Refactoring

September 17, 2024

1 Refactoring

`Refactoring is the systematic modification of code under the constraints of tests. If there ar`

The systematic modifications are available in a catalog:

https://refactoring.com/catalog/

They are of the type *Rename variable*, *Extract function*, *Inline function* and *Pull up field*. Some of them are not applicable in Python.

Pycharm is able to handle several of the refactoring recipies automatically within a project. Currently, it is the best IDE at doing so.

The refactorings fall in a few different categories:

All assume passing tests before we start

1. No interfaces change
   - Tests remain unchanged
   - Refactor
   - Assert that tests pass
2. The signature of a callable changes
   - We need to modify tests first
   - Assert that tests fail
   - Refactor
   - Assert that tests pass
3. A new interface is created (code is extracted from original callable)
   - Create test for new interface
   - Assert failing new test
   - Mock in tests of original callable
   - Assert original callable tests pass
   - Refactor
   - Assert that tests pass
4. An interface disappears
   - Modify tests for callable that absorbs the code
   - Assert that this test fails
   - Refactor
   - Assert that tests for absorbing callable passes
   - Assert that tests for disappearing callable fail
   - Remove tests for disappearing callable

5. Data structures get changed
    - *Assumes good code coverage*
    - Change data structures
    - Fix all failing tests

The quote at the top is my spoof of a friends Geology professor:

```
Mining is the profitable excavation of minerals from the earth. Note that it has to be profitab
```