Adafruit Circup Documentation

Release 1.0

Adafruit Industries

Jul 28, 2023

CONTENTS

1 Installation **3**

2 What does Circup Do? **5**

3 Usage **7**

4 Library Name Autocomplete **11**

5 How to Activate Library Name Autocomplete **13**

6 Contributing **15**
- 6.1 Licensing 15
- 6.2 Developer Setup 15

7 API **17**

8 License **23**

Python Module Index **25**

Index **27**

A tool to manage and update libraries (modules) on a CircuitPython device.

Contents

- *CircUp*
  - *Installation*
  - *What does Circup Do?*
  - *Usage*
  - *Library Name Autocomplete*
  - *How to Activate Library Name Autocomplete*
- *Contributing*
  - *Licensing*
  - *Developer Setup*
    - *How Does Circup Work?*
- *API*
- *License*

CHAPTER ONE

INSTALLATION

Circup requires Python 3.5 or higher.

In a virtualenv, `pip install circup` should do the trick. This is the simplest way to make it work.

If you have no idea what a virtualenv is, try the following command, `pip3 install --user circup`.

Note: If you use the `pip3` command to install CircUp you must make sure that your path contains the directory into which the script will be installed. To discover this path,

- On Unix-like systems, type `python3 -m site --user-base` and append `bin` to the resulting path.
- On Windows, type the same command, but append `Scripts` to the resulting path.

CHAPTER TWO

WHAT DOES CIRCUP DO?

Each CircuitPython library on the device usually has a version number as metadata within the module.

This utility looks at all the libraries on the device and checks if they are the most recent (compared to the versions found in the most recent version of the Adafruit CircuitPython Bundle and Circuitpython Community Bundle). If the libraries are out of date, the utility helps you update them.

The Adafruit CircuitPython Bundle can be found here:

https://github.com/adafruit/Adafruit_CircuitPython_Bundle/releases/latest

Full details of these libraries, what they're for and how to get them, can be found here:

https://circuitpython.org/libraries

The Circuitpython Community Bundle can be found here:

https://github.com/adafruit/CircuitPython_Community_Bundle/releases/latest

CHAPTER

THREE

USAGE

If you need more detailed help using Circup see the Learn Guide article "Use CircUp to easily keep your CircuitPython libraries up to date".

First, plug in a device running CircuiPython. This should appear as a mounted storage device called `CIRCUITPY`.

To get help, just type the command:

```
$ circup
Usage: circup [OPTIONS] COMMAND [ARGS]...

  A tool to manage and update libraries on a CircuitPython device.

Options:
  --verbose         Comprehensive logging is sent to stdout.
  --version         Show the version and exit.
  --path DIRECTORY  Path to CircuitPython directory. Overrides automatic
                    path detection.
  --help            Show this message and exit.
  -r --requirement  Supports requirements.txt tracking of library
                    requirements with freeze and install commands.

Commands:
  freeze        Output details of all the modules found on the connected...
  install       Installs .mpy version of named module(s) onto the device.
  install --py  Installs .py version of named module(s).
  list          Lists all out of date modules found on the connected...
  show          Show the long list of all available modules in the bundle.
  show <query>  Search the names in the modules in the bundle for a match.
  uninstall     Uninstall a named module(s) from the connected device.
  update        Update modules on the device. Use --all to automatically update
                all modules.
```

To automatically install all modules imported by `code.py`, `$ circup install --auto`:

```
$ circup install --auto
Found device at /Volumes/CIRCUITPY, running CircuitPython 7.0.0-alpha.5.
Searching for dependencies for: ['adafruit_bmp280']
Ready to install: ['adafruit_bmp280', 'adafruit_bus_device', 'adafruit_register']

Installed 'adafruit_bmp280'.
Installed 'adafruit_bus_device'.
Installed 'adafruit_register'.
```

To search for a specific module containing the name bme: `$ circup show bme`:

```
$ circup show bme
Found device at /Volumes/CIRCUITPY, running CircuitPython 6.1.0-beta.2.
adafruit_bme280
adafruit_bme680
2 shown of 257 packages.
```

To show version information for all the modules currently on a connected CIRCUITPYTHON device:

```
$ circup freeze
adafruit_binascii==v1.0
adafruit_bme280==2.3.1
adafruit_ble==1.0.2
```

With `$ circup freeze -r`, Circup will save, in the current working directory, a requirements.txt file with a list of all modules currently installed on the connected device.

To list all the modules that require an update:

```
$ circup list
The following modules are out of date or probably need an update.

Module             Version  Latest
------------------ -------- --------
adafruit_binascii  v1.0     1.0.1
adafruit_ble       1.0.2    4.0
```

To interactively update the out-of-date modules:

```
$ circup update
Found 3 module[s] needing update.
Please indicate which modules you wish to update:

Update 'adafruit_binascii'? [y/N]: Y
OK
Update 'adafruit_ble'? [y/N]: Y
OK
```

Install a module or modules onto the connected device with:

```
$ circup install adafruit_thermal_printer
Installed 'adafruit_thermal_printer'.

$ circup install adafruit_thermal_printer adafruit_bus_io
Installed 'adafruit_thermal_printer'.
Installed 'adafruit_bus_io'.
```

If you need to work with the original .py version of a module, use the –py flag.

> $ circup install –py adafruit_thermal_printer Installed ‘adafruit_thermal_printer’.

You can also install a list of modules from a requirements.txt file in the current working directory with:

```
$ circup install -r requirements.txt
Installed 'adafruit_bmp280'.
Installed 'adafruit_lis3mdl'.
Installed 'adafruit_lsm6ds'.
Installed 'adafruit_sht31d'.
Installed 'neopixel'.
```

Uninstall a module or modules like this:

```
$ circup uninstall adafruit_thermal_printer
Uninstalled 'adafruit_thermal_printer'.

$ circup uninstall adafruit_thermal_printer adafruit_bus_io
Uninstalled 'adafruit_thermal_printer'.
Uninstalled 'adafruit_bus_io'.
```

Use the `--verbose` flag to see the logs as the command is working:

```
$ circup --verbose freeze
Logging to /home/ntoll/.cache/circup/log/circup.log

10/18/2020 00:54:43 INFO: ### Started Circup ###
10/18/2020 00:54:43 INFO: Found device: /Volumes/CIRCUITPY
Found device at /Volumes/CIRCUITPY, running CircuitPython 6.0.0-alpha.1-1352-
↪gf0b37313c.
10/18/2020 00:54:44 INFO: Freeze
10/18/2020 00:54:44 INFO: Found device: /Volumes/CIRCUITPY
... etc ...
```

The `--path` flag let's you pass in a different path to the CircuitPython mounted volume. This is helpful when you have renamed or have more than one CircuitPython devices attached:

```
$ circup --path /run/media/user/CIRCUITPY1 list
```

The `--version` flag will tell you the current version of the `circup` command itself:

```
$ circup --version
CircUp, A CircuitPython module updater. Version 0.0.1
```

That's it!

CHAPTER

FOUR

LIBRARY NAME AUTOCOMPLETE

When enabled, circup will autocomplete library names, simliar to other command line tools.

For example:

> `circup install n` + tab -`circup install neopixel` (+tab: offers `neopixel` and `neopixel_spi` completions)
>
> `circup install a` + tab -`circup install adafruit\_` + m a g + tab -`circup install adafruit_magtag`

CHAPTER

FIVE

HOW TO ACTIVATE LIBRARY NAME AUTOCOMPLETE

In order to activate shell completion, you need to inform your shell that completion is available for your script. Any Click application automatically provides support for that.

For Bash, add this to ~/.bashrc:

```
eval "$(_CIRCUP_COMPLETE=bash_source circup)"
```

For Zsh, add this to ~/.zshrc:

```
eval "$(_CIRCUP_COMPLETE=zsh_source circup)"
```

For Fish, add this to ~/.config/fish/completions/foo-bar.fish:

```
eval (env _CIRCUP_COMPLETE=fish_source circup)
```

Open a new shell to enable completion. Or run the eval command directly in your current shell to enable it temporarily. ### Activation Script

The above eval examples will invoke your application every time a shell is started. This may slow down shell startup time significantly.

Alternatively, export the generated completion code as a static script to be executed. You can ship this file with your builds; tools like Git do this. At least Zsh will also cache the results of completion files, but not eval scripts.

For Bash:

```
_CIRCUP_COMPLETE=bash_source circup circup-complete.sh
```

For Zsh:

```
_CIRCUP_COMPLETE=zsh_source circup circup-complete.sh
```

For Fish:

```
_CIRCUP_COMPLETE=fish_source circup circup-complete.sh
```

In .bashrc or .zshrc, source the script instead of the eval command:

```
. /path/to/circup-complete.sh
```

For Fish, add the file to the completions directory:

```
_CIRCUP_COMPLETE=fish_source circup ~/.config/fish/completions/circup-complete.fish
```

Note: If you find a bug, or you want to suggest an enhancement or new feature feel free to create an issue or submit a pull request here:

https://github.com/adafruit/circup

Discussion of this tool happens on the Adafruit CircuitPython Discord channel.

CHAPTER

SIX

CONTRIBUTING

Please note that this project is released with a Contributor Code of Conduct. By participating in this project you agree to abide by its terms. Participation covers any forum used to converse about CircuitPython including unofficial and official spaces. Failure to do so will result in corrective actions such as time out or ban from the project.

6.1 Licensing

By contributing to this repository you are certifying that you have all necessary permissions to license the code under an MIT License. You still retain the copyright but are granting many permissions under the MIT License.

If you have an employment contract with your employer please make sure that they don't automatically own your work product. Make sure to get any necessary approvals before contributing. Another term for this contribution off-hours is moonlighting.

6.2 Developer Setup

Note: Please try to use Python 3.6+ while developing CircUp. This is so we can use the Black code formatter (which only works with Python 3.6+).

Clone the repository and from the root of the project, install the requirements:

```
pip install -e ".[dev]"
```

Run the test suite:

```
pytest --random-order --cov-config .coveragerc --cov-report term-missing --cov=circup
```

6.2.1 How Does Circup Work?

The `circup` tool checks for a connected CircuitPython device by interrogating the local filesystem to find a path to a directory which ends with `"CIRCUITPYTHON"` (the name under which a CircuitPython device is mounted by the host operating system). This is handled in the `find_device` function.

A Python module on a connected device is represented by an instance of the `Module` class. This class provides useful methods for discerning if the module is out of date, returning useful representations of it in order to display information to the user, or updating the module on the connected device with whatever the version is in the latest Adafruit CircuitPython Bundle.

All of the libraries included in the Adafruit CircuitPython Bundle contain, somewhere within their code, two metadata objects called `__version__` and `__repo__`.

The `__repo__` object is a string containing the GitHub repository URL, as used to clone the project.

The `__version__` object is interesting because *within the source code in Git* the value is **always** the string `"0.0.0-auto.0"`. When a new release is made of the bundle, this value is automatically replaced by the build scripts to the correct version information, which will always conform to the semver standard.

Given this context, the `circup` tool will check a configuration file to discern what *it* thinks is the latest version of the bundle. If there is no configuration file (for example, on first run), then the bundle version is assumed to be `"0"`.

Next, it checks GitHub for the tag value (denoting the version) of the very latest bundle release. Bundle versions are based upon the date of release, for instance `"20190904"`. If the latest version on GitHub is later than the version `circup` currently has, then the latest version of the bundle is automatically downloaded and cached away somewhere.

In this way, the `circup` tool is able to have available to it both a path to a connected CIRCUITPYTHON devce and a copy of the latest version, including the all important version information, of the Adafruit CircuitPython Bundle.

Exactly the same function (`get_modules`) is used to extract the metadata from the modules on both the connected device and in the bundle cache. This metadata is used to instantiate instances of the `Module` class which is subsequently used to facilitate the various commands the tool makes available.

These commands are defined at the very end of the `circup.py` code.

Unit tests can be found in the `tests` directory. CircUp uses pytest style testing conventions. Test functions should include a comment to describe its *intention*. We currently have 100% unit test coverage for all the core functionality (excluding functions used to define the CLI commands).

To run the full test suite, type:

```
pytest --random-order --cov-config .coveragerc --cov-report term-missing --cov=circup
```

All code is formatted using the stylistic conventions enforced by black. Python coding standard are enforced by Pylint and verification of licensing is handled by REUSE. All of these are run using pre-commit, which you can run by using:

```
pip install pre-commit
pre-commit run --all-files
```

Please see the output from `pre-commit` for more information about the various available options to help you work with the code base.

Before submitting a PR, please remember to `pre-commit run --all-files`. But if you forget the CI process in Github will run it for you. ;-)

CircUp uses the Click module to run command-line interaction. The AppDirs module is used to determine where to store user-specific assets created by the tool in such a way that meets the host operating system's usual conventions. The python-semver package is used to validate and compare the semver values associated with modules. The ubiquitous requests module is used for HTTP activity.

Documentation, generated by Sphinx, is based on this README and assembled by assets in the `doc` subdirectory. The latest version of the docs will be found on Read the Docs.

Discussion of this tool happens on the Adafruit CircuitPython Discord channel.

CHAPTER

SEVEN

API

CircUp – a utility to manage and update libraries on a CircuitPython device.

circup.**BAD_FILE_FORMAT = 'Invalid'**
Version identifier for a bad MPY file format

circup.**BOARDLESS_COMMANDS = ['show', 'bundle-add', 'bundle-remove', 'bundle-show']**
Commands that do not require an attached board

circup.**BUNDLE_CONFIG_FILE = '/home/docs/checkouts/readthedocs.org/user_builds/adafruit-circ**
The path to the JSON file containing the metadata about the bundles.

circup.**BUNDLE_CONFIG_LOCAL = '/home/docs/.local/share/circup/bundle_config_local.json'**
The path to the JSON file containing the local list of bundles.

circup.**BUNDLE_CONFIG_OVERWRITE = '/home/docs/.local/share/circup/bundle_config.json'**
Overwrite the bundles list with this file (only done manually)

circup.**BUNDLE_DATA = '/home/docs/.local/share/circup/circup.json'**
The path to the JSON file containing the metadata about the bundles.

class circup.**Bundle**(*repo*)
All the links and file names for a bundle

Initialise a Bundle created from its github info. Construct all the strings in one place.

Parameters **repo** (*str*) – Repository string for github: "user/repository"

property current_tag
Lazy load current cached tag from the BUNDLE_DATA json file.

Returns The current cached tag value for the project.

property latest_tag
Lazy find the value of the latest tag for the bundle.

Returns The most recent tag value for the project.

lib_dir(*platform*)
This bundle's lib directory for the platform.

Parameters **platform** (*str*) – The platform identifier (py/6mpy/. . .).

Returns The path to the lib directory for the platform.

requirements_for(*library_name*)
The requirements file for this library.

Parameters **library_name** (*str*) – The name of the library.

Returns The path to the requirements.txt file.

validate()
Test the existence of the expected URLs (not their content)

circup.**CPY_VERSION = ''**
The version of CircuitPython found on the connected device.

circup.**DATA_DIR = '/home/docs/.local/share/circup'**
The location of data files used by circup (following OS conventions).

circup.**LOGFILE = '/home/docs/.cache/circup/log/circup.log'**
The location of the log file for the utility.

circup.**LOG_DIR = '/home/docs/.cache/circup/log'**
The directory containing the utility's log file.

class circup.**Module**(*path*, *repo*, *device_version*, *bundle_version*, *mpy*, *bundle*, *compatibility*)
Represents a CircuitPython module.

The `self.file` and `self.name` attributes are constructed from the `path` value. If the path is to a directory based module, the resulting self.file value will be None, and the name will be the basename of the directory path.

Parameters

- **path** (*str*) – The path to the module on the connected CIRCUITPYTHON device.
- **repo** (*str*) – The URL of the Git repository for this module.
- **device_version** (*str*) – The semver value for the version on device.
- **bundle_version** (*str*) – The semver value for the version in bundle.
- **mpy** (*bool*) – Flag to indicate if the module is byte-code compiled.
- **bundle** (Bundle) – Bundle object where the module is located.
- **compatibility** (*(str, str)*) – Min and max versions of CP compatible with the mpy.

property bad_format
A boolean indicating that the mpy file format could not be identified

property major_update
Returns a boolean to indicate if this is a major version update.

Returns Boolean indicating if this is a major version upgrade

property mpy_mismatch
Returns a boolean to indicate if this module's MPY version is compatible with the board's current version of Circuitpython. A min or max version that evals to False means no limit.

Returns Boolean indicating if the MPY versions don't match.

property outofdate
Returns a boolean to indicate if this module is out of date. Treat mismatched MPY versions as out of date.

Returns Truthy indication if the module is out of date.

property row
Returns a tuple of items to display in a table row to show the module's name, local version and remote version, and reason to update.

Returns A tuple containing the module's name, version on the connected device, version in the latest bundle and reason to update.

update()
Delete the module on the device, then copy the module from the bundle back onto the device.

The caller is expected to handle any exceptions raised.

circup.**NOT_MCU_LIBRARIES = ['', 'adafruit-blinka', 'adafruit-blinka-bleio', 'adafruit-blink**
The libraries (and blank lines) which don't go on devices

circup.**PLATFORMS = {'7mpy': '7.x-mpy', '8mpy': '7.x-mpy', 'py': 'py'}**
Module formats list (and the other form used in github files)

circup.**REQUESTS_TIMEOUT = 30**
Timeout for requests calls like get()

circup.**VERBOSE = False**
Flag to indicate if the command is being run in verbose mode.

circup.**clean_library_name**(*assumed_library_name*)
Most CP repos and library names are look like this:

repo: Adafruit_CircuitPython_LC709203F library: adafruit_lc709203f

But some do not and this handles cleaning that up. Also cleans up if the pypi or reponame is passed in instead of the CP library name.

Parameters **assumed_library_name** (*str*) – An assumed name of a library from user or requirements.txt entry

Returns str proper library name

circup.**completion_for_install**(*ctx*, *param*, *incomplete*)
Returns the list of available modules for the command line tab-completion with the `circup install` command.

circup.**ensure_latest_bundle**(*bundle*)
Ensure that there's a copy of the latest library bundle available so circup can check the metadata contained therein.

Parameters **bundle** (Bundle) – the target Bundle object.

circup.**extract_metadata**(*path*)
Given an file path, return a dictionary containing metadata extracted from dunder attributes found therein. Works with both .py and .mpy files.

For Python source files, such metadata assignments should be simple and single-line. For example:

```
__version__ = "1.1.4"
__repo__ = "https://github.com/adafruit/SomeLibrary.git"
```

For byte compiled .mpy files, a brute force / backtrack approach is used to find the __version__ number in the file – see comments in the code for the implementation details.

Parameters **path** (*str*) – The path to the file containing the metadata.

Returns The dunder based metadata found in the file, as a dictionary.

circup.**find_device**()
Return the location on the filesystem for the connected CircuitPython device. This is based upon how Mu discovers this information.

Returns The path to the device on the local filesystem.

`circup.`**`find_modules`**`(`*device_path*, *bundles_list*`)`

Extracts metadata from the connected device and available bundles and returns this as a list of Module instances representing the modules on the device.

Parameters

- **`device_path`** (*`str`*) – The path to the connected board.
- **`bundles_list`** (`Bundle`) – List of supported bundles as Bundle objects.

Returns A list of Module instances describing the current state of the modules on the connected device.

`circup.`**`get_bundle`**`(`*bundle*, *tag*`)`

Downloads and extracts the version of the bundle with the referenced tag. The resulting zip file is saved on the local filesystem.

Parameters

- **`bundle`** (`Bundle`) – the target Bundle object.
- **`tag`** (*`str`*) – The GIT tag to use to download the bundle.

`circup.`**`get_bundle_versions`**`(`*bundles_list*, *avoid_download=False*`)`

Returns a dictionary of metadata from modules in the latest known release of the library bundle. Uses the Python version (rather than the compiled version) of the library modules.

Parameters

- **`bundles_list`** (`Bundle`) – List of supported bundles as Bundle objects.
- **`avoid_download`** (*`bool`*) – if True, download the bundle only if missing.

Returns A dictionary of metadata about the modules available in the library bundle.

`circup.`**`get_bundles_dict`**`()`

Retrieve the dictionary from BUNDLE_CONFIG_FILE (JSON). Put the local dictionary in front, so it gets priority. It's a dictionary of bundle string identifiers.

Returns Combined dictionaries from the config files.

`circup.`**`get_bundles_list`**`()`

Retrieve the list of bundles from the config dictionary.

Returns List of supported bundles as Bundle objects.

`circup.`**`get_bundles_local_dict`**`()`

Retrieve the local bundles from BUNDLE_CONFIG_LOCAL (JSON).

Returns Raw dictionary from the config file(s).

`circup.`**`get_circuitpython_version`**`(`*device_path*`)`

Returns the version number of CircuitPython running on the board connected via `device_path`, along with the board ID. This is obtained from the `boot_out.txt` file on the device, whose first line will start with something like this:

```
Adafruit CircuitPython 4.1.0 on 2019-08-02;
```

While the second line is:

```
Board ID:raspberry_pi_pico
```

Parameters **`device_path`** (*`str`*) – The path to the connected board.

Returns A tuple with the version string for CircuitPython and the board ID string.

`circup.`**`get_circup_version`**`()`

Return the version of circup that is running. If not available, return None.

Returns Current version of circup, or None.

`circup.`**`get_dependencies`**`(`**requested_libraries*, *mod_names*, *to_install=()*`)`

Return a list of other CircuitPython libraries

Parameters

- **`requested_libraries`** (*`tuple`*) – The libraries to search for dependencies
- **`mod_names`** (*`object`*) – All the modules metadata from bundle
- **`to_install`** (*`list (str)`*) – Modules already selected for installation.

Returns tuple of module names to install which we build

`circup.`**`get_device_versions`**`(`*device_path*`)`

Returns a dictionary of metadata from modules on the connected device.

Parameters **`device_path`** (*`str`*) – Path to the device volume.

Returns A dictionary of metadata about the modules available on the connected device.

`circup.`**`get_latest_release_from_url`**`(`*url*`)`

Find the tag name of the latest release by using HTTP HEAD and decoding the redirect.

Parameters **`url`** (*`str`*) – URL to the latest release page on a git repository.

Returns The most recent tag value for the release.

`circup.`**`get_modules`**`(`*path*`)`

Get a dictionary containing metadata about all the Python modules found in the referenced path.

Parameters **`path`** (*`str`*) – The directory in which to find modules.

Returns A dictionary containing metadata about the found modules.

`circup.`**`install_module`**`(`*device_path*, *device_modules*, *name*, *pyext*, *mod_names*`)`

Finds a connected device and installs a given module name if it is available in the current module bundle and is not already installed on the device. TODO: There is currently no check for the version.

Parameters

- **`device_path`** (*`str`*) – The path to the connected board.
- **`device_modules`** (*`list (dict)`*) – List of module metadata from the device.
- **`name`** (*`str`*) – Name of module to install
- **`pyext`** (*`bool`*) – Boolean to specify if the module should be installed from source or from a pre-compiled module
- **`mod_names`** – Dictionary of metadata from modules that can be generated with get_bundle_versions()

`circup.`**`libraries_from_imports`**`(`*code_py*, *mod_names*`)`

Parse the given code.py file and return the imported libraries

Parameters **`code_py`** (*`str`*) – Full path of the code.py file

Returns sequence of library names

`circup.`**`libraries_from_requirements`**`(`*requirements*`)`

Clean up supplied requirements.txt and turn into tuple of CP libraries

Parameters **`requirements`** (*`str`*) – A string version of a requirements.txt

Returns tuple of library names

`circup.`**`save_local_bundles`**`(`*bundles_data*`)`

Save the list of local bundles to the settings.

Parameters **`key`** (*`str`*) – The bundle's identifier/key.

`circup.`**`tags_data_load`**`()`

Load the list of the version tags of the bundles on disk.

Returns a dict() of tags indexed by Bundle identifiers/keys.

`circup.`**`tags_data_save_tag`**`(`*key*, *tag*`)`

Add or change the saved tag value for a bundle.

Parameters

- **`key`** (*`str`*) – The bundle's identifier/key.
- **`tag`** (*`str`*) – The new tag for the bundle.

CHAPTER EIGHT

LICENSE

PYTHON MODULE INDEX

c

circup, 17

INDEX

B

BAD_FILE_FORMAT (*in module circup*), 17
bad_format() (*circup.Module property*), 18
BOARDLESS_COMMANDS (*in module circup*), 17
Bundle (*class in circup*), 17
BUNDLE_CONFIG_FILE (*in module circup*), 17
BUNDLE_CONFIG_LOCAL (*in module circup*), 17
BUNDLE_CONFIG_OVERWRITE (*in module circup*), 17
BUNDLE_DATA (*in module circup*), 17

C

circup (*module*), 17
clean_library_name() (*in module circup*), 19
completion_for_install() (*in module circup*), 19
CPY_VERSION (*in module circup*), 18
current_tag() (*circup.Bundle property*), 17

D

DATA_DIR (*in module circup*), 18

E

ensure_latest_bundle() (*in module circup*), 19
extract_metadata() (*in module circup*), 19

F

find_device() (*in module circup*), 19
find_modules() (*in module circup*), 19

G

get_bundle() (*in module circup*), 20
get_bundle_versions() (*in module circup*), 20
get_bundles_dict() (*in module circup*), 20
get_bundles_list() (*in module circup*), 20
get_bundles_local_dict() (*in module circup*), 20
get_circuitpython_version() (*in module circup*), 20
get_circup_version() (*in module circup*), 21
get_dependencies() (*in module circup*), 21
get_device_versions() (*in module circup*), 21
get_latest_release_from_url() (*in module circup*), 21
get_modules() (*in module circup*), 21

I

install_module() (*in module circup*), 21

L

latest_tag() (*circup.Bundle property*), 17
lib_dir() (*circup.Bundle method*), 17
libraries_from_imports() (*in module circup*), 21
libraries_from_requirements() (*in module circup*), 21
LOG_DIR (*in module circup*), 18
LOGFILE (*in module circup*), 18

M

major_update() (*circup.Module property*), 18
Module (*class in circup*), 18
mpy_mismatch() (*circup.Module property*), 18

N

NOT_MCU_LIBRARIES (*in module circup*), 19

O

outofdate() (*circup.Module property*), 18

P

PLATFORMS (*in module circup*), 19

R

REQUESTS_TIMEOUT (*in module circup*), 19
requirements_for() (*circup.Bundle method*), 17
row() (*circup.Module property*), 18

S

save_local_bundles() (*in module circup*), 22

T

tags_data_load() (*in module circup*), 22 `tags_data_save_tag()` (*in module circup*), 22

U

`update()` (*circup.Module method*), 18

V

`validate()` (*circup.Bundle method*), 17
`VERBOSE` (*in module circup*), 19