UNIVERSITY OF LEEDS

Testing & Packaging Python Code

Zoe Hancox

Z.L.Hancox@leeds.ac.uk

PyPi

pytest

By the end of this session...

UNIVERSITY OF LEEDS

- Understand why testing your code is important and how to write tests for you code in Pytest
- Have a go at writing basic Pytests for example code
- Learn what makes a good code repository
- Turn a repository into a simple PyPI package

You will need:

- A GitHub account
- Git on your machine
- A TestPyPI account
- Anaconda

Format of session

UNIVERSITY OF LEEDS

- What is testing and how to Pytest
- Task 1: Pytesting tutorial
- Task 2: Compare repositories
- Making nice GitHub repositories (for others, but mostly your future self)
- What is packaging and how to package
- Task 3: Turn your code into a TestPyPI package
- Next steps

What is testing?

UNIVERSITY OF LEEDS

Check your code works as you expect it to in a variety of situations

When writing tests for your code, try to find ways your code could break, test to make sure it doesn't.

How many tests should you write?

It can be time consuming to produce tests, but what is the cost of your code being wrong?

Why do we need to test code?

UNIVERSITY OF LEEDS

Testing frameworks promotes:

- Code quality
- Reliability
- Maintainability
- Catching bugs early
- Error reduction

Benefits:

- Validation: Verify that your code behaves and gives the expected results.
- Regression Testing: Detect and fix issues/bugs when modifying your code.
- Continuous Integration: Ensures your code remains functional and reliable as you develop new features or make modifications.
- Code Maintainability: Refactor or modify your code with confidence.
- Collaboration: Everyone can quickly verify that their changes have not broken existing functionality.

Types of Tests

Test	Description
Unit Testing	Tests parts of the code in chunks.
Regression Testing	Looks for a specific output given a certain input. Used after changing code e.g. adding a new feature.
Functional Testing	Tests for a specific behaviour.
Fuzzing Testing	Testing random data.
Stress Testing	Attempting to overwhelm/flood the system to check for stability.

Different Python Testing Libraries

unittest: built-in testing framework in Python's standard library

doctest: write tests in docstrings of your functions

hypothesis: useful for generating test cases

pytest

pytest: 3rd party, supports extras e.g. fixtures, parameterized testing, and test discovery

nose

is nicer testing for python

nose: extension of Unittest, automatically finds test cases

Example file structure

UNIVERSITY OF LEEDS

- src
- test_streamlit
  - __init__.py
  - test_edge_count.py
- workbooks
- .gitignore
- LICENSE
- MANIFEST.in
- README.md

```python
def test_max_bf_hyperarcs():
    """Testing the calculation of the maximum number of BF-hyperarcs is
    correct.

    Hand calculated answer versus numpy calculated.
    """
    n_dis = 3
    exp_ans = 15
    numpy_ans = numpy_utils.N_max_hyperarcs(n_dis, b_hyp=False)
    assert numpy_ans == exp_ans
```

Folder and file names should begin with ‘test’

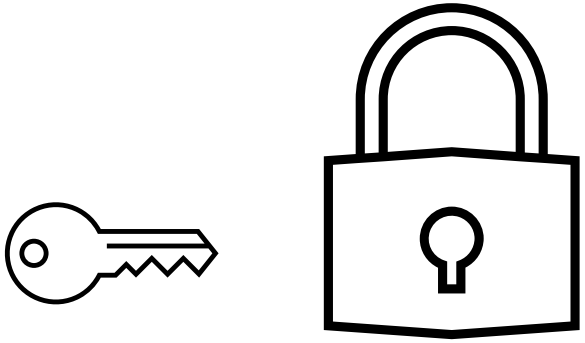
Your project to PyTest and PyPI-ify

UNIVERSITY OF LEEDS

Password generator

Requirements:

- Password must be more than 8 letters long
- One punctuation character must be included
- One capital letter must be included
- One number must be included



Task 1: Practice Pytesting

UNIVERSITY OF LEEDS

1. Go to GitHub repository: https://github.com/ZoeHancox/pytesting_and_pypi_tutorial
2. Follow the README.md instructions.

Task 2: Compare these repositories

UNIVERSITY OF LEEDS

README GPL-3.0 license

Hypergraphs for Multimorbidity Research

Tests no status DOI

This is a collection of tools for constructing and analysing hypergraphs from data. Hypergraphs are very general and powerful objects for data analysis which connect nodes and edges. As for binary graphs, nodes can conect to any number of edges but in a hypergraph, edges can connect to any number of nodes which leads to some very useful features!

https://github.com/SwanseaUniversityMedical/multimorbidity_hypergraphs

What's good,
what's missing,
how could they be better?

README Code of conduct MIT license

Hypergraph Multimorbidity (hypergraph-mm)

NHS England - Digital Analytics and Research Team - PhD Internship Projects

About the Project

status experimental Tests passing

style MkDocs Material code style black

https://github.com/nhsx/hypergraph-mm

README MIT license

Hypergraphical

About the project

status experimental code style black

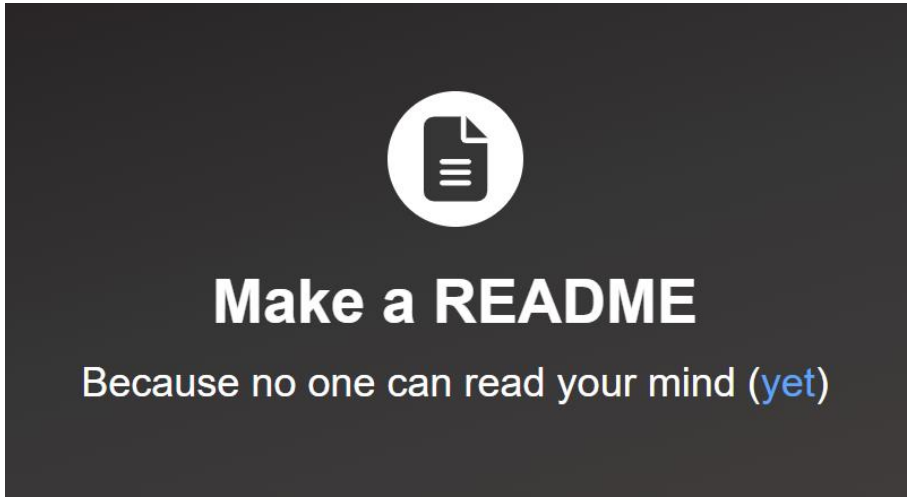
https://github.com/nhsx/hypergraphical

Writing a good ReadMe file

UNIVERSITY OF LEEDS

- Title
- Brief project description
- Project status (e.g. experimental)
- Contents table
- Installation
  - Prerequisites (e.g. Python version)
- How to install (e.g. clone, install package)
- How to use (simple example)
- Features
- How to contribute (*optional*)
- License
- Contact details for questions and help
- Acknowledgements (references, thanks)
- How to reference the repository (include a DOI (Zenodo))

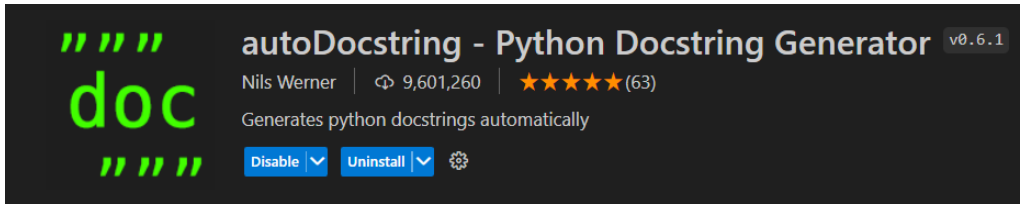
https://www.makeareadme.com/



https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax

Make your functions pretty

UNIVERSITY OF LEEDS




```python
def create_edges_df(patient_graph, act_graph):

    num_edges = np.count_nonzero(patient_graph)
    edges_df = pd.DataFrame(columns=['start_node', 'end_node', 'activated', 'weight', 'time_
    num_nodes = patient_graph.shape[1]
    timesteps = patient_graph.shape[0]

    row_num = 0
    for t in range(timesteps):
        for i in range(num_nodes):
            for j in range(num_nodes):
                if patient_graph[t, i, j] != 0:
                    if t == 0:
                        start_node_v = 0
                        end_node_v = 1
                    else:
                        start_node_v = t
                        end_node_v = t+1

                    edges_df.at[row_num, 'end_node'] = f'{j}_v{end_node_v}' #[row num, col r
                    edges_df.at[row_num, 'weight'] = act_graph[t, i, j]
                    edges_df.at[row_num, 'time_between'] = patient_graph[t, i, j] #add a col
                    edges_df.at[row_num, 'start_node'] = f'{i}_v{start_node_v}' #[row num, c

                    row_num += 1
```

```python
def create_max_act_df(class_name:str, pat_graphs:np.array, filters:np.array, labels:list, verbose:bool):
    """Calculate the maximum activation from each filter on each patient graph. Running a filter over
    each patient graph and getting the max.
    Assumes a stride length of one.

    Args:
        class_name (str): name to describe prediction outcome.
        pat_graphs (np.array): 4D array containing x 3D patient graphs.
        filters (np.array): 4D array containing x filters.
        labels (list): list of binary values representing positive or negative outcomes.
        verbose (bool): print or not to print extra dataframes or print statements.

    Raises:
        ValueError: if numpy array isn't 4D.

    Returns:
        pd.DataFrame: dataframe with columns for filter number, maximum activation and chosen class_name string.
    """
```

```python
, verbose:bool=False, show_plot:bool=False):
```

Adding buttons to your repo

UNIVERSITY OF LEEDS

How to add a button to show your Pytests are working

Tests passing

1. .github\workflows
   tests.yml
2. __init__.py

Create a tests.yml file as a workflow within ‘actions’ on your GitHub repository.

Make sure to add an __init__.py file to the src folder otherwise it won’t be found. Any folder which you import files from should contain a __init__.py file.

Issues with the test files finding the folder containing the python files, hence we need to make the python path find it manually in the tests.yml file. Src should be changed to the highest directory name.

3.

```yaml
name: Tests

on:
  push:
    branches: [ main ]

jobs:
  test:
    runs-on: windows-latest

    steps:
      - uses: actions/checkout@main
      - name: Set up Python 3.8
        uses: actions/setup-python@main
        with:
          python-version: 3.8

      - name: Install dependencies
        run: pip install -r requirements.txt

      - name: Add src to PYTHONPATH
        run: |
          echo "PYTHONPATH=$env:PYTHONPATH;$($PWD.Path)/src" >> $env:GITHUB_ENV

      - name: Test with pytest
        run: pytest test_graphs/test_calculations.py
```

What is code packaging?

Organising and distributing code in a way that allows others to easily install and use it

Different methods to package code

UNIVERSITY OF LEEDS

Zip your Python files up, downloadable files

COOKIECUTTER

pypi v2.6.0 | python 3.7 | 3.8 | 3.9 | 3.10 | 3.11 | 3.12 | CI/CD Tests passing | codecov 100% | Discord cookiecutter | docs passing | code quality 9.21

Cookiecutter

Create projects swiftly from **cookiecutters** (project templates) with this command-line utility. Ideal for generating Python package projects and more.

docker

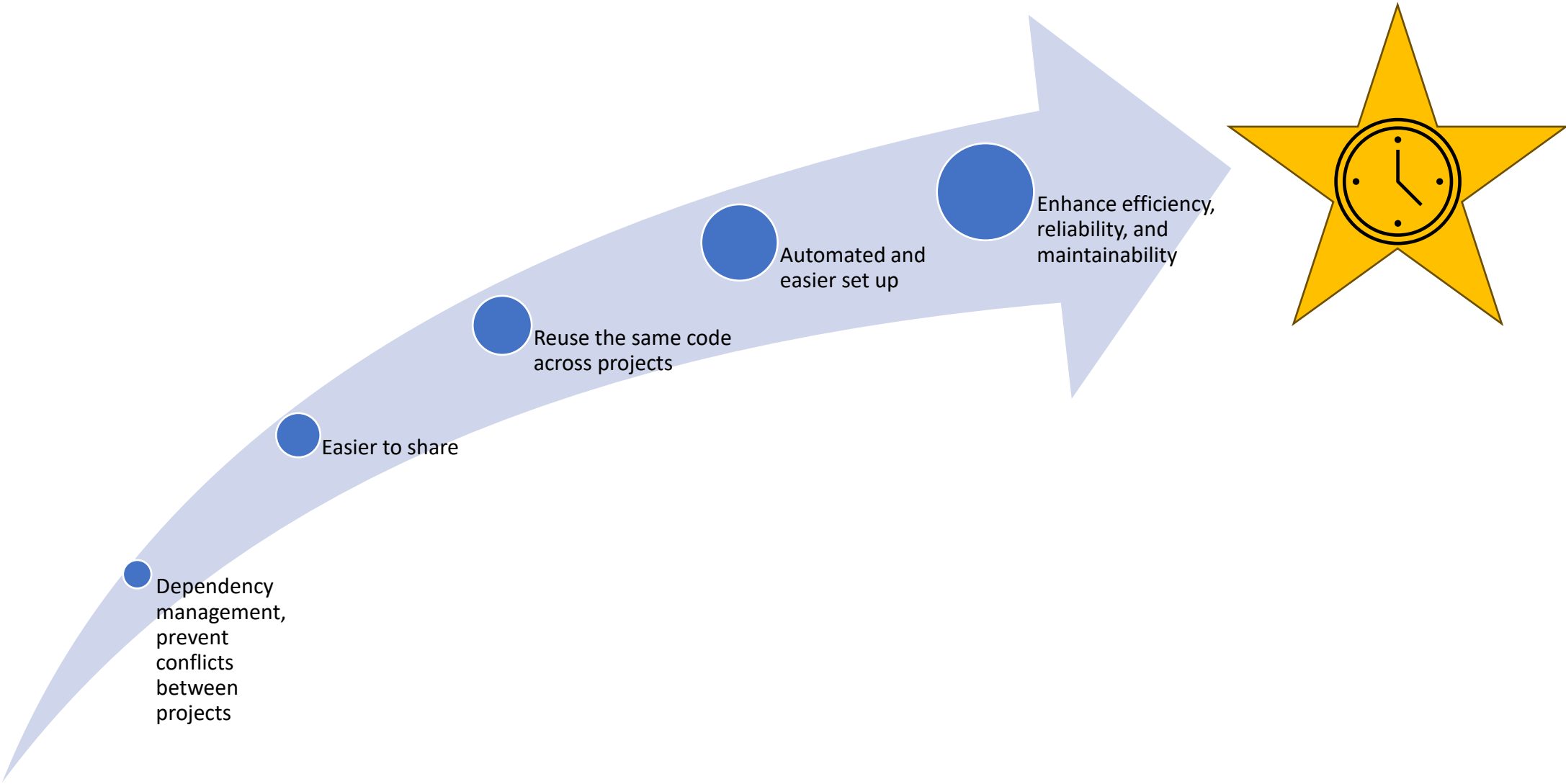
Package your code and its environment in one container

PyPi

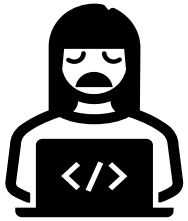
Package your code and let people use your package by running 'pip install <package_name>'

'conda install <package_name>'

Why is packaging code useful?



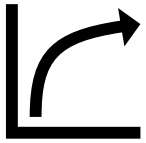
Limitations & Challenges



Documentation and testing time

Maintenance burden (fixing bugs, updating for new Python versions)

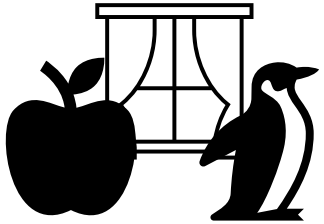
Security risks (vulnerabilities, malicious packages)



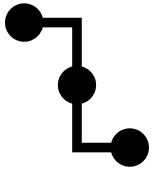
Steep learning curve

$V1.2$

Versioning Compatibility



OS Compatibility



Dependency conflicts

Turn a repo into a PyPI package

UNIVERSITY OF LEEDS

>pip install –-upgrade build
>pip install setuptools wheel twine

Create setup.py file:

setup.py U ×

tgcnn_activation_graphs > setup.py

```python
from setuptools import setup, find_packages

setup(
    name="tgcnn_act_graph",
    version="0.1",
    packages=find_packages(),
    install_requires=[
        #
    ]
)
```

Build your package:
>python setup.py sdist bdist_wheel

To get your READ.md markdown on your PyPI Page add:

```python
from setuptools import setup, find_packages


with open("README.md", r) as f:
    description = f.read()

setup(
    name="tgcnn_act_graph",
    version="0.2",
    packages=find_packages(),
    install_requires=[
        #
    ],
    long_description=description,
    long_description_content_type="text/markdown",
)
```

Turn a repo into a PyPI package

UNIVERSITY OF LEEDS

To **test your code locally** and install your package in a new env:
>pip install dist/tgcnn_act_graph-0.1-py3-none-any.whl <change to your .whl file name>

To see your package on the Python env and it should appear in a list of your packages:
>pip list

```
tgcnn_act_graph     0.1
```

```
from tgcnn_act_graph.figures import edge_activated_graph
import numpy as np
```

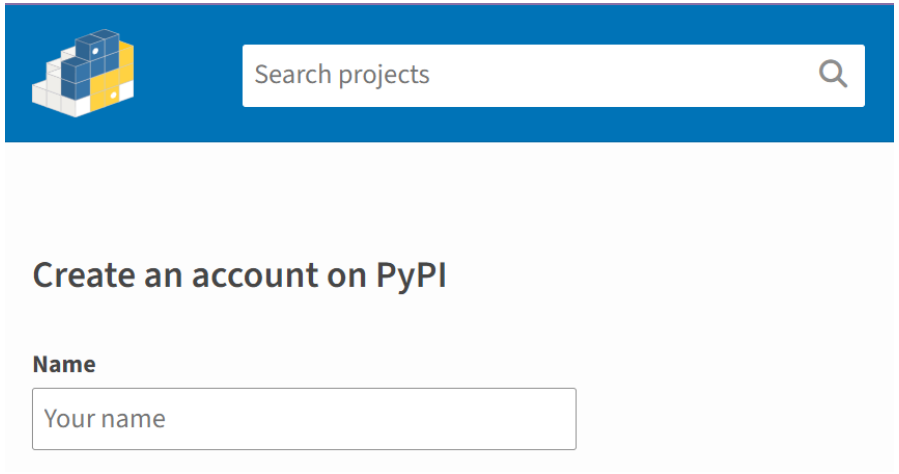
```
edge_activated_graph(input_tensors=input_tensors, patient_number=2, filters=filters, labels=labels,  verbose=False, show_plot=False)
```

Note: you can add suffix –force-reinstall after .whl if it says it's already installed

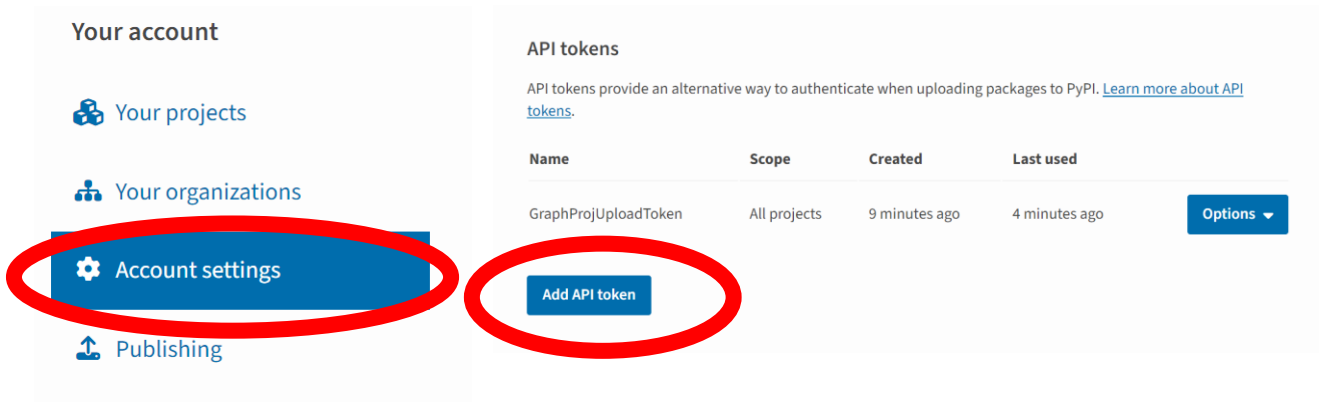
Turn a repo into a PyPI package

UNIVERSITY OF LEEDS

1.



2.



3. In your **cmd** window:

```
>set TWINE_USERNAME=__token__
> set TWINE_PASSWORD=your-token-value
---
>twine upload dist/*
```

OR

In your **PowerShell**:

```
>$env:TWINE_USERNAME="__token__"
>$env:TWINE_PASSWORD="your-token-value“
---
>twine upload dist/*
```

When uploading new versions:

```
>twine upload --skip-existing dist/*
```

Note: __token__ is literally kept as __token__ it's only the your-token-value that's changed and generated via the API

Installing a PyPI Package

UNIVERSITY OF LEEDS

tgcnn-act-graph 0.3.1

```
pip install tgcnn-act-graph==0.3.1
```

Create a new Python environment:
>conda create --name test_package_env python=3.8
>conda activate test_package_env
>pip install tgcnn-act-graph

Task 3: Create a password generator package on TestPyPI

UNIVERSITY OF LEEDS

Let's use TestPyPI to play around with making packages.
(TestPyPI is a good place to play around with making packages, rather than clogging up PyPI)

Follow the instructions in the README.md file:
https://github.com/ZoeHancox/pytesting_and_pypi_tutorial

Turn your password generating code into a package ☺

Next steps

UNIVERSITY OF LEEDS

v1234.1

Versioning

Code formatters and style guides

Documentation

Creating issues

autopep8 v2024.0.0 Preview

Microsoft microsoft.com | 2,109,151 | ★★★☆☆ (12)

Formatting support for Python files using the autopep8 formatter.

Install

DETAILS FEATURES CHANGELOG DEPENDENCIES

Formatter extension for Visual Studio Code using autopep8

A Visual Studio Code extension with support for the autopep8 formatter. The extension ships with `autopep8=2.1.0`.

> Note: The minimum supported version of autopep8 is `1.7.0`. If you have any issues with formatting with autopep8, please report it to this issue tracker as this extension is just a wrapper around autopep8.

This extension supports for all actively supported versions of the Python language (i.e., Python >= 3.8).

For more information on autopep8, see https://pypi.org/project/autopep8/.

MkDocs Home Getting Started User Guide Developer Guide About Search Previous Next Edit on GitHub

MkDocs

Project documentation with Markdown.

MkDocs is a **fast**, **simple** and **downright gorgeous** static site generator that's geared towards building project documentation. Documentation source files are written in Markdown, and configured with a single YAML configuration file. Start by reading the introductory tutorial, then check the User Guide for more information.

NumPy User Guide API reference Building from source Development Release notes Learn More 2.0 (stable)

NumPy documentation

Version: 2.0

Download documentation: Historical versions of documentation

Useful links: Installation | Source Repository | Issue Tracker | Q&A Support | Mailing List

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Getting started

New to NumPy? Check out the Absolute Beginner's

User guide

The user guide provides in-depth information on the

Code Issues Pull requests