

Installing your own package

DEVELOPING PYTHON PACKAGES



James Fulton

Climate informatics researcher

Why should you install your own package?

Inside `example_script.py`

```
import mysklearn
```

Directory tree for package with subpackages

```
home/  
|-- mysklearn                                <-- in same directory  
|   |-- __init__.py  
|   |-- preprocessing  
|   |   |-- __init__.py  
|   |   |-- normalize.py  
|   |   |-- standardize.py  
|   |-- regression  
|   |   |-- __init__.py  
|   |   |-- regression.py  
|   `-- utils.py  
|-- example_script.py                        <-- in same directory
```

Why should you install your own package?

Inside `example_script.py`

```
import mysklearn
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'mysklearn'
```

Directory tree

```
home/
|-- mypackages
|   |-- mysklearn    <---
|       |-- __init__.py
|       |-- preprocessing
|           |-- __init__.py
|           |-- normalize.py
|           |-- standardize.py
|       |-- regression
|           |-- __init__.py
|           |-- regression.py
|-- myscripts
    |-- example_script.py    <---
```

setup.py

- Is used to install the package
- Contains metadata on the package

Package directory structure

Directory tree for package with subpackages

```
mysklearn/  
|-- __init__.py  
|-- preprocessing  
|   |-- __init__.py  
|   |-- normalize.py  
|   |-- standardize.py  
|-- regression  
|   |-- __init__.py  
|   |-- regression.py  
|-- utils.py
```

Package directory structure

Directory tree for package with subpackages

```
mysklearn/      <-- outer directory
|-- mysklearn   <--- inner source code directory
    |-- __init__.py
    |-- preprocessing
        |-- __init__.py
        |-- normalize.py
        |-- standardize.py
    |-- regression
        |-- __init__.py
        |-- regression.py
    |-- utils.py
```

Package directory structure

Directory tree for package with subpackages

```
mysklearn/      <-- outer directory
|-- mysklearn   <--- inner source code directory
|   |-- __init__.py
|   |-- preprocessing
|   |   |-- __init__.py
|   |   |-- normalize.py
|   |   |-- standardize.py
|   |-- regression
|   |   |-- __init__.py
|   |   |-- regression.py
|   |-- utils.py
|-- setup.py    <-- setup script in outer
```

Inside setup.py

```
# Import required functions
from setuptools import setup

# Call setup function
setup(
    author="James Fulton",
    description="A complete package for linear regression.",
    name="mysklearn",
    version="0.1.0",
)
```

version number = (major number) . (minor number) . (patch number)

Inside setup.py

```
# Import required functions
from setuptools import setup, find_packages

# Call setup function
setup(
    author="James Fulton",
    description="A complete package for linear regression.",
    name="mysklearn",
    version="0.1.0",
    packages=find_packages(include=["mysklearn", "mysklearn.*"]),
)
```

Editable installation

```
pip install -e .
```

- `.` = package in current directory
- `-e` = editable

Directory tree for package with subpackages

```
mysklearn/ <-- navigate to here
|-- mysklearn
|   |-- __init__.py
|   |-- preprocessing
|   |   |-- __init__.py
|   |   |-- normalize.py
|   |   |-- standardize.py
|   |-- regression
|   |   |-- __init__.py
|   |   |-- regression.py
|   |-- utils.py
|-- setup.py
```

Let's practice!
DEVELOPING PYTHON PACKAGES

Dealing with dependencies

DEVELOPING PYTHON PACKAGES



James Fulton

Climate informatics researcher

What are dependencies?

- Other packages you import inside your package
- Inside `mymodule.py` :

```
# These imported packages are dependencies
import numpy as np
import pandas as pd
...
```

Adding dependencies to setup.py

```
from setuptools import setup, find_packages

setup(
    ...
    install_requires=['pandas', 'scipy', 'matplotlib'],
)
```

Controlling dependency version

```
from setuptools import setup, find_packages

setup(
    ...
    install_requires=[
        'pandas>=1.0',
        'scipy==1.1',
        'matplotlib>=2.2.1,<3'
    ],
)
```

Controlling dependency version

```
from setuptools import setup, find_packages

setup(
    ...
    install_requires=[
        'pandas>=1.0',          # good
        'scipy==1.1',            # bad
        'matplotlib>=2.2.1,<3'  # good
    ],
)
```

- Allow as many package versions as possible
- Get rid of unused dependencies

Python versions

```
from setuptools import setup, find_packages

setup(
    ...
    python_requires='>=2.7, !=3.0.*, !=3.1.*',
)
```

Choosing dependency and package versions

- Check the package history or release notes
 - e.g. the [NumPy release notes](#)
- Test different versions

Release Notes

- 1.19.0
 - Highlights
 - Expired deprecations
 - `numpy.insert` and `numpy.delete` can no longer be passed an axis on 0d arrays
 - `numpy.delete` no longer ignores out-of-bounds indices
 - `numpy.insert` and `numpy.delete` no longer accept non-integral indices
 - `numpy.delete` no longer casts boolean indices to integers
 - Compatibility notes
 - Changed random variate stream from `numpy.random.Generator.dirichlet`
 - Scalar promotion in `PyArray_ConvertToCommonType`
 - Fasttake and fastputmask slots are deprecated and NULL'ed
 - `np.ediff1d` casting behaviour with `to_end` and `to_begin`
 - Converting of empty array-like objects to NumPy arrays
 - Removed `multiarray.int_asbuffer`

Making an environment for developers

```
pip freeze
```

```
alabaster==0.7.12  
appdirs==1.4.4  
argh==0.26.2  
...  
wrapt==1.11.2  
yapf==0.29.0  
zipp==3.1.0
```

Making an environment for developers

Save package requirements to a file

```
pip freeze > requirements.txt
```

Install requirements from file

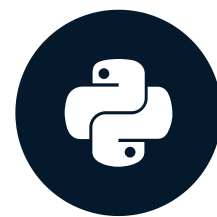
```
pip install -r requirements.txt
```

```
mysklearn/  
|-- mysklearn  
|   |-- __init__.py  
|   |-- preprocessing  
|   |   |-- __init__.py  
|   |   |-- normalize.py  
|   |   |-- standardize.py  
|   |-- regression  
|   |   |-- __init__.py  
|   |   |-- regression.py  
|   |-- utils.py  
|-- setup.py  
|-- requirements.txt    <-- developer environment
```

Let's practice!
DEVELOPING PYTHON PACKAGES

Including licences and writing READMEs

DEVELOPING PYTHON PACKAGES



James Fulton

Climate informatics researcher

Why do I need a license?

- To give others permission to use your code

Open source licenses

- Find more information [here](https://choosealicense.com)
- Allow users to
 - use your package
 - modify your package
 - distribute versions of your package

¹ <https://choosealicense.com>

What is a README?

- The "front page" of your package
- Displayed on Github or PyPI

What to include in a README

README sections

- Title
- Description and Features
- Installation
- Usage examples
- Contributing
- License

README format

Markdown (commonmark)

- Contained in `README.md` file
- Simpler
- Used in this course and in the wild

reStructuredText

- Contained in `README.rst` file
- More complex
- Also common in the wild

Commonmark

Contents of README.md

What it looks like when rendered

Commonmark

Contents of README.md

```
# mysklearn
mysklearn is a package for complete
**linear regression** in Python.

You can find out more about this package
on [DataCamp](https://datacamp.com)
```

What it looks like when rendered

mysklearn

mysklearn is a package for complete **linear regression** in python.

You can find out more about this package on [DataCamp](https://datacamp.com)

Commonmark

Contents of README.md

```
# mysklearn
mysklearn is a package for complete
**linear regression** in Python.

You can find out more about this package
on [DataCamp](https://datacamp.com)

## Installation
You can install this package using
```

What it looks like when rendered

mysklearn

mysklearn is a package for complete **linear regression** in python.

You can find out more about this package on [DataCamp](https://datacamp.com)

Installation

You can install this package using

Commonmark

Contents of README.md

```
# mysklearn
mysklearn is a package for complete
**linear regression** in Python.

You can find out more about this package
on [DataCamp](https://datacamp.com)

## Installation
You can install this package using

...

pip install mysklearn
...
```

What it looks like when rendered

mysklearn

mysklearn is a package for complete **linear regression** in python.

You can find out more about this package on [DataCamp](https://datacamp.com)

Installation

You can install this package using

```
pip install mysklearn
```

Adding these files to your package

Directory tree for package with subpackages

```
mysklearn/  
|-- mysklearn  
|   |-- __init__.py  
|   |-- preprocessing  
|   |   |-- ...  
|   |-- regression  
|   |   |-- ...  
|   |-- utils.py  
|-- setup.py  
|-- requirements.txt  
|-- LICENSE      <--- new files  
|-- README.md    <--- added to top directory
```


MANIFEST.in

Lists all the extra files to include in your package distribution.

MANIFEST.in

Contents of MANIFEST.in

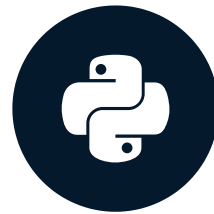
```
include LICENSE
include README.md
```

```
mysklearn/
|-- mysklearn
|   |-- __init__.py
|   |-- preprocessing
|   |   |-- ...
|   |-- regression
|   |   |-- ...
|   |-- utils.py
|-- setup.py
|-- requirements.txt
|-- LICENSE
|-- README.md
|-- MANIFEST.in <---
```

Let's practice!
DEVELOPING PYTHON PACKAGES

Publishing your package

DEVELOPING PYTHON PACKAGES



James Fulton

Climate informatics researcher

PyPI

Python Package Index

- `pip` installs packages from here
- Anyone can upload packages
- You should upload your package as soon as it might be useful

¹ <https://pypi.org/>

Distributions

- **Distribution package** - a bundled version of your package which is ready to install.
- **Source distribution** - a distribution package which is mostly your source code.
- **Wheel distribution** - a distribution package which has been processed to make it faster to install.

How to build distributions

```
python setup.py sdist bdist_wheel
```

- `sdist` = source distribution
- `bdist_wheel` = wheel distribution

```
mysklearn/  
|-- mysklearn  
|-- setup.py  
|-- requirements.txt  
|-- LICENSE  
|-- README.md  
|-- dist <---  
|   |-- mysklearn-0.1.0-py3-none-any.whl  
|   |-- mysklearn-0.1.0.tar.gz  
|-- build  
|-- mysklearn.egg-info
```

Getting your package out there

Upload your distributions to [PyPI](#)

```
twine upload dist/*
```

Upload your distributions to [TestPyPI](#)

```
twine upload -r testpypi dist/*
```

```
mysklearn/  
|-- mysklearn  
|-- setup.py  
|-- requirements.txt  
|-- LICENSE  
|-- README.md  
|-- dist  
|   |-- mysklearn-0.1.0-py3-none-any.whl  
|   |-- mysklearn-0.1.0.tar.gz  
|-- build  
|-- mysklearn.egg-info
```


How other people can install your package

Install package from **PyPI**

```
pip install mysklearn
```

Install package from **TestPyPI**

```
pip install --index-url https://test.pypi.org/simple  
            --extra-index-url https://pypi.org/simple  
            mysklearn
```

Let's practice!
DEVELOPING PYTHON PACKAGES