

How to make an awesome package in python

Michalis Panayides

PGR-Talks

About me



About me



THIS.

Tools for an awesome python package

based on Anton Zhiyanov's blog

Awesome Python Package

Tools for an awesome python package

based on Anton Zhiyanov's blog

Awesome

Python Package

- ▶ Git
- ▶ GitHub
- ▶ Python
- ▶ Flit

Tools for an awesome python package

based on Anton Zhiyanov's blog

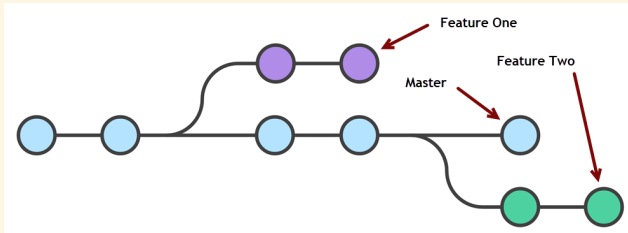
Awesome

- ▶ Documentation
- ▶ Testing
- ▶ Linters
- ▶ Tox
- ▶ GitHub actions

Python Package

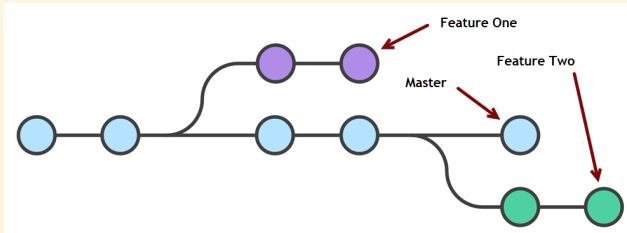
- ▶ Git
- ▶ GitHub
- ▶ Python
- ▶ Flit

Git

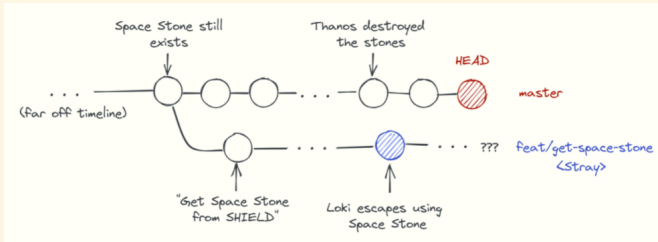


Git visualisation tool

Git

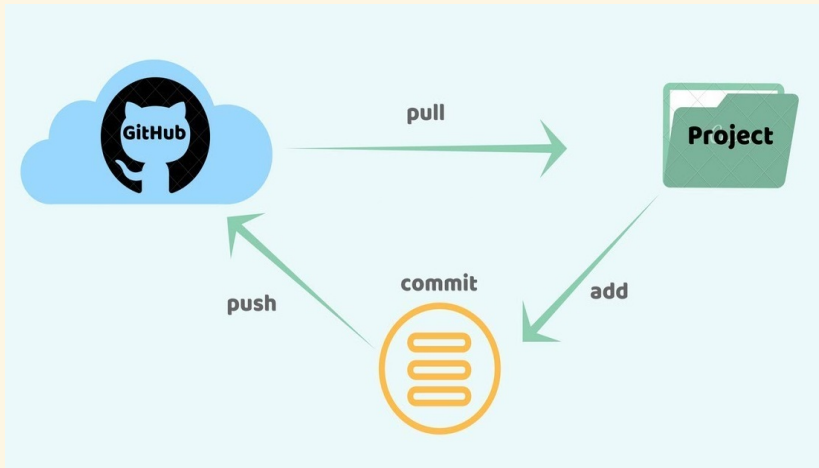


Git visualisation tool



Git VS Avengers: Endgame

GitHub



Python









Python



IP[y]: IPython
Interactive Computing



Rock-Paper-Scissors

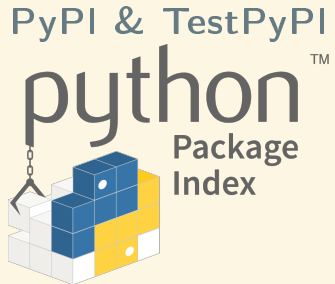
			
	0	+1	-1
	-1	0	+1
	+1	-1	0

Flit

- ▶ Initialising
- ▶ Packaging
- ▶ Publishing

Flit

- ▶ Initialising
- ▶ Packaging
- ▶ Publishing



Documentation

- ▶ **Readme.md**: A markdown file that contains an overview of the project
- ▶ **Changelog.md**: A file with a log of all the changes ever made to the project
- ▶ **Docstrings**: A comment in the code that is used to explain a block of code

Testing

```
def convert_symbolic_transition_matrix(Q_sym, lambda_2, lambda_1, mu):
    """Converts the symbolic matrix obtained from the get_symbolic_transition_matrix()
    function to the corresponding numerical matrix. The output of this function
    should be the same as the output of get_transition_matrix()

    Parameters
    -----
    Q_sym : sympy.matrices object
        The symbolic transition matrix obtained from get_symbolic_transition_matrix()

    Returns
    -----
    numpy.ndarray
        The transition matrix Q

    T000: get rid of first four lines somehow
    """
    sym_lambda = sym.symbols("lambda")
    sym_lambda_1 = sym.symbols("lambda_1")
    sym_lambda_2 = sym.symbols("lambda_2")
    sym_mu = sym.symbols("mu")

    Q = np.array(
        Q_sym.subs(
            {
                sym_lambda: lambda_2 + lambda_1,
                sym_lambda_1: lambda_1,
                sym_lambda_2: lambda_2,
                sym_mu: mu,
            }
        )
    ).astype(np.float64)
    return Q
```

```
@given(threshold=Integers(min_value=0, max_value=10))
@settings(deadline=None)
def test_convert_symbolic_transition_matrix(threshold):
    """
    Test that ensures that for fixed parameters and different values of the threshold
    the function that converts the symbolic matrix into a numeric one gives the
    same results as the get_transition_matrix function.
    """
    lambda_2 = 0.3
    lambda_1 = 0.2
    mu = 0.05
    num_of_servers = 10
    system_capacity = 8
    buffer_capacity = 2

    transition_matrix = get_transition_matrix(
        lambda_2=lambda_2,
        lambda_1=lambda_1,
        mu=mu,
        num_of_servers=num_of_servers,
        threshold=threshold,
        system_capacity=system_capacity,
        buffer_capacity=buffer_capacity,
    )

    sym_transition_matrix = get_symbolic_transition_matrix(
        num_of_servers=num_of_servers,
        threshold=threshold,
        system_capacity=system_capacity,
        buffer_capacity=buffer_capacity,
    )

    converted_matrix = convert_symbolic_transition_matrix(
        Q_sym=sym_transition_matrix, lambda_2=lambda_2, lambda_1=lambda_1, mu=mu
    )

    assert np.allclose(converted_matrix, transition_matrix)
```

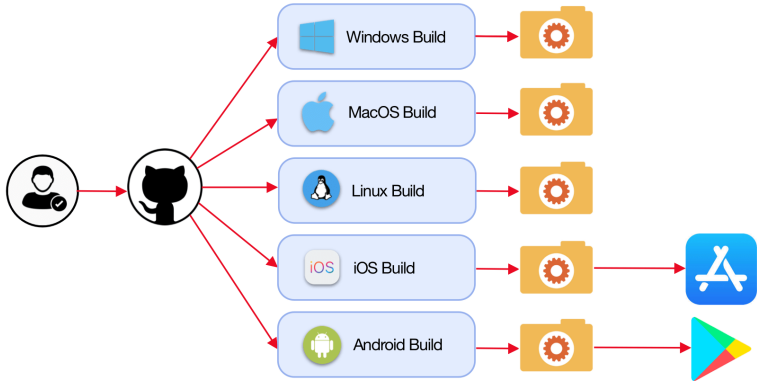

Linters

Linting is the automated checking of your source code for programmatic and stylistic errors without running the code

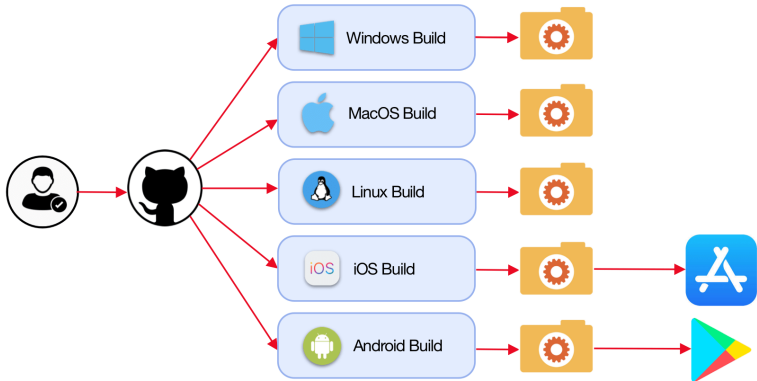
Linting tools:

- ▶ black
- ▶ pylint
- ▶ tox
- ▶ flake8
- ▶ mccabe
- ▶ mypy

GitHub Actions





GitHub Actions



- ▶ \LaTeX actions
- ▶ Nicholas Cage action

Thank you!

 PanayidesM@cardiff.ac.uk

 @Michalis_Pan

 @11michalis11