
github3-utils

Release 0.7.1

Handy utilities for github3.py

Dominic Davis-Foster

May 28, 2023

Contents

1	Installation	1
1.1	from PyPI	1
1.2	from Anaconda	1
1.3	from GitHub	1
I	API Reference	3
2	github3_utils	5
2.1	Impersonate	5
2.2	RateLimitExceeded	6
2.3	echo_rate_limit	6
2.4	get_repos	7
2.5	get_user	7
2.6	iter_repos	7
2.7	protect_branch	7
3	github3_utils.apps	9
3.1	ContextSwitcher	9
3.2	iter_installed_repos	10
3.3	make_footer_links	11
4	github3_utils.check_labels	13
4.1	Checks	13
4.2	Label	13
4.3	check_status_labels	15
4.4	get_checks_for_pr	15
4.5	label_pr_failures	15
5	github3_utils.click	17
5.1	token_option	17
6	github3_utils.secrets	19
6.1	PublicKey	19
6.2	build_secrets_url	19
6.3	encrypt_secret	20
6.4	get_public_key	20
6.5	get_secrets	20
6.6	set_secret	20
7	github3_utils.testing	21
7.1	cassette	21

7.2	github_client	21
7.3	module_cassette	22
II	About	23
8	Contributing	25
8.1	Coding style	25
8.2	Automated tests	25
8.3	Type Annotations	25
8.4	Build documentation locally	26
9	Downloading source code	27
9.1	Building from source	28
10	License	29
	Python Module Index	31
	Index	33

Installation

1.1 from PyPI

```
$ python3 -m pip install github3-utils --user
```

1.2 from Anaconda

First add the required channels

```
$ conda config --add channels https://conda.anaconda.org/conda-forge  
$ conda config --add channels https://conda.anaconda.org/domdfcoding
```

Then install

```
$ conda install github3-utils
```

1.3 from GitHub

```
$ python3 -m pip install git+https://github.com/domdfcoding/github3-utils@master --user
```


Part I

API Reference

github3_utils

Handy utilities for `github3.py`.

Classes:

<code>Impersonate(name, email)</code>	Context manager to make commits as a specific user.
---------------------------------------	---

Exceptions:

<code>RateLimitExceeded(reset_time)</code>	Custom exception class to indicate the GitHub rate limit has been exceeded and no further requests should be made.
--	--

Functions:

<code>echo_rate_limit(github[, verbose])</code>	Contextmanager to echo the GitHub API rate limit before and after making a series of requests.
<code>get_repos(user_or_org[, full])</code>	Returns an iterator over the user or organisation's repositories.
<code>get_user(github)</code>	Retrieve a <code>github3.users.User</code> object for the authenticated user.
<code>iter_repos(github[, users, orgs])</code>	Returns an iterator over the repositories belonging to all <code>users</code> and all <code>orgs</code> .
<code>protect_branch(branch[, status_checks])</code>	Enable force push protection and configure status check enforcement.

class `Impersonate` (*name, email*)

Bases: `object`

Context manager to make commits as a specific user.

Sets the following environment variables:

- `GIT_COMMITTER_NAME`
- `GIT_COMMITTER_EMAIL`
- `GIT_AUTHOR_NAME`
- `GIT_AUTHOR_EMAIL`

Attention: Any changes to environment variables made during the scope of the context manager will be reset on exit.

Example:

```
name = "repo-helper[bot]"
email = f"74742576+{name}@users.noreply.github.com"

commit_as_bot = Impersonate(name=name, email=email)

with commit_as_bot():
    ...
```

Parameters

- **name** (`str`) – The name of the committer.
- **email** (`str`) – The email address of the committer.

Attributes:

<code>email</code>	The email address of the committer.
<code>name</code>	The name of the committer.

email**Type:** `str`

The email address of the committer.

name**Type:** `str`

The name of the committer.

exception RateLimitExceeded (`reset_time`)Bases: `RuntimeError`

Custom exception class to indicate the GitHub rate limit has been exceeded and no further requests should be made.

reset_time**Type:** `datetime`

The time at which the rate limit will be reset.

echo_rate_limit (`github`, `verbose=True`)

Contextmanager to echo the GitHub API rate limit before and after making a series of requests.

Parameters

- **github** (`GitHub`)
- **verbose** (`bool`) – If `False` no output will be printed. Default `True`.

Raises `click.Abort` if the rate limit has been exceeded.**Return type** `Iterator[GitHub]`

get_repos (*user_or_org*, *full=False*)

Returns an iterator over the user or organisation's repositories.

New in version 0.5.0.

Parameters

- **user_or_org** (`Union[User, Organization]`)
- **full** (`bool`) – If `True` yields `Repository` objects. Otherwise, yields `ShortRepository` objects. Default `False`.

Return type `Union[Iterator[Repository], Iterator[ShortRepository]]`

Overloads

- `get_repos(user_or_org, full: Literal[True]) -> Iterator[Repository]`
- `get_repos(user_or_org, full: Literal[False] = ...) -> Iterator[ShortRepository]`

get_user (*github*)

Retrieve a `github3.users.User` object for the authenticated user.

Parameters **github** (`GitHub`)

Return type `User`

iter_repos (*github*, *users=()*, *orgs=()*)

Returns an iterator over the repositories belonging to all users and all orgs.

New in version 0.5.0.

Parameters

- **github** (`GitHub`)
- **users** (`Iterable[str]`) – An iterable of usernames to fetch the repositories for. Default `()`.
- **orgs** (`Iterable[str]`) – An iterable of organization names to fetch the repositories for. Default `()`.

Return type `Iterator[ShortRepository]`

protect_branch (*branch*, *status_checks=None*)

Enable force push protection and configure status check enforcement.

Parameters

- **branch** (`Branch`) – The branch to enable protection for.
- **status_checks** (`Optional[List[str]]`) – A list of strings naming status checks which must pass before merging. Use `None` or omit to use the already associated value. Default `None`.

Return type `bool`

Returns `True` if successful, `False` otherwise.

github3_utils.apps

Functions and classes for GitHub apps.

Classes:

<i>ContextSwitcher</i> (client, private_key_pem, app_id)	Class to aid switching contexts between the app itself and its installations.
--	---

Functions:

<i>iter_installed_repos</i> (*[, context_switcher, ...])	Returns an iterator over all repositories the app is installed for.
<i>make_footer_links</i> (owner, name[, event_date, ...])	Create markdown footer links for a GitHub app.

class ContextSwitcher (*client, private_key_pem, app_id*)

Bases: `object`

Class to aid switching contexts between the app itself and its installations.

Parameters

- **client** (`GitHub`)
- **private_key_pem** (`bytes`) – The bytes of the private key for this GitHub App.
- **app_id** (`int`) – The integer identifier for this GitHub App.

Attributes:

<i>app_id</i>	The integer identifier for this GitHub App.
<i>client</i>	
<i>private_key_pem</i>	The bytes of the private key for this GitHub App.

Methods:

<i>login_as_app</i> ()	Login as the GitHub app.
<i>login_as_org_installation</i> (organization)	Login as an organization installation of a GitHub app, and return its installation ID.
<i>login_as_repo_installation</i> (owner, repository)	Login as a repository installation of a GitHub app, and return its installation ID.
<i>login_as_user_installation</i> (username)	Login as a user installation of a GitHub app, and return its installation ID.

app_id

Type: `int`

The integer identifier for this GitHub App.

client

Type: `GitHub`

login_as_app()

Login as the GitHub app.

login_as_org_installation(organization)

Login as an organization installation of a GitHub app, and return its installation ID.

New in version 0.5.0.

Parameters `organization(str)`

Return type `int`

login_as_repo_installation(owner, repository)

Login as a repository installation of a GitHub app, and return its installation ID.

Parameters

- **owner** (`str`)
- **repository** (`str`)

Return type `int`

login_as_user_installation(username)

Login as a user installation of a GitHub app, and return its installation ID.

Parameters `username(str)`

Return type `int`

private_key_pem

Type: `bytes`

The bytes of the private key for this GitHub App.

iter_installed_repos (*, *context_switcher=None*, *client=None*, *private_key_pem=None*,
app_id=None)

Returns an iterator over all repositories the app is installed for.

Parameters

- **context_switcher** (`Optional[ContextSwitcher]`) – A `ContextSwitcher` used to switch contexts between the app itself and its installations. Default `None`.
- **client** (`Optional[GitHub]`) – The bytes of the private key for this GitHub App. Default `None`.
- **private_key_pem** (`Optional[bytes]`) – The bytes of the private key for this GitHub App. Default `None`.
- **app_id** (`Optional[int]`) – The integer identifier for this GitHub App. Default `None`.

Either `context_switcher` or all of `client`, `private_key_pem` and `app_id` must be provided.

Return type `Iterator[Dict]`

make_footer_links (*owner*, *name*, *event_date*=None, *type*='marketplace', *docs_url*=None)

Create markdown footer links for a GitHub app.

Parameters

- **owner** (`str`) – The owner of the repository.
- **name** (`str`) – The name of the repository.
- **event_date** (`Optional[date]`) – The date on which the footer is being created. Determines the emoji shown. Default `None`.
- **type** (`Literal['marketplace', 'app']`) – Whether the footer is for a GitHub app or an item in the marketplace. Default `'marketplace'`.
- **docs_url** (`Optional[str]`) – The URL of the app's documentation. If `None` no link will be shown. Default `None`.

New in version 0.3.0.

Return type `str`

github3_utils.check_labels

Helpers for creating labels to mark pull requests with which tests are failing.

New in version 0.4.0.

Classes:

<code>Checks(successful, failing, running, ...)</code>	Represents the sets of status checks returned by <code>get_checks_for_pr()</code> .
<code>Label(name, color[, description])</code>	Represents an issue or pull request label.

Data:

<code>check_status_labels</code>	Labels corresponding to failing pull request checks.
----------------------------------	--

Functions:

<code>get_checks_for_pr(pull)</code>	Returns a <code>Checks</code> object containing sets of check names grouped by their status.
<code>label_pr_failures(pull)</code>	Labels the given pull request to indicate which checks are failing.

namedtuple Checks (*successful, failing, running, skipped, neutral*)

Bases: `NamedTuple`

Represents the sets of status checks returned by `get_checks_for_pr()`.

Fields

- 0) **successful** (`Set[str]`) – Alias for field number 0
- 1) **failing** (`Set[str]`) – Alias for field number 1
- 2) **running** (`Set[str]`) – Alias for field number 2
- 3) **skipped** (`Set[str]`) – Alias for field number 3
- 4) **neutral** (`Set[str]`) – Alias for field number 4

`__repr__()`

Return a nicely formatted representation string

class Label (*name, color, description=None*)

Bases: `object`

Represents an issue or pull request label.

Parameters

- **name** (`str`) – The text of the label.

- **color** (*str*) – The background colour of the label.
- **description** (*Optional[str]*) – A short description of the label. Default *None*.

Methods:

<code>__repr__()</code>	Return a string representation of the <i>Label</i> .
<code>__str__()</code>	Return <code>str(self)</code> .
<code>create(repo)</code>	Create this label on the given repository.
<code>to_dict()</code>	Return the <i>Label</i> as a dictionary.

Attributes:

<i>color</i>	The background colour of the label.
<i>description</i>	A short description of the label.
<i>name</i>	The text of the label.

`__repr__()`
Return a string representation of the *Label*.

Return type *str*

`__str__()`
Return `str(self)`.

Return type *str*

color
Type: *str*
The background colour of the label.

create (*repo*)
Create this label on the given repository.

Parameters **repo** (*Repository*)

Return type *Label*

description
Type: *str*
A short description of the label.

name
Type: *str*
The text of the label.

to_dict ()
Return the *Label* as a dictionary.
Return type *Dict[str, str]*

```
check_status_labels = {'failure: Linux': Label(name='failure: Linux', color='#F6BDC0', color_name='failure: Linux')
    Type: Dict[str, Label]
```

Labels corresponding to failing pull request checks.

get_checks_for_pr (*pull*)

Returns a *Checks* object containing sets of check names grouped by their status.

Parameters *pull* (*Union*[*PullRequest*, *ShortPullRequest*]) – The pull request to obtain checks for.

Return type *Checks*

label_pr_failures (*pull*)

Labels the given pull request to indicate which checks are failing.

Parameters *pull* (*Union*[*PullRequest*, *ShortPullRequest*])

Return type *Set*[*str*]

Returns The new labels set for the pull request.

github3_utils.click

Extensions for `click`.

New in version 0.2.0.

Functions:

<code>token_option([token_var])</code>	Creates a <code>-t / --token</code> option for the GitHub API token.
--	--

token_option (*token_var*='GITHUB_TOKEN')

Creates a `-t / --token` option for the GitHub API token.

New in version 0.2.0.

Parameters `token_var` (*str*) – Default 'GITHUB_TOKEN'.

Return type `Callable[[click.Command], click.Command]`

github3_utils.secrets

Functions for setting and updating GitHub Actions secrets.

Classes:

<i>PublicKey</i>	<code>typing.TypedDict</code> representing the return type of <code>get_public_key()</code> .
------------------	---

Functions:

<i>build_secrets_url(repo)</i>	Returns the URL via which secrets can be checked and set.
<i>encrypt_secret(public_key, secret_value)</i>	Encrypt a GitHub Actions secret.
<i>get_public_key(repo)</i>	Returns the public key used to encrypt secrets for the given repository.
<i>get_secrets(repo)</i>	Returns a list of secret names for the given repository.
<i>set_secret(repo, secret_name, value, public_key)</i>	Set the value of the given secret.

typeddict PublicKey

`typing.TypedDict` representing the return type of `get_public_key()`.

Required Keys

- **key** (`str`)
- **key_id** (`str`)

Optional Keys

- **ETag** (`str`)
- **Last-Modified** (`str`)

build_secrets_url(repo)

Returns the URL via which secrets can be checked and set.

Parameters `repo` (`Repository`) – The repository to check/set secrets for.

Return type `URL`

encrypt_secret (*public_key*, *secret_value*)

Encrypt a GitHub Actions secret.

Parameters

- **public_key** (*str*)
- **secret_value** (*str*)

If the key has been obtained with `get_secrets()` then `public_key` will be:

```
get_secrets(repo) ['key']
```

Return type *str*

get_public_key (*repo*)

Returns the public key used to encrypt secrets for the given repository.

Parameters **repo** (*Repository*) – The repository the secrets are to be set for.

Return type *PublicKey*

get_secrets (*repo*)

Returns a list of secret names for the given repository.

Parameters **repo** (*Repository*)

Return type *List[str]*

set_secret (*repo*, *secret_name*, *value*, *public_key*)

Set the value of the given secret.

Parameters

- **repo** (*Repository*)
- **secret_name** (*str*)
- **value** (*str*)
- **public_key** (*PublicKey*)

Return type *Response*

github3_utils.testing

Fixtures for `pytest`.

Attention: This module has the following additional requirements:

```
betamax>=0.8.1
pytest>=6.0.0
```

These can be installed as follows:

```
$ python -m pip install github3-utils[testing]
```

New in version 0.2.0.

To use this module you need to add, at a minimum, the following to your `conftest.py`:

```
from betamax import Betamax

pytest_plugins = ("github3_utils.testing", )

with Betamax.configure() as config:
    config.cassette_library_dir = "<path to cassettes directory>"
```

Functions:

<code>cassette(request, github_client)</code>	Provides a Betamax cassette scoped to the test function which record and plays back interactions with the GitHub API.
<code>github_client()</code>	Provides an instance of <code>github3.github.GitHub</code> , using a fake token to authenticate.
<code>module_cassette(request, github_client)</code>	Provides a Betamax cassette scoped to the test module which record and plays back interactions with the GitHub API.

fixture `cassette`

Scope: function

Provides a Betamax cassette scoped to the test function which record and plays back interactions with the GitHub API.

Return type `Iterator[GitHub]`

fixture `github_client`

Scope: function

Provides an instance of `github3.github.GitHub`, using a fake token to authenticate.

Return type `GitHub`

fixture module_cassette

Scope: function

Provides a Betamax cassette scoped to the test module which record and plays back interactions with the GitHub API.

Return type `Iterator[GitHub]`

Part II

About

Contributing

github3-utils uses `tox` to automate testing and packaging, and `pre-commit` to maintain code quality.

Install `pre-commit` with `pip` and install the git hook:

```
$ python -m pip install pre-commit
$ pre-commit install
```

8.1 Coding style

`formate` is used for code formatting.

It can be run manually via `pre-commit`:

```
$ pre-commit run formate -a
```

Or, to run the complete autoformatting suite:

```
$ pre-commit run -a
```

8.2 Automated tests

Tests are run with `tox` and `pytest`. To run tests for a specific Python version, such as Python 3.6:

```
$ tox -e py36
```

To run tests for all Python versions, simply run:

```
$ tox
```

8.3 Type Annotations

Type annotations are checked using `mypy`. Run `mypy` using `tox`:

```
$ tox -e mypy
```

8.4 Build documentation locally

The documentation is powered by Sphinx. A local copy of the documentation can be built with `tox`:

```
$ tox -e docs
```

Downloading source code

The `github3-utils` source code is available on GitHub, and can be accessed from the following URL: <https://github.com/domdfcoding/github3-utils>

If you have `git` installed, you can clone the repository with the following command:

```
$ git clone https://github.com/domdfcoding/github3-utils
```

```
Cloning into 'github3-utils'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a ‘zip’ file by clicking:

Clone or download → Download Zip

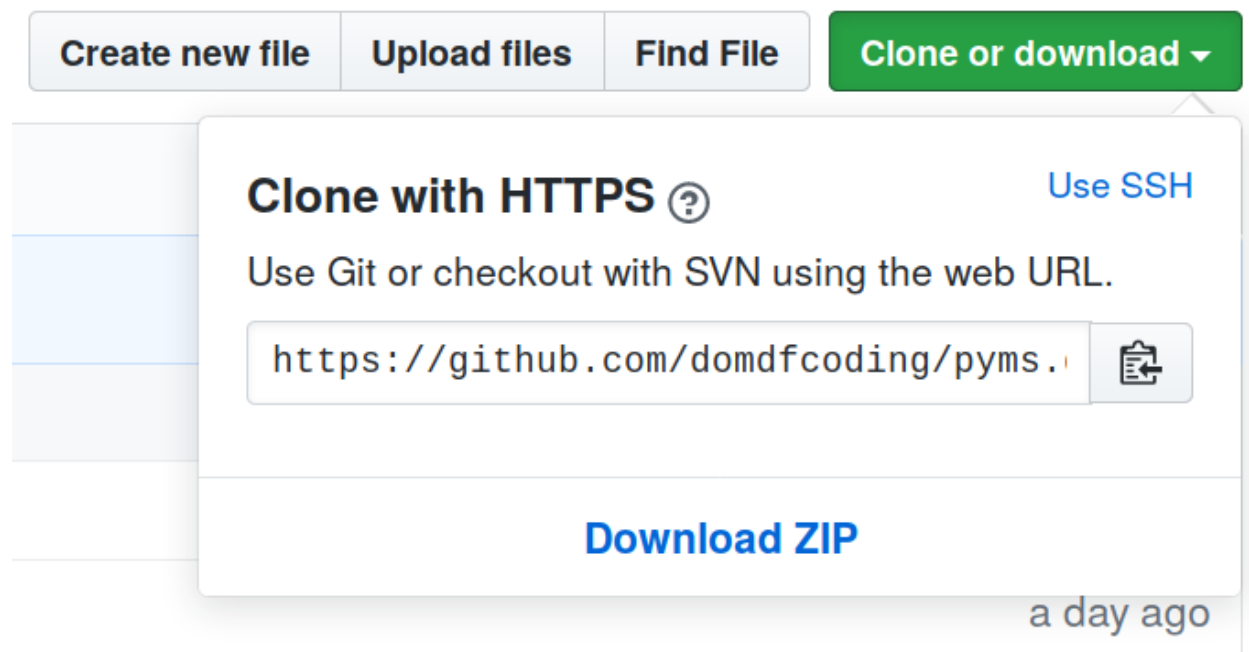


Fig. 1: Downloading a ‘zip’ file of the source code

9.1 Building from source

The recommended way to build `github3-utils` is to use `tox`:

```
$ tox -e build
```

The source and wheel distributions will be in the directory `dist`.

If you wish, you may also use `pep517.build` or another **PEP 517**-compatible build tool.

License

github3-utils is licensed under the [MIT License](#)

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- Commercial use – The licensed material and derivatives may be used for commercial purposes.
- Modification – The licensed material may be modified.
- Distribution – The licensed material may be distributed.
- Private use – The licensed material may be used and modified in private.

Conditions

- License and copyright notice – A copy of the license and copyright notice must be included with the licensed material.

Limitations

- Liability – This license includes a limitation of liability.
- Warranty – This license explicitly states that it does NOT provide any warranty.

[See more information on choosealicense.com](#) ⇒

```
Copyright (c) 2020-2022 Dominic Davis-Foster
```

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

```
The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE
OR OTHER DEALINGS IN THE SOFTWARE.
```


Python Module Index

g

- `github3_utils`, [5](#)
- `github3_utils.apps`, [9](#)
- `github3_utils.check_labels`, [13](#)
- `github3_utils.click`, [17](#)
- `github3_utils.secrets`, [19](#)
- `github3_utils.testing`, [21](#)

Symbols

`__repr__()` (*Checks method*), 13

`__repr__()` (*Label method*), 14

`__str__()` (*Label method*), 14

A

`app_id` (*ContextSwitcher attribute*), 9

B

`build_secrets_url()` (*in module `github3_utils.secrets`*), 19

C

`check_status_labels` (*in module `github3_utils.check_labels`*), 14

`Checks` (*namedtuple in `github3_utils.check_labels`*), 13

failing (*namedtuple field*), 13

neutral (*namedtuple field*), 13

running (*namedtuple field*), 13

skipped (*namedtuple field*), 13

successful (*namedtuple field*), 13

`client` (*ContextSwitcher attribute*), 10

`color` (*Label attribute*), 14

`ContextSwitcher` (*class in `github3_utils.apps`*), 9

`create()` (*Label method*), 14

D

`description` (*Label attribute*), 14

E

`echo_rate_limit()` (*in module `github3_utils`*), 6

`email` (*Impersonate attribute*), 6

`encrypt_secret()` (*in module `github3_utils.secrets`*), 20

F

`failing` (*namedtuple field*)
 Checks (*namedtuple in `github3_utils.check_labels`*), 13

G

`get_checks_for_pr()` (*in module `github3_utils.check_labels`*), 15

`get_public_key()` (*in module `github3_utils.secrets`*), 20

`get_repos()` (*in module `github3_utils`*), 6

`get_secrets()` (*in module `github3_utils.secrets`*), 20

`get_user()` (*in module `github3_utils`*), 7

`github3_utils`

module, 5

`github3_utils.apps`

module, 9

`github3_utils.check_labels`

module, 13

`github3_utils.click`

module, 17

`github3_utils.secrets`

module, 19

`github3_utils.testing`

module, 21

I

`Impersonate` (*class in `github3_utils`*), 5

`iter_installed_repos()` (*in module `github3_utils.apps`*), 10

`iter_repos()` (*in module `github3_utils`*), 7

L

`Label` (*class in `github3_utils.check_labels`*), 13

`label_pr_failures()` (*in module `github3_utils.check_labels`*), 15

`login_as_app()` (*ContextSwitcher method*), 10

`login_as_org_installation()`
 (*ContextSwitcher method*), 10

`login_as_repo_installation()`
 (*ContextSwitcher method*), 10

`login_as_user_installation()`
 (*ContextSwitcher method*), 10

M

`make_footer_links()` (*in module `github3_utils.apps`*), 11

MIT License, 29

`module`

github3_utils, 5

github3_utils.apps, 9

github3_utils.check_labels, 13

`github3_utils.click`, 17
`github3_utils.secrets`, 19
`github3_utils.testing`, 21

N

`name` (*Impersonate attribute*), 6
`name` (*Label attribute*), 14
`neutral` (*namedtuple field*)
 Checks (*namedtuple in*
 github3_utils.check_labels), 13

P

`private_key_pem` (*ContextSwitcher attribute*), 10
`protect_branch()` (*in module github3_utils*), 7
`PublicKey` (*typeddict in github3_utils.secrets*), 19
Python Enhancement Proposals
 PEP 517, 28

R

`RateLimitExceeded`, 6
`reset_time` (*RateLimitExceeded attribute*), 6
`running` (*namedtuple field*)
 Checks (*namedtuple in*
 github3_utils.check_labels), 13

S

`set_secret()` (*in module github3_utils.secrets*), 20
`skipped` (*namedtuple field*)
 Checks (*namedtuple in*
 github3_utils.check_labels), 13
`successful` (*namedtuple field*)
 Checks (*namedtuple in*
 github3_utils.check_labels), 13

T

`to_dict()` (*Label method*), 14
`token_option()` (*in module github3_utils.click*), 17