Tool analysis document

Pynguin: Automated Unit Test Generation for Python, https://doi.org/10.1145/3510454.3516829

Team members: Crisan Dragos, Matei Daria, Mereu Bianca

Installation

The application is easy to install. The user needs to open a terminal and write `pip install pynguin`. After a successful installation, the only step left is to add an environment variable PYNGUIN_DANGER_AWARE and set it to a random value.

Usage

To generate unit tests for a specific module, the user needs to write inside the terminal the following command `pynguin --project-path C:\Users\Daria\PycharmProjects\Calculator\src --output-path C:\Users\Daria\PycharmProjects\Calculator\pynguin-results --module-name Calculator` and replace the project path, output path and module name with their own information. If there are no issues, Pynguin successfully generates a Python file with the unit tests in the output path. It also generates in the directory in which the terminal is opened a directory pynguin-report with files pynguin-config.txt and statistics.csv. The first one contains information about the project path, output path, module name, the algorithms and strategies used, while statistics.csv contains, for each execution of the previously mentioned command, the test coverage achieved by the generated tests for the target module.

Results

The tool generates valid and useful test cases, especially if parameters and return types are annotated. The reports are useful as well, especially the code coverage statistics. However, some test cases are either useless, too complicated for a simple function they test (in our case, simple methods like arithmetic add, subtract) or test for too many things inside one unit test, which defeats the purpose of a unit test. In conclusion, this tool is generally useful and, even when its performance is not as great, is a good starting point for writing unit tests manually.