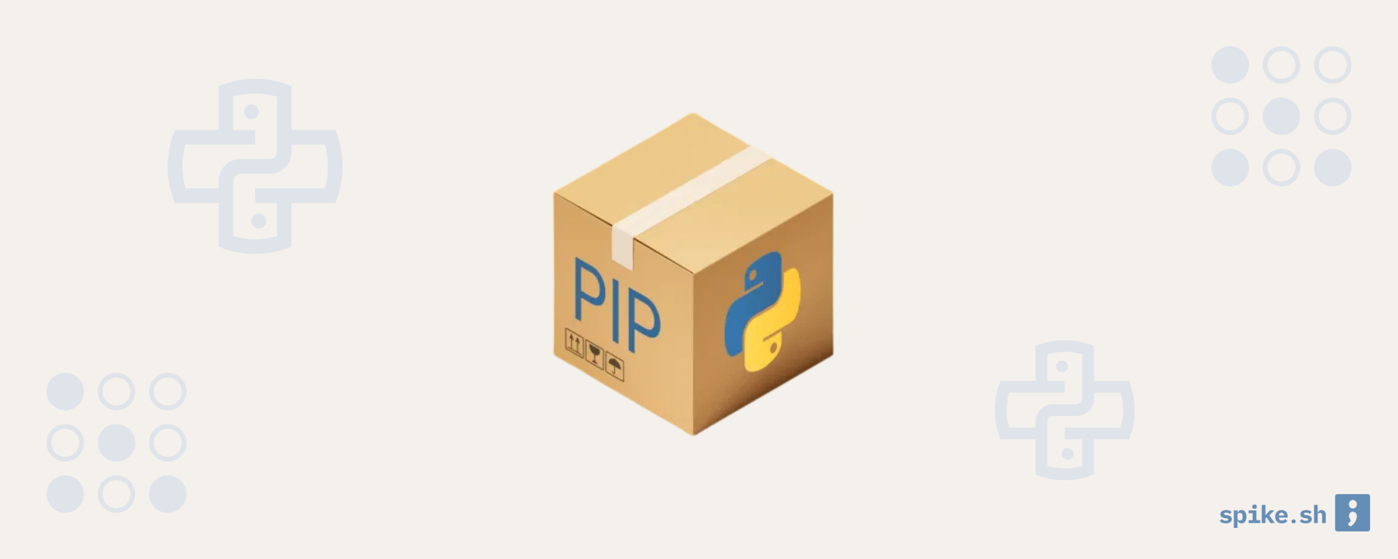
How to create a pip package for Python

PRUTHVI
3 APR 2021 • 2 MIN READ



Python is one of the most popular languages in the world, used for everything from data science to web applications. And one of the reasons is the rich library of packages available via its package management system called pip.

In this post, we will learn how to create our own Python package and upload it to the Python Package Index where millions of developers can access it.

Create your project

Create your Python project with the following structure.

```
myproject/
    -- src/
        -- hello_package/
            -- __init__.py
    -- pyproject.toml
    -- setup.cfg
    -- README.md
```

hello_package: This is the package directory and should be created inside the *src* directory. Create an empty *__init__.py* file inside your package directory.

README.md: This file contains the details of your package that will be shown on your package page. It is in Markdown format.

pyproject.toml: This file is required by the build tools. You should copy the contents below and paste it inside your file.

```
[build-system]
requires = [
    "setuptools>=42",
    "wheel"
]
build-backend = "setuptools.build_meta"
```

setup.cfg: This file contains your package metadata like name, author, license etc. You should copy the content below and change the package name, version and other details specific to your package. **Make sure to choose a unique package name that doesn't exist already.**

```
[metadata]
name = hello-package
version = 1.0.0
author = James
author_email = james@example.com
description = A small example package
long_description = file: README.md
long_description_content_type = text/markdown
url = https://example.com
project_urls =
    Docs = https://docs.example.com
classifiers =
    Programming Language :: Python :: 3
    License :: OSI Approved :: MIT License
    Operating System :: OS Independent

[options]
package_dir =
    = src
packages = find:
python_requires = >=3.6

[options.packages.find]
where = src
```

Adding some functionality

The package is ready to be created, but it doesn't do anything right now. Let's add a simple method that prints hello on the console. Copy the code below and paste it in the *__init__.py* file.

```
def hello():
    print("Hello! You look nice today")
```

Create the package

To create a distribution package, install *build.*

```
$ python3 -m pip install --upgrade build
```

Now run the build command from the directory where the *pyproject.toml* is located.

```
$ python3 -m build
```

This will create a new directory named *dist* and the packages will be created inside it.

Upload package

To upload your package to Python Package Index, create an account on pypi. Inside your account, create a new API token. Copy this token once created because it will be shown only once. **This token will be used as password later.**

Next, install *twine* package, and use it to upload the packages in *dist* directory.

```
$ python3 -m pip install --user --upgrade twine
```

```
$ python3 -m twine upload dist/*
```

That's it! If all goes well, your packages will be uploaded to the package index and you will be shown the URL of the newly uploaded package.

Let's now install and test it. **Use the package name that you chose in setup.cfg above, which could be different from the one I use here.**

```
$ pip install hello-package
```

Inside another Python file in your project, you can import the package and use it.

```
# test.py

# Import package
from hello_package import hello

# Call function
hello()
```

Conclusion

As you can see, it's easy to get started with distributing your software via pip for Python developers. You can now build on this foundation to build more complex and powerful

---

Product

- Tour
- Pricing
- Features
- Services
- On-call
- Escalations
- Integrations
- Alerts
- Reports
- Schedule a demo
- FAQ

Guides

- AWS Integration
- Google Cloud Integration
- Webhook Integration
- Datadog Integration
- Sematext Integration
- New Relic Integration
- Healthchecks Integration
- Apex Ping Integration
- Microsoft Azure Integration
- Sentry Integration
- Pingdom Integration

Resources

- Docs
- Intro to Spike
- What is an incident?
- Incident lifecycle
- Incident statuses
- Monitoring checklist

Compare

- Spike.sh vs. Pagerduty
- Spike.sh vs. Opsgenie
- Spike.sh vs. Splunk On-Call (VictorOps)

Company

- Security
- Contact us
- Privacy policy
- Terms & conditions
- Press kit

Robot Integration

Honeybadger Integration

Prometheus Integration

Grafana Integration

Twilio Integration

Travis CI Integration

Bugsnag Integration

Cronitor Integration

Got questions? Email us at demo@spike.sh or support@spike.sh

spike.sh