

[Search projects](#)[Help](#)[Donate](#)[Log in](#)[Register](#)

python-i18n 0.3.7



Latest version

`pip install python-i18n` Last released: Oct 4,
2019

Translation library for Python

Navigation

[Project
description](#) [Release
history](#) [Download
files](#)

Project links

[Homepage](#) [Download](#)

Statistics

GitHub statistics:

[Stars: 59](#)

Project description

python-i18n build unknown coverage 96% maintainability B

This library provides i18n functionality for Python 3 out of the box.
The usage is mostly based on Rails i18n library.

Installation

Just run

```
pip install python-i18n
```

If you want to use YAML to store your translations, use

```
pip install python-i18n[YAML]
```


 Forks: 20 

 Open

issues/PRs: 3 

View statistics for this project via

[Libraries.io](#) , or by

using [our public dataset on Google BigQuery](#) 

Meta

License: MIT License (MIT)

Author: [Daniel Perez](#) 

Maintainers



[tuvistavie](#)

Classifiers

Development Status

- 4 - Beta

Environment

- Other Environment

Intended Audience

- Developers

License

- OSI Approved :: MIT License

Operating System

- OS Independent

Programming Language

- Python :: 2

Usage

Basic usage

The simplest, though not very useful usage would be

```
import i18n
i18n.add_translation('foo', 'bar')
i18n.t('foo') # bar
```

Using translation files

YAML and JSON formats are supported to store translations. With the default configuration, if you have the following `foo.en.yml` file

```
en:
  hi: Hello world !
```

in `/path/to/translations` folder, you simply need to add the folder to the translations path.

```
import i18n
i18n.load_path.append('/path/to/translations')
i18n.t('foo.hi') # Hello world !
```

Please note that YAML format is used as default file format if you have `yaml` module installed. If both `yaml` and `json` modules available and you want to use JSON to store translations, explicitly specify that: `i18n.set('file_format', 'json')`

Memoization

Setting the configuration value `enable_memoization` in the settings dir will load the files from disk the first time they are loaded and then store their content in memory. On the next use the file content will be provided from memory and not loaded from disk, preventing disk access. While this can be useful in some contexts, keep in mind there is no current way of issuing a command to the reloader to re-read the

- [Python :: 3](#)

Topic

- [Software Development :: Libraries](#)

files from disk, so if you are updating your translation file without restarting the interpreter do not use this option.

Namespaces

File namespaces

In the above example, the translation key is `foo.hi` and not just `hi`. This is because the translation filename format is by default `{namespace}.{locale}.{format}`, so the `{namespace}` part of the file is used as translation.

To remove `{namespace}` from filename format please change the `filename_format` configuration.

```
i18n.set('filename_format', '{locale}.{format}')
```

Directory namespaces

If your files are in subfolders, the foldernames are also used as namespaces, so for example if your translation root path is `/path/to/translations` and you have the file `/path/to/translations/my/app/name/foo.en.yml`, the translation namespace for the file will be `my.app.name` and the file keys will therefore be accessible from `my.app.name.foo.my_key`.

Functionalities

Placeholder

You can of course use placeholders in your translations. With the default configuration, the placeholders are used by inserting `%{placeholder_name}` in the ntranslation string. Here is a sample usage.

```
i18n.add_translation('hi', 'Hello %{name} !')
i18n.t('hi', name='Bob') # Hello Bob !
```

Pluralization

Pluralization is based on Rail i18n module. By passing a `count` variable to your translation, it will be pluralized. The translation value should be a dictionary with at least the keys `one` and `many`. You can add a `zero` or `few` key when needed, if it is not present `many` will be used instead. Here is a sample usage.

```
i18n.add_translation('mail_number', {
    'zero': 'You do not have any mail.',
    'one': 'You have a new mail.',
    'few': 'You only have %{count} mails.',
    'many': 'You have %{count} new mails.'
})
i18n.t('mail_number', count=0) # You do not have any m
i18n.t('mail_number', count=1) # You have a new mail.
i18n.t('mail_number', count=3) # You only have 3 new m
i18n.t('mail_number', count=12) # You have 12 new mail
```

Fallback

You can set a fallback which will be used when the key is not found in the default locale.

```
i18n.set('locale', 'jp')
i18n.set('fallback', 'en')
i18n.add_translation('foo', 'bar', locale='en')
i18n.t('foo') # bar
```

Skip locale from root

Sometimes i18n structure file came from another project or not contains root element with locale eg. `en` name.




```
{
    "foo": "FooBar"
}
```

However we would like to use this i18n .json file in our Python sub-project or micro service as base file for translations. `python-i18n` has special configuration tha is skipping locale eg. `en` root data element from the file.




```
i18n.set('skip_locale_root_data', True)
```






Help

[Installing packages](#) 
[Uploading packages](#) 
[User guide](#) 
[FAQs](#)



About PyPI

[PyPI on Twitter](#) 
[Infrastructure dashboard](#) 
[Package index name retention](#) 
[Our sponsors](#)

Contributing to PyPI

[Bugs and feedback](#)
[Contribute on GitHub](#) 
[Translate PyPI](#) 
[Development credits](#) 

Using PyPI

[Code of conduct](#) 
[Report security issue](#)
[Privacy policy](#) 
[Terms of use](#)

Status: All Systems Operational 

Developed and maintained by the Python community, for the Python community.
[Donate today!](#)

© 2019 Python Software Foundation 
[Site map](#)

[Switch to desktop version](#)

» English español français 日本語 Português Brasileiro Українська Ελληνικά Deutsch

- Elastic
Search
- Pingdom
Monitoring
- Google
BigQuery
- Sentry
Error logging
- AWS
Cloud computing
- DataDog
Monitoring
- Fastly
CDN
- SignalFx
Supporter
- DigiCert
EV certificate
- StatusPage
Status page