Parameters

- **csvfile** (*io.TextIOWrapper*) – The open file to read from
- **delimiter** (*str*) – (Optional) The CSV delimiter, default is comma (,)
- **quotechar** (*str*) – (Optional) The CSV quote character for encapsulating special characters including the delimiter, default is double quotation mark (")

class circuitpython_csv.**writer**(*csvfile: TextIOWrapper*, *delimiter: str = ','*, *quoterchar: str = '"'*)

Basic CSV writer class that behaves like CPython's `csv.writer()`

Parameters

- **csvfile** (*io.TextIOWrapper*) – The open CSVfile to write to
- **delimiter** (*str*) – (Optional) The CSV delimiter, default is comma (,)
- **quotechar** (*str*) – (Optional) The CSV quote character for encapsulating special characters including the delimiter, default is double quotation mark (")

writerow(*seq: Sequence[Any]*) → None

Write a row to the CSV file

Parameters

seq (*Sequence[Any]*) – The list of values to write, which must all be str or be able to be cast to str

writerows(*rows: Iterable[Sequence[Any]]*) → None

Write multiple rows to the CSV file

Parameters

rows (*Iterable[Sequence[Any]]*) – An iterable item that yields multiple rows to write (e.g., list)

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX