PCL2–Tutorial 04

Lucas Suomela, Rong Li, Tosca Peruzzi-Vieli, Isabelle Cretton

Welcome!

Introduction to the __init__.py File

*The __**init**__.py file initializes a Python package when imported, defining package-level variables, importing submodules, and executing initialization code.*

Package Initialization

- Presence of __**init**__.py file signals directory as a package
- Without it, Python wouldn't recognize directory as a package
- Essential for proper functioning of imports within the package
- Acts as marker for Python to understand package hierarchy
- Facilitates better project organization and management

Module Exports

- __**init**__.py controls package exports
- Explicit imports expose modules/symbols
- Enhances package accessibility
- Simplifies package usage for end-user

Custom initialization in __init__.py
Examples: logging setup, constants, dependencies

Best practices:
- Keep it concise
- Use relative imports

Packaging (click me)

Python Packaging Tools

distutils

- Python's built-in packaging library
- Provides basic functionality for packaging and distribution
- Lacks advanced features compared to setuptools
- Does not support dependency management or package metadata

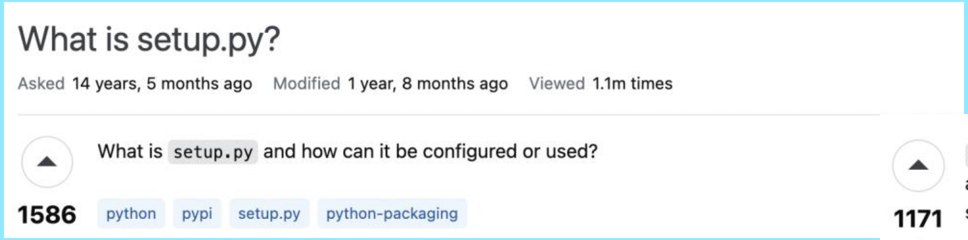
Pip

- Python's package installer
- Allows easy installation and management of packages
- Retrieves packages from PyPI or other repositories
- Automatically resolves dependencies for installation

Setuptools

- Widely used Python packaging library
- Simplifies packaging and distribution
- Extends capabilities of distutils
- Offers features like dependency management and plugin support

setup.py (click me)



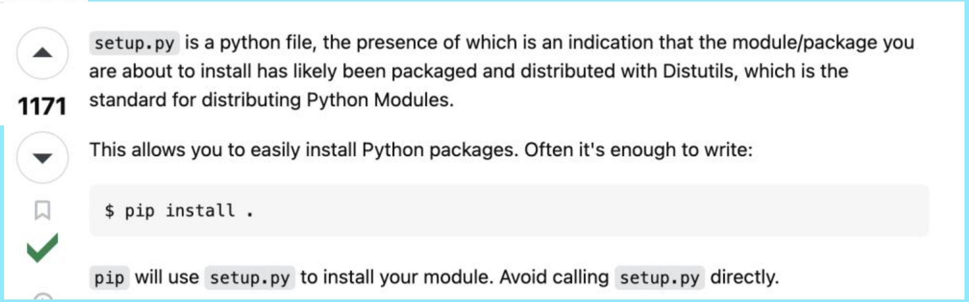
What is setup.py?

Asked 14 years, 5 months ago Modified 1 year, 8 months ago Viewed 1.1m times

1586

What is `setup.py` and how can it be configured or used?

python pypi setup.py python-packaging



1171

`setup.py` is a python file, the presence of which is an indication that the module/package you are about to install has likely been packaged and distributed with Distutils, which is the standard for distributing Python Modules.

This allows you to easily install Python packages. Often it's enough to write:

```
$ pip install .
```

`pip` will use `setup.py` to install your module. Avoid calling `setup.py` directly.

Read the whole thread here:

(more documentation here)

Versioning

Semantic Versioning

- **Definition:** Semantic versioning (often abbreviated as SemVer) is a versioning scheme that specifies a structured format for version numbers, typically represented as major.minor.patch.
- **Major Version:** Increments when there are incompatible API changes.
- **Minor Version:** Increases for backward-compatible functionality additions.
- **Patch Version:** Updated for backward-compatible bug fixes.
- **Example:** A change that adds new features without breaking existing functionality would increment the minor version (e.g., 2.1.0 to 2.2.0), while a change that introduces breaking changes would necessitate a major version increment (e.g., 2.1.0 to 3.0.0).

Importance

- **Tracking Changes:** Semantic versioning provides a clear and consistent way to communicate changes in software.
- **Managing Dependencies:** By adhering to semantic versioning, developers can effectively manage dependencies, ensuring compatibility with the versions of packages they rely on.
- **Communication:** Version numbers convey information about the nature and impact of changes, aiding developers and users in understanding the implications of upgrading to a new version.
- **Stability:** Consistent versioning helps maintain stability in software ecosystems by providing a standardized approach to version management.
- **Community Adoption:** Semantic versioning has gained widespread adoption across the software development community, making it easier for developers to understand and work with different packages and libraries.

TOML and YAML

TOML: Tom's Obvious Minimal Language

- Minimal configuration file format
- "Obvious" semantics => easy to read!
- Has implementations in almost every popular programming language
- File extension: **.toml**

YAML: YAML Ain't Markup Language

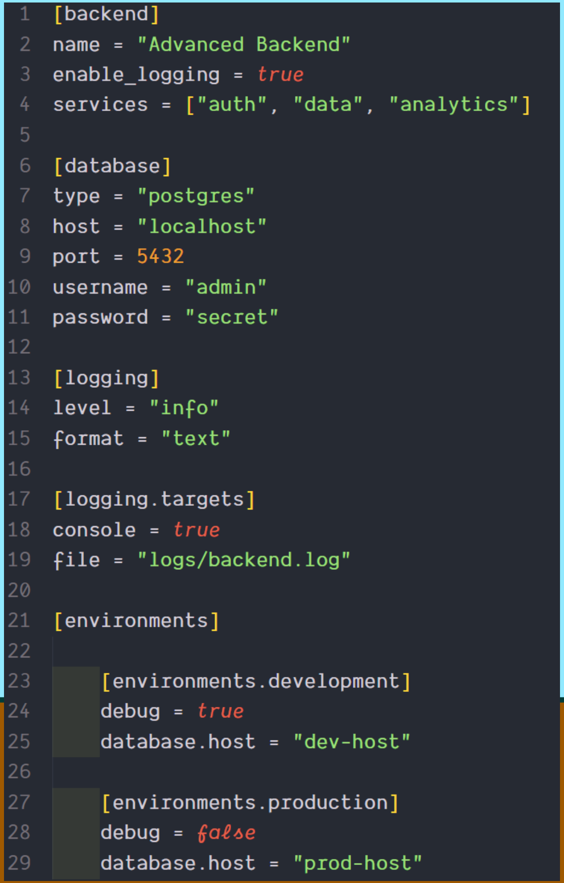
- Human-readable data serialization language
- Used for (not only) configuration files
- Encodes data types based on the Perl programming language
- File extension **.yml** is still used, but the IETF finalized the media type yaml with the file extension **.yaml**

See the official websites for syntax rules

TOML YAML

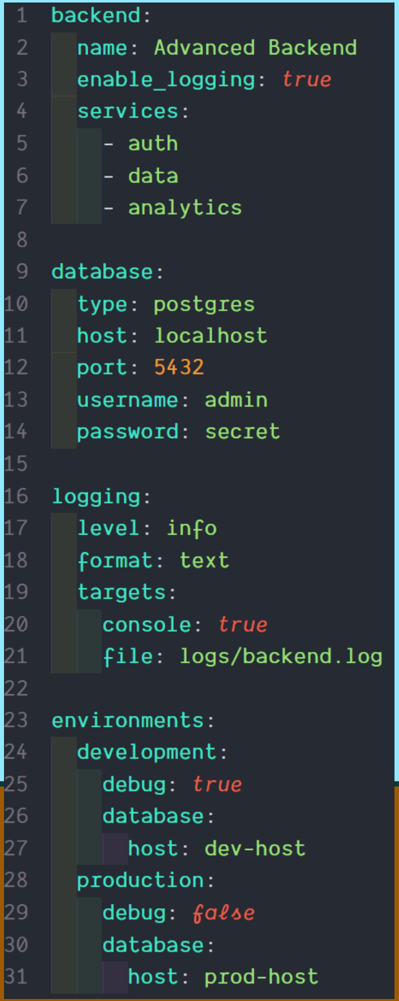
TOML and YAML

TOM
L



```toml
[backend]
name = "Advanced Backend"
enable_logging = true
services = ["auth", "data", "analytics"]

[database]
type = "postgres"
host = "localhost"
port = 5432
username = "admin"
password = "secret"

[logging]
level = "info"
format = "text"

[logging.targets]
console = true
file = "logs/backend.log"

[environments]

    [environments.development]
    debug = true
    database.host = "dev-host"

    [environments.production]
    debug = false
    database.host = "prod-host"
```

YAML



```yaml
backend:
  name: Advanced Backend
  enable_logging: true
  services:
    - auth
    - data
    - analytics

database:
  type: postgres
  host: localhost
  port: 5432
  username: admin
  password: secret

logging:
  level: info
  format: text
  targets:
    console: true
    file: logs/backend.log

environments:
  development:
    debug: true
    database:
      host: dev-host
  production:
    debug: false
    database:
      host: prod-host
```

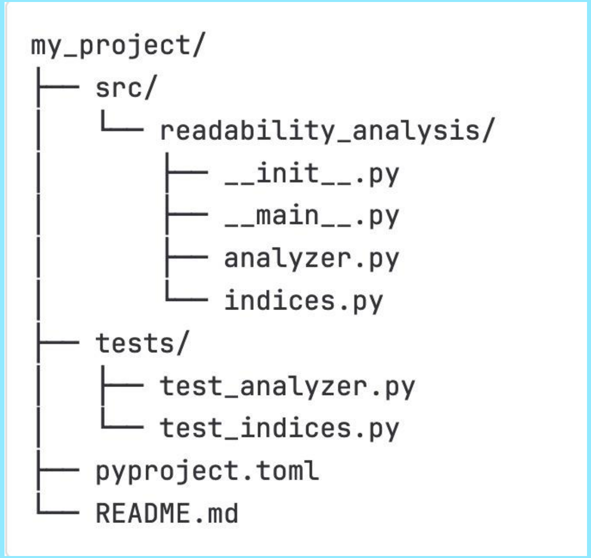
Exercise 03

Objectives

Learn to transform a Python module into an object-oriented package.

- *pyproject.toml* file crafting
- Package structuring
- *TestPyPI* uploading
- Testing pipeline implementing

package structure sample:



```
my_project/
├── src/
│   └── readability_analysis/
│       ├── __init__.py
│       ├── __main__.py
│       ├── analyzer.py
│       └── indices.py
├── tests/
│   ├── test_analyzer.py
│   └── test_indices.py
├── pyproject.toml
└── README.md
```

Provided Modules:
readability_analysis.py
test_readability_analysis.py

Now It's Your Turn...

To-Do

- ❑ Work on the exercise
- ❑ Issues with GitLab? Please let us know.
- ❑ ~~Have a nice dinner date with chatGPT~~
- ❑ Enjoy your weekend!