Dress up

Paulo S. Costa

Nov 27, 2022

CONTENTS

Convert your strings to various Unicode characters. Turn “words” into “”, “”, and “”.

CHAPTER

ONE

INSTALLATION

To install Dress up, run this command in your terminal:

```
python -m pip install dressup
```

If you're using it primarily as a command-line tool, it's recommended you install it via pipx.

```
pipx install dressup
```

CHAPTER

TWO

USAGE

There are two primary ways to use Dress up—as a command-line tool, or as Python library.

2.1 Command-line usage

Display all possible transformations by running:

```
 dressup Hello
Circle



Negative circle



Monospace



Math bold



...
```

Return only a specific transformation by using the `--type` flag.

```
 dressup Vibes --type inverted
ıqs
```

See more options in the *Command-line usage*, or by running

```
 dressup --help
```

2.2 Autocompletion

Dress up supports argument completions along with live previews. To enable autocompletion run.

```
 dressup --install-completion zsh
zsh completion installed in /Users/username/.zshrc.
Completion will take effect once you restart the terminal.
```

`zsh` may be replaced with `bash`, `fish`, `powershell`, or `pwsh`. Along with typical autocompletion, when typing in a value for `--type` if `[TAB]` is pressed the matching parameter values will be displayed below along with a preview of the conversion.

```
 dressup Words --type math [TAB]
math-bold              --
math-bold-fraktur      --
math-bold-italic       --
math-bold-script       --
math-double-struck     --
math-fraktur           --
math-monospace         --
math-sans              --
math-sans-bold         --
math-sans-bold-italic  --
math-sans-italic       --
```

2.3 Library usage

To convert characters, use `convert`.

```
import dressup

dressup.convert("Hello", unicode_type="negative circle")
```

```
''
```

To return all possible conversions, use `show_all`.

```
import dressup

dressup.show_all("Hello")
```

```
{'Circle': '', 'Negative circle': '',
'Monospace': '', 'Math bold': '',
'Math bold fraktur': '', 'Math bold italic': '',
'Math bold script': '', 'Math double struck': '',
'Math monospace': '', 'Math sans': '', 'Math sans bold':
'', 'Math sans bold italic': '', 'Math sans italic':
'', 'Parenthesized': '', 'Square': '',
'Negative square': '', 'Cute': 'HéĺĺŐ', 'Math fraktur':
'', 'Rock dots': 'ëö', 'Small caps': '', 'Stroked':
'ɨɬø', 'Subscript': '', 'Superscript': '',
'Inverted': 'o', 'Reversed': 'Ho'}
```

See *dressup.converter* for more arguments and examples.

CHAPTER

THREE

PRIOR ART

Pseudomappings were inspired by Unicode Text Converter.

CHAPTER

FOUR

INDEX

4.1 Reference

4.1.1 Command-line usage

Dress up's command-line usage looks like:

```
dressup [OPTIONS] [CHARACTERS]
```

-s, --strict-case
Do not fallback to different cases.

-r, --reverse
Reverse the output.

-t, --type TEXT
The Unicode type to convert to.

--version
Display the version and exit.

--help
Display a short usage message and exit.

--install-completion [bash|zsh|fish|powershell|pwsh]
Install completion for the specified shell.

--show-completion [bash|zsh|fish|powershell|pwsh]
Show completion for the specified shell, to copy it or customize the installation.

4.1.2 dressup.converter

Convert Unicode characters.

dressup.converter.**convert**(*characters: str*, *unicode_type: str*, *strict_case: bool = False*, *reverse: bool = False*) → str
Convert characters to a Unicode character type.

Parameters

- **characters** (*str*) – The characters to convert.
- **unicode_type** (*str*) – The type of Unicode character types to convert to. Valid values are "circle", "negative circle", "monospace", "math bold", "math bold fraktur", "math bold italic", "math bold script", "math double struck", "math monospace", "math sans",

"math sans bold", "math sans bold italic", "math sans italic", "parenthesized", "square", "negative square", "cute", "math fraktur", "rock dots", "small caps", "stroked", "subscript", "superscript", "inverted", and "reversed".

- **strict_case** (*bool*) – Whether to forbid a character from being converted to its lower or upper case counterpart if an exact mapping is not found. By default False.
- **reverse** (*bool*) – Whether to reverse the returned characters. This can be useful when converting to `unicode_type` "inverted" or "reverse". By default False.

Returns The converted Unicode characters.

Return type str

Raises **InvalidUnicodeTypeError** – Raised if value inputted in `unicode_type` is invalid.

Examples

Convert the string "Hello" to negative circle characters.

```
>>> import dressup
>>> dressup.convert("Hello", unicode_type="negative circle")
''
```

Convert the string "Hello" to negative circle characters, but don't convert lowercase to uppercase if a perfect match isn't found.

```
>>> import dressup
>>> dressup.convert(
...     "Hello",
...     unicode_type="negative circle",
...     strict_case=True,
... )
'ello'
```

Concvert the string "Hello" to reversed characters, but

```
>>> import dressup
>>> import dressup
>>> dressup.convert(
...     "Hello",
...     unicode_type="reversed",
...     reverse=True,
... )
'oH'
```

dressup.converter.**show_all**(*characters: str*, *strict_case: bool = False*, *reverse: bool = False*) → Dict[str, str]

Return all possible unicode conversions.

Parameters

- **characters** (*str*) – The characters to convert.
- **strict_case** (*bool*) – Whether to forbid a character from being converted to its lower or upper case counterpart if an exact mapping is not found. By default False.
- **reverse** (*bool*) – Whether to reverse the returned characters. This can be useful when converting to `unicode_type` "inverted" or "reverse". By default False.

Returns

A dictionary with the converted characters.

The dictionary keys are the names of character types and the values are the converted characters.

Return type Dict(str, str)

Example

Show all possible conversions for the string “Hello”.

```
>>> import dressup
>>> dressup.show_all("Hello")
{'Circle': '', 'Negative circle': '',
'Monospace': '', 'Math bold': '',
'Math bold fraktur': '', 'Math bold italic': '',
'Math bold script': '', 'Math double struck': '',
'Math monospace': '', 'Math sans': '', 'Math sans bold':
'', 'Math sans bold italic': '', 'Math sans italic':
'', 'Parenthesized': '', 'Square': '',
'Negative square': '', 'Cute': 'HéÍÍő', 'Math fraktur':
'', 'Rock dots': 'ëö', 'Small caps': '', 'Stroked':
'łłø', 'Subscript': '', 'Superscript': '',
'Inverted': 'o', 'Reversed': 'Ho'}
```

4.2 Contributor Guide

Thank you for your interest in improving this project. This project is open-source under the MIT License and welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- Source Code
- Documentation
- Issue Tracker
- *Code of Conduct*

4.2.1 How to report a bug

Report bugs on the Issue Tracker.

When filing an issue, make sure to answer these questions:

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

4.2.2 How to request a feature

Request features on the Issue Tracker.

4.2.3 How to set up your development environment

You need Python 3.6+ and the following tools:

- Poetry
- Nox

Install the package with development requirements:

```
poetry install
```

You can now run an interactive Python session, or the command-line interface:

```
poetry run python
poetry run dressup
```

4.2.4 How to test the project

Run the full test suite:

```
nox
```

List the available Nox sessions:

```
nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
nox --session=tests
```

Unit tests are located in the `tests` directory, and are written using the pytest testing framework.

4.2.5 How to submit changes

Open a pull request to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

You can ensure that your changes adhere to the code style by reformatting with Black:

```
nox --session=black
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

4.3 Contributor Covenant Code of Conduct

4.3.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

4.3.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

4.3.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

4.3.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

4.3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at Paulo.S.Costa.5@gmail.com. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

4.3.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

4.3.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

4.4 License

genindex

PYTHON MODULE INDEX

d

INDEX

Symbols

C

D

M

S