***Pynguin: Automated Unit Test Generation for Python – Analysis***

Iaguta Alen-Mihael Ion Bogdan Guceanu Marian-George

1. ***Installation***

Pynguin can be installed from the Python Package Index (PyPI) using the ‘pip’

utility tool, which is the easiest way to install it, you can also check its source code.

* pip install pynguin

After installation, there are potential risks associated with Pynguin’s execution,

so therefore, ensure to set the ‘PYNGUIN\_DANGER\_AWARE’ environment variable.

1. ***Setup***

To utilize Pynguin, specify certain parameters when invoking it from the command line:

* ‘--project-path’: Path of the project for which Pynguin should generate tests.
* ‘--module-name’: Name of the module for which tests should be generated.
* ‘--output-path’: Path where Pynguin stores the generated test cases.

In order to work, at least these 3 parameters must be provided, additional arguments will come with default values, but they can also be set.

Poza poza

1. ***Test Project***

Once configured, Pynguin generates test cases for the specified module using the default algorithm (DynaMOSA). For example:

* pynguin --project-path ./ --output-path /tmp/pynguin-tests --module-name triangle

Users can also specify alternative algorithms, such as MIO, by including --algorithm followed by the desired algorithm name in the command line:

* pynguin --project-path ./ --output-path /tmp/pynguin-results --module-name triangle --algorithm MIO

poza poza

1. ***Results***

Pynguin generates test cases based on the specified module's functionality. These test cases execute various branches of the function's implementation and include assertions to verify the function's behavior. The generated test cases are stored in the specified output path.

It's important to note that running Pynguin multiple times will overwrite the existing result files. For more verbose output, the -v or -vv parameters can be added to the command line.

Poza poza