confusable$_{h}omoglyphsDocumentation$

Release 3.2.0+21.ga0b9aa3.dirty

Victor Felder

Jan 31, 2024

CONTENTS

1 confusable_homoglyphs [doc] **3**
- 1.1 API documentation . . . 3
- 1.2 Is the data up to date? . . . 3

2 Installation **5**

3 Usage **7**

4 API Documentation **9**
- 4.1 confusable_homoglyphs package . . . 9

5 Contributing **15**
- 5.1 Types of Contributions . . . 15
- 5.2 Get Started! . . . 16
- 5.3 Pull Request Guidelines . . . 17

6 Credits **19**
- 6.1 Original Author and Former Maintainer . . . 19
- 6.2 Current Maintainer . . . 19
- 6.3 Contributors . . . 19

7 History **21**
- 7.1 1.0.0 . . . 21
- 7.2 2.0.0 . . . 21
- 7.3 2.0.1 . . . 21
- 7.4 3.0.0 . . . 21
- 7.5 3.1.0 . . . 22
- 7.6 3.1.1 . . . 22
- 7.7 3.2.0 . . . 22
- 7.8 3.3.0 . . . 22
- 7.9 3.3.1 . . . 22

Index **23**

Contents:

CHAPTER

ONE

CONFUSABLE_HOMOGLYPHS [DOC]

This project has been adopted from the original confusable_homoglyphs by Victor Felder.

a homoglyph is one of two or more graphemes, characters, or glyphs with shapes that appear identical or very similar wikipedia:Homoglyph

Unicode homoglyphs can be a nuisance on the web. Your most popular client, AlaskaJazz, might be upset to be impersonated by a trickster who deliberately chose the username laskaJazz.

- `AlaskaJazz` is single script: only Latin characters.
- `laskaJazz` is mixed-script: the first character is a greek letter.

You might also want to avoid people being tricked into entering their password on `www.microsft.com` or `www.faebook.com` instead of `www.microsoft.com` or `www.facebook.com`. Here is a utility to play with these **confusable homoglyphs**.

Not all mixed-script strings have to be ruled out though, you could only exclude mixed-script strings containing characters that might be confused with a character from some unicode blocks of your choosing.

- `Allo` and are fine: single script.
- `Allo` is fine when our preferred script alias is 'latin': mixed script, but is not confusable.
- `Allo` is dangerous: mixed script and could be confused with `p`.

This library is compatible with Python 3.

1.1 API documentation

1.2 Is the data up to date?

Yep.

The unicode blocks aliases and names for each character are extracted from this file provided by the unicode consortium.

The matrix of which character can be confused with which other characters is built using this file provided by the unicode consortium.

This data is stored in two JSON files: `categories.json` and `confusables.json`. If you delete them, they will both be recreated by downloading and parsing the two abovementioned files and stored as JSON files again.

CHAPTER

TWO

INSTALLATION

If available, install an appropriate package from your distribution:

Otherwise you can install from PyPi:

at the command line:

```
$ easy_install confusable_homoglyphs
```

or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv confusable_homoglyphs
$ pip install confusable_homoglyphs
```

CHAPTER

THREE

USAGE

To use confusable_homoglyphs in a project:

```
pip install confusable_homoglyphs
import confusable_homoglyphs
```

To update the data files, you first need to install the "cli" bundle, then run the "update" command:

```
pip install confusable_homoglyphs[cli]
confusable_homoglyphs update
```

CHAPTER

FOUR

API DOCUMENTATION

4.1 confusable_homoglyphs package

4.1.1 Submodules

4.1.2 confusable_homoglyphs.categories module

`confusable_homoglyphs.categories.`**`alias`**(*chr*)

Retrieves the script block alias for a unicode character.

```
>>> categories.alias('A')
'LATIN'
>>> categories.alias('')
'GREEK'
>>> categories.alias('-')
'COMMON'
```

Parameters

chr (*str*) – A unicode character

Returns

The script block alias.

Return type

str

`confusable_homoglyphs.categories.`**`aliases_categories`**(*chr*)

Retrieves the script block alias and unicode category for a unicode character.

```
>>> categories.aliases_categories('A')
('LATIN', 'L')
>>> categories.aliases_categories('')
('GREEK', 'L')
>>> categories.aliases_categories('-')
('COMMON', 'Pd')
```

Parameters

chr (*str*) – A unicode character

Returns

The script block alias and unicode category for a unicode character.

Return type
(str, str)

`confusable_homoglyphs.categories.category(chr)`

Retrieves the unicode category for a unicode character.

```
>>> categories.category('A')
'L'
>>> categories.category('')
'L'
>>> categories.category('-')
'Pd'
```

Parameters
chr (*str*) – A unicode character

Returns
The unicode category for a unicode character.

Return type
str

`confusable_homoglyphs.categories.unique_aliases(string)`

Retrieves all unique script block aliases used in a unicode string.

```
>>> categories.unique_aliases('ABC')
{'LATIN'}
>>> categories.unique_aliases('A-')
{'GREEK', 'LATIN', 'COMMON'}
```

Parameters
string (*str*) – A unicode character

Returns
A set of the script block aliases used in a unicode string.

Return type
(str, str)

4.1.3 confusable_homoglyphs.cli module

4.1.4 confusable_homoglyphs.confusables module

exception `confusable_homoglyphs.confusables.Found`

Bases: `Exception`

`confusable_homoglyphs.confusables.is_confusable(string, greedy=False, preferred_aliases=[])`

Checks if `string` contains characters which might be confusable with characters from `preferred_aliases`.

If `greedy=False`, it will only return the first confusable character found without looking at the rest of the string, `greedy=True` returns all of them.

`preferred_aliases=[]` can take an array of unicode block aliases to be considered as your 'base' unicode blocks:

- considering paa,
  - with `preferred_aliases=['latin']`, the 3rd character would be returned because this greek letter can be confused with latin `p`.
  - with `preferred_aliases=['greek']`, the 1st character `p` would be returned because this latin letter can be confused with greek .
  - with `preferred_aliases=[]` and `greedy=True`, you'll discover the 29 characters that can be confused with `p`, the 23 characters that look like `a`, and the one that looks like (which is, of course, *p* aka *LATIN SMALL LETTER P*).

```
>>> confusables.is_confusable('paa', preferred_aliases=['latin'])[0]['character']
''
>>> confusables.is_confusable('paa', preferred_aliases=['greek'])[0]['character']
'p'
>>> confusables.is_confusable('Abç', preferred_aliases=['latin'])
False
>>> confusables.is_confusable('Allo', preferred_aliases=['latin'])
False
>>> confusables.is_confusable('', preferred_aliases=['greek'])
False
>>> confusables.is_confusable('.', preferred_aliases=['greek', 'common'])
False
>>> confusables.is_confusable('p')
[{'homoglyphs': [{'c': 'p', 'n': 'LATIN SMALL LETTER P'}], 'alias': 'GREEK',
↪'character': ''}]
```

Parameters

- **string** (*str*) – A unicode string
- **greedy** (*bool*) – Don't stop on finding one confusable character - find all of them.
- **preferred_aliases** (*list(str)*) – Script blocks aliases which we don't want `string`'s characters to be confused with.

Returns
False if not confusable, all confusable characters and with what they are confusable otherwise.

Return type
bool or list

`confusable_homoglyphs.confusables.`**`is_dangerous`**(*string*, *preferred_aliases=[]*)

Checks if `string` can be dangerous, i.e. is it not only mixed-scripts but also contains characters from other scripts than the ones in `preferred_aliases` that might be confusable with characters from scripts in `preferred_aliases`

For `preferred_aliases` examples, see `is_confusable` docstring.

```
>>> bool(confusables.is_dangerous('Allo'))
False
>>> bool(confusables.is_dangerous('Allo', preferred_aliases=['latin']))
False
>>> bool(confusables.is_dangerous('Allo'))
True
>>> bool(confusables.is_dangerous('AlaskaJazz'))
```

(continues on next page)

(continued from previous page)

```
False
>>> bool(confusables.is_dangerous('laskaJazz'))
True
```

Parameters

- **string** (*str*) – A unicode string
- **preferred_aliases** (*list(str)*) – Script blocks aliases which we don't want `string`'s characters to be confused with.

Returns
Is it dangerous.

Return type
bool

`confusable_homoglyphs.confusables.`**`is_mixed_script`**(*string*, *allowed_aliases=['COMMON']*)

Checks if `string` contains mixed-scripts content, excluding script blocks aliases in `allowed_aliases`.

E.g. `B. C` is not considered mixed-scripts by default: it contains characters from **Latin** and **Common**, but **Common** is excluded by default.

```
>>> confusables.is_mixed_script('Abç')
False
>>> confusables.is_mixed_script('.')
False
>>> confusables.is_mixed_script('.', allowed_aliases=[])
True
>>> confusables.is_mixed_script('Allo')
True
```

Parameters

- **string** (*str*) – A unicode string
- **allowed_aliases** (*list(str)*) – Script blocks aliases not to consider.

Returns
Whether `string` is considered mixed-scripts or not.

Return type
bool

4.1.5 confusable_homoglyphs.utils module

`confusable_homoglyphs.utils.`**`delete`**(*filename*)

Deletes a JSON data file if it exists.

`confusable_homoglyphs.utils.`**`dump`**(*filename*, *data*)

`confusable_homoglyphs.utils.`**`get`**(*url*, *timeout=None*)

`confusable_homoglyphs.utils.load(filename)`

Loads a JSON data file.

Returns
A dict.

Return type
dict

`confusable_homoglyphs.utils.path(filename)`

Returns a file path relative to the data directory.

This is the package directory by default, or the env variable CONFUSABLE_DATA if set.

Returns
A file path string.

Return type
str

`confusable_homoglyphs.utils.u(x)`

4.1.6 Module contents

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://todo.sr.ht/~valhalla/confusable_homoglyphs

If you are reporting a bug, please include:

- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the sourcehut tickets for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the sourcehut tickets for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

confusable_homoglyphs could always use more documentation, whether as part of the official confusable_homoglyphs docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://todo.sr.ht/~valhalla/confusable_homoglyphs.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *confusable_homoglyphs* for local development.

1. Clone the git repository from sourcehut:

```
$ git clone https://git.sr.ht/~valhalla/confusable_homoglyphs
```

2. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv confusable_homoglyphs
$ cd confusable_homoglyphs/
$ python setup.py develop
```

3. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 confusable_homoglyphs tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

5. Commit your changes:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
```

7. Send the patch to mailto:~valhalla/confusable_homoglyphs-devel@lists.sr.ht:

```
$ git send-email \
  --to="mailto:~valhalla/confusable_homoglyphs-devel@lists.sr.ht" \
  HEAD^
```

   you can see https://git-send-email.io/ for details on how to install and configure git-send-email.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for all supported Python versions.

CHAPTER

SIX

CREDITS

6.1 Original Author and Former Maintainer

- Victor Felder <victorfelder@gmail.com>

6.2 Current Maintainer

- Elena "of Valhalla" Grandi <valhalla@trueelena.org>

6.3 Contributors

- Ryan P Kilby <rpkilby@ncsu.edu>

CHAPTER

SEVEN

HISTORY

7.1 1.0.0

Initial release.

7.2 2.0.0

- *allowed_categories* renamed to *allowed_aliases*

7.3 2.0.1

- Fix a TypeError: https://github.com/vhf/confusable_homoglyphs/pull/2

7.4 3.0.0

Courtesy of Ryan P Kilby, via https://github.com/vhf/confusable_homoglyphs/pull/6 :

- Changed file paths to be relative to the *confusable_homoglyphs* package directory instead of the user's current working directory.
- Data files are now distributed with the packaging.
- Fixes tests so that they use the installed distribution instead of the local files. (Originally, the data files were erroneously showing up during testing, despite not being included in the distribution).
- Moves the data file generation into a simple CLI. This way, users have a method for controlling when the data files are updated.
- Since the data files are now included in the distribution, the CLI is made optional. Its dependencies can be installed with the *cli* bundle, eg. *pip install confusable_homoglyphs[cli]*.

7.5 3.1.0

- Update unicode data

7.6 3.1.1

- Update unicode data (via ftp)

7.7 3.2.0

- Drop support for Python 3.3
- Fix #11: work as expected when char not found in datafiles

7.8 3.3.0

- Drop support for Python 2
- Drop support for Python < 3.7, add support for Python up to 3.12
- Allow using data files from a custom location set with the CONFUSABLE_DATA environment variable.
- Fix the return value of confusables.is_dangerous() to the documented API of a boolean value. It used to return either False or the list output of confusable.is_confusable().
- Added a check command for command line use.

7.9 3.3.1

- Update unicode data

INDEX

A

alias() (*in module confusable_homoglyphs.categories*), 9
aliases_categories() (*in module confusable_homoglyphs.categories*), 9

C

category() (*in module confusable_homoglyphs.categories*), 10
confusable_homoglyphs
    module, 13
confusable_homoglyphs.categories
    module, 9
confusable_homoglyphs.confusables
    module, 10
confusable_homoglyphs.utils
    module, 12

D

delete() (*in module confusable_homoglyphs.utils*), 12
dump() (*in module confusable_homoglyphs.utils*), 12

F

Found, 10

G

get() (*in module confusable_homoglyphs.utils*), 12

I

is_confusable() (*in module confusable_homoglyphs.confusables*), 10
is_dangerous() (*in module confusable_homoglyphs.confusables*), 11
is_mixed_script() (*in module confusable_homoglyphs.confusables*), 12

L

load() (*in module confusable_homoglyphs.utils*), 12

M

module
    confusable_homoglyphs, 13
    confusable_homoglyphs.categories, 9
    confusable_homoglyphs.confusables, 10
    confusable_homoglyphs.utils, 12

P

path() (*in module confusable_homoglyphs.utils*), 13

U

u() (*in module confusable_homoglyphs.utils*), 13
unique_aliases() (*in module confusable_homoglyphs.categories*), 10