Mutation testing

How does it work?

Mutation testing involves modifying a program in small ways. Each mutated version is called a *mutant.* Ideally it should introduce a breaking change to the source code. Test suite is run against the mutant to check if the change is detected by tests.

Introduce a mutation —> Run the tests suite —> Mutant survived? —> Write a new test

Mutation examples:

- `>` changed into `>=`
- `+` change into `-`
- `break` change into `continue`
- `is` change into `is not`

Let’s look at examples

```
git clone git@github.com:lenarother/mutation-testing-demo.git
cd mutation-testing-demo
pip install -r requirements.txt
pytest —cov
```

Let’s look at examples

```python
ROMAN = 'IVXLCDM'
ARABIC = (1, 5, 10, 50, 100, 500, 1000)
ROMAN_ARABIC_MAP = dict(zip(ROMAN, ARABIC))


Magdalena Rother
def get_current_roman(i, previous):
    value = ROMAN_ARABIC_MAP.get(i)
    return -value if value < previous else value


Magdalena Rother
def roman2arabic(roman):
    previous = 0
    arabic = 0
    for i in roman[::-1]:
        value = get_current_roman(i, previous)
        arabic += value
        previous = abs(value)
    return arabic
```

What can we mutate?

```python
import pytest

from src.roman_to_arabic.roman import roman2arabic


Magdalena Rother
@pytest.mark.parametrize(
    'roman, arabic',
    (
        ('I', 1),
        ('II', 2),
        ('III', 3),
        ('V', 5),
        ('X', 10),
        ('XV', 15),
        ('VI', 6),
        ('IV', 4),
        ('MCMLXXXV', 1985),
    ),
)
def test_roman_is_change_into_arabic(roman, arabic):
    assert roman2arabic(roman) == arabic
```

Which mutations will survive?

Tool example: mutmut

https://github.com/boxed/mutmut

```
mutmut

mutmut run

mutmut results

mutmut apply 15
```

Tool example: mutmut

```
(mutation-testing-demo) ➜  mutation-testing-demo git:(master) mutmut run

- Mutation testing starting -

These are the steps:
1. A full test suite run will be made to make sure we
   can run the tests successfully and we know how long
   it takes (to detect infinite loops for example)
2. Mutants will be generated and checked

Results are stored in .mutmut-cache.
Print found mutants with `mutmut results`.

Legend for output:
🎉 Killed mutants.   The goal is for everything to end up in this bucket.
⏰ Timeout.          Test suite took 10 times as long as the baseline so were killed.
🤔 Suspicious.       Tests took a long time, but not long enough to be fatal.
🙁 Survived.         This means your tests need to be expanded.
🔇 Skipped.          Skipped.

1. Using cached time for baseline tests, to run baseline again delete the cache file

2. Checking mutants
⠸ 24/24  🎉 22  ⏰ 0  🤔 0  🙁 2  🔇 0
(mutation-testing-demo) ➜  mutation-testing-demo git:(master)
```

Summary

Goals:

- evaluate the quality of existing software tests
- design new software tests
- prevent bugs before they happened

Limitations:

- extremely large number of copies of the program -> slow, requires a lot of resources

Bonus questions:

What other types of tests exist out there?

Have fun mutating 🎉