

Do zero à publicação de um pacote de distribuição no PyPI

Alexandre Yukio Harano

São Paulo, 14 de Julho de 2018 – Just Python

`alexandre@harano.net.br`

`https://alexandre.harano.net.br/`

O que implementar?

- Gerenciadores de endereços IP: ferramentas em software usadas para planejar, rastrear e gerenciar endereços e subredes.
- Usado por administradores de redes corporativas, principalmente Provedores de Serviço Internet (ISPs): administrar redes delegadas, subredes de um bloco IP e endereços de equipamentos.

Baseado em https://en.wikipedia.org/wiki/IP_address_management



{php}IPAM

Open-source IP address management

[Download phpIPAM](#)

[News](#)

[Donate](#)

[Demo](#)

[Contact](#)

[Support](#)

[Api](#) ▼

[Documents](#) ▼

Home / phpIPAM IPAM IP address management /

The screenshot displays the phpIPAM web interface. At the top, there's a navigation bar with links for News, Donate, Demo, Contact, Support, Api, and Documents. Below this is a breadcrumb trail: Home / phpIPAM IPAM IP address management /. The main content area is titled 'Expired subnet shares:' and shows a table of IP addresses. The table has columns for IP address, Hostname, and Description. The first row shows the IP address 173.194.112.1 with a description of 'frs07s27-in-f2.1e100.net'. The second row shows the IP address 173.194.112.2 with a description of 'frs07s27-in-f2.1e100.net'. The interface also includes a sidebar with links for VRFs, IP addresses in subnets, and a search bar. A 'Get phpIPAM' button is visible in the bottom right corner.



Features do phpIPAM

phpIPAM Feature list

- ✓ IPv4/IPv6 IP address management
- ✓ Section / Subnet management
- ✓ Automatic free space display for subnets
- ✓ Visual subnet display
- ✓ Automatic subnet scanning / IP status checks
- ✓ PowerDNS integration
- ✓ NAT support
- ✓ RACK management
- ✓ Domain authentication (AD, LDAP, Radius)
- ✓ Per-group section/subnet permissions
- ✓ Device / device types management
- ✓ RIPE subnets import
- ✓ XLS / CVS subnets import
- ✓ IP request module
- ✓ REST API
- ✓ Locations module
- ✓ VLAN management
- ✓ VRF management
- ✓ IPv4 / IPv6 calculator
- ✓ IP database search
- ✓ E-mail notifications
- ✓ Custom fields support
- ✓ Translations
- ✓ Changelogs

E agora?

- Baseado na expressão *poor's man*: usado para comparar algo menos bem sucedido comparado a outra pessoa: He's a kind of poor man's James Bond.
- *Person* em vez de *man* para ser generalista.
- Similar ao bem conhecido phpIPAM.
- Python no nome.
- Referência fraca a PPPoE, um protocolo usado em Provedores de Serviço Internet.

Por quê?

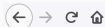


Table Of Contents

22.28. `ipaddress` — IPv4/IPv6 manipulation library

- 22.28.1. Convenience factory functions
- 22.28.2. IP Addresses
 - 22.28.2.1. Address objects
 - 22.28.2.2. Conversion to Strings and Integers
 - 22.28.2.3. Operators
 - 22.28.2.3.1. Comparison operators
 - 22.28.2.3.2. Arithmetic operators
- 22.28.3. IP Network definitions
 - 22.28.3.1. Prefix, net mask and host mask
 - 22.28.3.2. Network objects
 - 22.28.3.3. Operators
 - 22.28.3.3.1. Logical operators
 - 22.28.3.3.2. Iteration
 - 22.28.3.3.3. Network address

22.28. `ipaddress` — IPv4/IPv6 manipulation library

Source code: [Lib/ipaddress.py](#)

`ipaddress` provides the capabilities to create, manipulate and operate on IPv4 and IPv6 addresses and networks.

The functions and classes in this module make it straightforward to handle various tasks related to IP addresses, including checking whether or not two hosts are on the same subnet, iterating over all hosts in a particular subnet, checking whether or not a string represents a valid IP address or network definition, and so on.

This is the full module API reference—for an overview and introduction, see [An introduction to the `ipaddress` module](#).

New in version 3.3.

22.28.1. Convenience factory functions

The `ipaddress` module provides factory functions to conveniently create IP addresses, networks and interfaces:

`ipaddress.ip_address(address)`

Return an `IPv4Address` or `IPv6Address` object depending on the IP address passed as argument. The string `address` may be a string or a bytes object. It is considered to be IPv4 by default. A `ValueError` is raised if `address` does not represent a valid IPv4 or IPv6 address.



ipaddress

```
>>> import ipaddress
>>> rede_ipv6 = ipaddress.ip_network(
...     "2001:db8:01a::/64")
>>> endereco_ipv6 = ipaddress.ip_address(
...     "2001:db8:01a::a10")
>>> endereco_ipv6 in rede_ipv6
True
>>> doc_ipv6 = ipaddress.IPv6Network("2001:db8::/32")
>>> doc_ipv6.supernet_of(rede_ipv6)
True
>>> endereco_ipv6.version
6
>>>
```

Como começar do zero?

Python Software Foundation (US) | <https://docs.python.org/3/library/venv.html>

Python » English » 3.7.0 » Documentation » The Python Standard Library » 29. Software Packaging and Distribution »

Table Of Contents

- 29.3. **venv** — Creation of virtual environments
 - 29.3.1. Creating virtual environments
 - 29.3.2. API
 - 29.3.3. An example of extending `EnvBuilder`

Previous topic

29.2. **ensurepip** — Bootstrapping the `pip` installer

Next topic

29.4. **zipapp** — Manage executable python zip archives

This Page

[Report a Bug](#)
[Show Source](#)

29.3. **venv** — Creation of virtual environments

New in version 3.3.

Source code: [Lib/venv/](#)

The **venv** module provides support for creating lightweight “virtual environments” with their own site directories, optionally isolated from system site directories. Each virtual environment has its own Python binary (allowing creation of environments with various Python versions) and can have its own independent set of installed Python packages in its site directories.

See [PEP 405](#) for more information about Python virtual environments.

Note: The `pyvenv` script has been deprecated as of Python 3.6 in favor of using `python3 -m venv` to help prevent any potential confusion as to which Python interpreter a virtual environment will be based on.

29.3.1. Creating virtual environments

Creation of [virtual environments](#) is done by executing the command `venv`:

```
python3 -m venv /path/to/new/virtual/environment
```

- virtual environments (ambientes virtuais)
- Não confundir com máquina virtual
- Ambientes leves isolados com cópias próprias dos binários
- Isola dependências entre projetos
- Isola o ambiente do python usado pelo sistema

```
$ python -m venv .venv  
$ source ./venv/bin/activate  
(.venv) pppipam y$
```

Python Packaging User Guide

Python Software Foundation (US) | <https://packaging.python.org>

...

PyPA » Python Packaging User Guide » Quick

Table Of Contents

- Tutorials
- Guides
- Discussions
- PyPA specifications
- Project Summaries
- Glossary
- How to Get Support
- Contribute to this guide
- News

Python Packaging User Guide

Welcome to the *Python Packaging User Guide*, a collection of tutorials and references to help you distribute and install Python packages with modern tools.

This guide is maintained on [GitHub](#) by the [Python Packaging Authority](#). We happily accept any [contributions and feedback](#). 😊

Note: Looking for guidance on migrating from legacy PyPI to [pypi.org](#)? Please see [Migrating to PyPI.org](#).

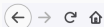
Get started

Essential tools and concepts for working with the Python packaging ecosystem are covered in our [Tutorials](#) section:

- to learn how to install packages, see the [tutorial on installing packages](#).
- to learn how to manage dependencies in a version controlled project, see the [tutorial on managing application dependencies](#).
- to learn how to package and distribute your projects, see the [tutorial on packaging and distributing](#)



Packaging Python Projects



Python Software Foundation (US) | <https://packaging.python.org/tutorials/packaging-projects/>



PyPA » Python Packaging User Guide » Tutorials »

Quick search

Table Of Contents

Tutorials

- Installing Packages
- Managing Application Dependencies
- Packaging Python Projects

Guides

Discussions

PyPA specifications

Project Summaries

Glossary

How to Get Support

Contribute to this guide

News

Previous topic

Managing Application
Dependencies

Next topic

Guides

Packaging Python Projects

This tutorial walks you through how to package a simple Python project. It will show you how to add the necessary files and structure to create the package, how to build the package, and how to upload it to the Python Package Index.

A simple project

This tutorial uses a simple project named `example_pkg`. If you are unfamiliar with Python's modules and [import packages](#), take a few minutes to read over the [Python documentation for packages and modules](#).

To create this project locally, create the following file structure:

```
/example_pkg  
/example_pkg  
__init__.py
```

Once you create this structure, you'll want to run all of the commands in this tutorial within the top-level folder - so be sure to `cd example_pkg`.

You should also edit `example_pkg/__init__.py` and put the following code in there:

```
name = "example_pkg"
```

This is just so that you can verify that it installed correctly later in this tutorial.



<https://github.com/ayharano/just-python/2018/>

Estrutura de mínima pelo tutorial

```
.
├── example_pkg
│   ├── LICENSE
│   ├── README.md
│   ├── example_pkg
│   │   └── __init__.py
│   └── setup.py
```

Estrutura inicial usada no PPPIPAM

```
.
├── pppipam
│   ├── LICENSE
│   ├── README.md
│   ├── pppipam
│   │   ├── __init__.py
│   └── tests
│       ├── __init__.py
```

Test Driven Development (TDD)

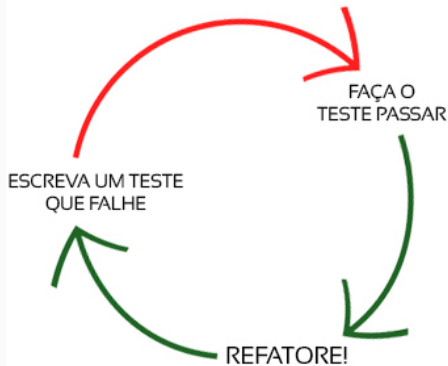


Imagem obtida de <http://tdd.caelum.com.br/>

Dicas relacionadas a TDD (e em geral)

- Use bastante o REPL para testar o comportamento esperado
- Leia com atenção os erros e falhas
- Gerou erro? Procure entender o que está escrito
- Não tenha medo de voltar alguns passos

Exemplo de TestCase do unittest

```
import dataclass
import unittest

from pppipam.pppipam import AddressSpace

class AddressSpace_dataclass_TestCase(unittest.TestCase):
    """Tests related to verify if AddressSpace is a dataclass."""
    def test_address_space_is_dataclass(self):
        """Validate if AddressSpace is a dataclass."""
        self.assertTrue(
            dataclasses.is_dataclass(AddressSpace),
            "AddressSpace expected to be a dataclass"
        )
```

Exibe docstrings ao acionar a builtin help.

```
>>> help(helpers.clean_network)
```

Help on function clean_network in module pppipam.helpers:

```
clean_network(network_parameter: Union[str, ipaddress.IPv4Network, ipaddress.IPv6Network]) ->  
    Process given parameter as a Network instance.
```

If parameter results into a valid IPv4 or IPv6 network,
it respectively returns IPv4Network or IPv6Network.
Otherwise, returns None.

```
>>> clean_network("invalid network")  
>>> clean_network("10.0.0.0/8")  
IPv4Network('10.0.0.0/8')  
>>> clean_network("fe80::/64")  
IPv6Network('fe80::/64')
```

Args:

network_parameter: value to be processed as an IP network.

Returns:

IPv4Network instance, IPv6Network instance or None.

- Fornece exemplo executável no REPL
- Diferente do propósito do unittest: apresenta exatamente o resultado esperado após execução

Exemplo de integração unittest e doctest

```
import doctest
import unittest

from pppipam import helpers, pppipam

def load_tests(loader, tests, ignore):
    """Base example provided in doctest documentation."""

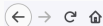
    tests.addTests(doctest.DocTestSuite(helpers))
    tests.addTests(doctest.DocTestSuite(pppipam))
    return tests
```

Exceptions

```
class StrictSupernetError(Exception):  
    """Error related to supernet missing or present."""  
    pass  
  
class SameDelegationAsNewError(Exception):  
    """Attempt to insert already existing delegated network."""  
    pass
```



Python Packaging User Guide



Python Software Foundation (US) | <https://packaging.python.org>



PyPA » Python Packaging User Guide »

Quick

Table Of Contents

- Tutorials
- Guides
- Discussions
- PyPA specifications
- Project Summaries
- Glossary
- How to Get Support
- Contribute to this guide
- News

Python Packaging User Guide

Welcome to the *Python Packaging User Guide*, a collection of tutorials and references to help you distribute and install Python packages with modern tools.

This guide is maintained on [GitHub](#) by the [Python Packaging Authority](#). We happily accept any [contributions and feedback](#). 😊

Note: Looking for guidance on migrating from legacy PyPI to [pypi.org](#)? Please see [Migrating to PyPI.org](#).

Get started

Essential tools and concepts for working with the Python packaging ecosystem are covered in our [Tutorials](#) section:

- to learn how to install packages, see the [tutorial on installing packages](#).
- to learn how to manage dependencies in a version controlled project, see the [tutorial on managing application dependencies](#).
- to learn how to package and distribute your projects, see the [tutorial on packaging and distributing](#)



<https://github.com/ayharano/just-python/2018/>

Packaging Python Projects

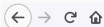


Table Of Contents

Tutorials

- Installing Packages
- Managing Application Dependencies
- Packaging Python Projects

Guides

Discussions

PyPA specifications

Project Summaries

Glossary

How to Get Support

Contribute to this guide

News

Previous topic

Managing Application
Dependencies

Next topic

Guides

Packaging Python Projects

This tutorial walks you through how to package a simple Python project. It will show you how to add the necessary files and structure to create the package, how to build the package, and how to upload it to the Python Package Index.

A simple project

This tutorial uses a simple project named `example_pkg`. If you are unfamiliar with Python's modules and [import packages](#), take a few minutes to read over the [Python documentation for packages and modules](#).

To create this project locally, create the following file structure:

```
/example_pkg  
/example_pkg  
__init__.py
```

Once you create this structure, you'll want to run all of the commands in this tutorial within the top-level folder - so be sure to `cd example_pkg`.

You should also edit `example_pkg/__init__.py` and put the following code in there:

```
name = "example_pkg"
```

This is just so that you can verify that it installed correctly later in this tutorial.



pypa/sampleproject

GitHub, Inc. (US) | <https://github.com/pypa/sampleproject>



Search or jump to...



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



[pypa](#) / [sampleproject](#)

Watch ▾

90

★ Star

1,866

Fork

829

<> Code

Issues 13

Pull requests 4

Projects 0

Wiki

Insights

A sample project that exists for PyPUG's "Tutorial on Packaging and Distributing Projects"

117 commits

1 branch

0 releases

28 contributors

MIT

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾



di Merge pull request #74 from ncoghlan/patch-1 ...

Latest commit 129382f on 6 Jun

data	add data_files example	4 years ago
sample	defer the topic of single-sourcing the version to the PPUG.	4 years ago
tests	have the tests we're including actually run and pass	3 years ago
.gitignore	have the tests we're including actually run and pass	3 years ago
.travis.yml	Test against 3.5 and 3.6	7 months ago
LICENSE.txt	Add a LICENSE.txt file	3 years ago
MANIFEST.in	Migrated README to Markdown (PEP 566)	4 months ago
README.txt	Fix README links	3 months ago
setup.cfg	Removed confusing minutiae	3 months ago



<https://github.com/ayharano/just-python/2018/>

24

Estrutura final usada no PPPIPAM

```
.
├── pppipam
│   ├── CHANGELOG.rst
│   ├── CONTRIBUTORS.rst
│   ├── LICENSE
│   ├── README.md
│   ├── setup.py
│   ├── setup.cfg
│   └── pppipam
│       ├── __init__.py
│       ├── helpers.py
│       ├── pppipam.py
│       └── test_strictness.py
└── .
    ├── .
    └── .
```

Estrutura final usada no PPPIPAM

```
.
├── pppipam
│   ├── .
│   ├── .
│   └── tests
│       ├── __init__.py
│       ├── test_dataclass.py
│       ├── test_description.py
│       ├── test_helpers.py
│       ├── test_pppipam_doctests.py
│       └── test_strictness.py
```

- Source distribution
- Wheels
 - Universal Wheels (Python 2 e 3)
 - Pure Python Wheels (Python somente 2 ou somente 3)
 - Platform Wheels (Linux, macOS, Windows, ...)



 <https://test.pypi.org>



⚠ You are using TestPyPI – a separate instance of the Python Package Index that allows you to try distribution tools and processes without affecting the real index.



Welcome back, ayharano ▾

Test Python package publishing with the Test Python Package Index

Search projects



Or browse projects

25,835 projects

113,987 releases

142,452 files

33,724 users

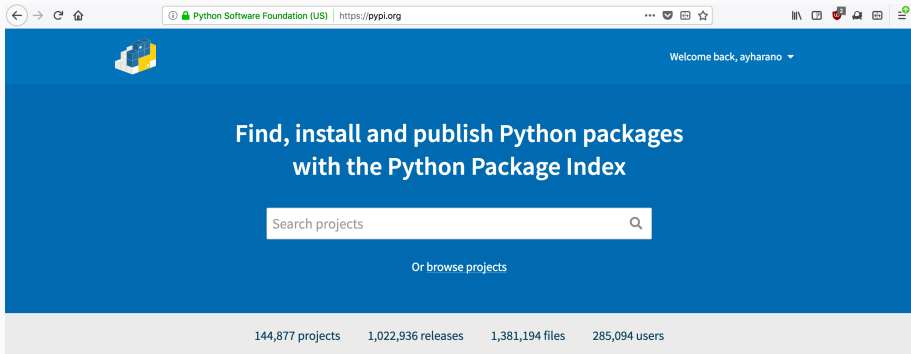


The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. [Learn about installing packages.](#)

Package authors use PyPI to distribute their software. [Learn how to package your Python code](#)

```
$ python3 -m pip install \
  --index-url https://test.pypi.org/simple/ pppipam
Looking in indexes: https://test.pypi.org/simple/
Collecting pppipam
  Downloading https://test-files.pythonhosted.org/packages/.../pppipam-0.1.0-py3-none-any.whl
Installing collected packages: pppipam
Successfully installed pppipam-0.1.0
```



The screenshot shows the PyPI website homepage. At the top, there's a navigation bar with the PyPI logo on the left and a user greeting "Welcome back, ayharano" on the right. The main content area has a large blue background with the text "Find, install and publish Python packages with the Python Package Index". Below this is a search bar labeled "Search projects" with a magnifying glass icon. Underneath the search bar is the text "Or browse projects". At the bottom of the main content area, there are four statistics: "144,877 projects", "1,022,936 releases", "1,381,194 files", and "285,094 users".

Find, install and publish Python packages
with the Python Package Index

Search projects

Or browse projects

144,877 projects 1,022,936 releases 1,381,194 files 285,094 users



The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. [Learn about installing packages.](#)

Package authors use PyPI to distribute their software. [Learn how to package your Python code for PyPI.](#)

PyPI: <https://pypi.org/>

- Python Package Index (PyPI)
- Repositório padrão de software para Python

```
(.venv) pppipam y$ twine upload --sign dist/*  
Uploading distributions to https://upload.pypi.org/legacy/  
Enter your username: ayharano  
Enter your password:  
Signing pppipam-0.1.0-py3-none-any.whl  
Uploading pppipam-0.1.0-py3-none-any.whl  
100%|██████████| 15.0k/15.0k [00:02<00:00, 5.64kB/s]  
Signing pppipam-0.1.0.tar.gz  
Uploading pppipam-0.1.0.tar.gz  
100%|██████████| 15.2k/15.2k [00:01<00:00, 10.3kB/s]  
(.venv) pppipam y$
```

<https://pypl.org/project/pppipam/>
67%
...
...
...

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

pppipam 0.1.0

Last released: About 4 hours ago

Poor person's Python IP Address Manager

Navigation

[Project description](#)

[Release history](#)

[Download files](#)

Project links

[Homepage](#)

[Source](#)

[Bug Reports](#)

Statistics

GitHub statistics:

★ Stars: 0

🍴 Forks: 0

🐛 Open Issues/PBs: 0

View statistics for this project via [Libraries.io](#), or by using [Google BigQuery](#)

Meta

License: MIT License

Project description

PPPIPAM: Poor Person's Python IP Address Manager

[pypl 0.1.0](#)

[license](#)

[python 3.7](#)

PPPIPAM is a distribution package to provide a single IP address space manager for both IPv4 and IPv6 as a Python module for developers.

Installation

PPPIPAM can be installed using `pip`. It requires Python 3.7.0+ to use.

```
$ pip install pppipam
```

Usage

```
>>> from pppipam import AddressSpace
```

Features

- Single address space manager for both IPv4 and IPv6 networks and addresses.
- Strict or loose address space description (If strict, must add delegated networks first).

```
$ pip install pppipam
Collecting pppipam
  Downloading https://files.pythonhosted.org/packages/.../pppi
Installing collected packages: pppipam
Successfully installed pppipam-0.1.0
```

```
$ python
```

```
Python 3.7.0 (default, Jun 29 2018, 23:55:57)
```

```
[Clang 9.1.0 (clang-902.0.39.2)] on darwin
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> from pppipam import AddressSpace
```

```
>>>
```



```
$ pip3 install pppipam
```

```
$ python3
```

```
>>> from pppipam import AddressSpace
```

```
>>>
```

Obrigado!

Alexandre Yukio Harano

alexandre@harano.net.br

`https://alexandre.harano.net.br/`

`https://github.com/ayharano/pppipam`

`https://pypi.org/project/pppipam/`

`https://github.com/ayharano/just-python/2018/`