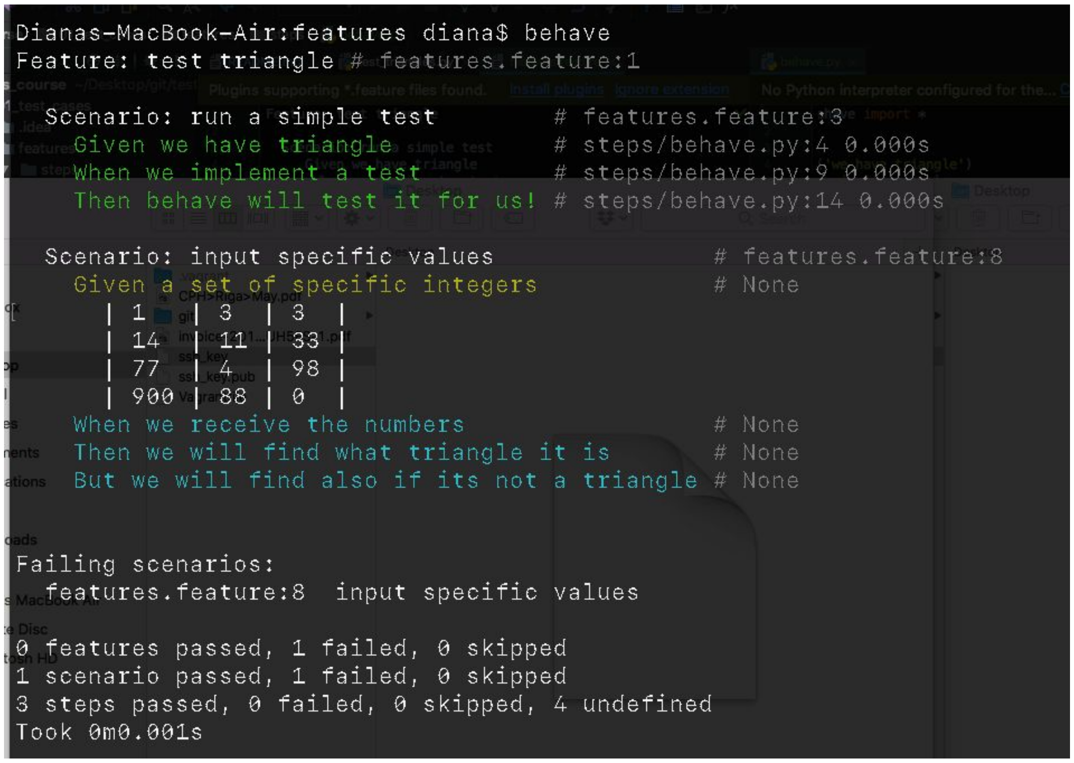
Technical proof of concept documentation

Highlights of Cucumber

Cucumber seems like a good feature for testing a project, as when you set it up, you really get a good idea of what's going on, when having to set it up with "Features, Scenarios and Steps". It's also great for covering black box testing, as it can be very repetitive. So for example in out gutenberg project, we could test various queries, as we have so many books. And we will also be able to specify a query with the cucumber test.

As the triangle project was done in Python programming language we used `behave` for features and scenarios. A feature file has a natural language format describing a feature or part of a feature with representative examples of expected outcomes. The "Given", "When" and "Then" parts form the actual steps that will be taken by behave in testing your system. Steps used in the scenarios are implemented in Python files in the "steps" directory.



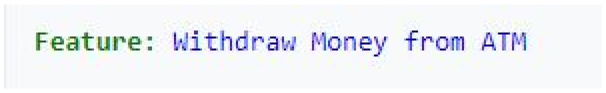
```
Dianas-MacBook-Air:features diana$ behave
Feature: test triangle # features.feature:1

  Scenario: run a simple test        # features.feature:3
    Given we have triangle           # steps/behave.py:4 0.000s
    When we implement a test         # steps/behave.py:9 0.000s
    Then behave will test it for us! # steps/behave.py:14 0.000s

  Scenario: input specific values                     # features.feature:8
    Given a set of specific integers                  # None
      | 1   | 3  | 3  |
      | 14  | 11 | 33 |
      | 77  | 4  | 98 |
      | 900 | 88 | 0  |
    When we receive the numbers                       # None
    Then we will find what triangle it is             # None
    But we will find also if its not a triangle # None


Failing scenarios:
  features.feature:8  input specific values

0 features passed, 1 failed, 0 skipped
1 scenario passed, 1 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 4 undefined
Took 0m0.001s
```

Cucumber test

A cucumber tests consists of 3 things:

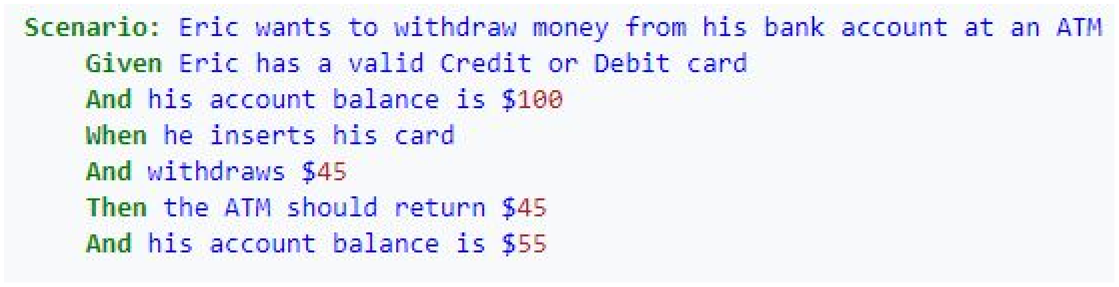
- Features
- Scenarios
- Steps

A feature is what describes the function of the software being tested:
In the picture below we see the feature for the scenario.



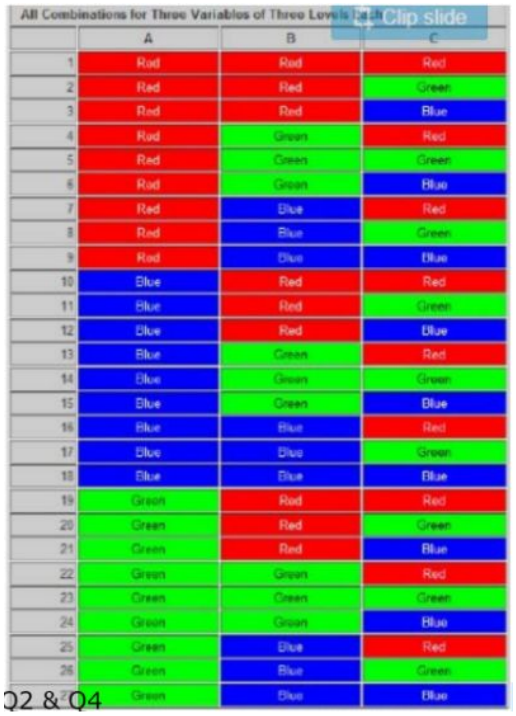
```
Feature: Withdraw Money from ATM
```

A feature contains different kind of scenarios. For example in this example it explains that Eric wants to withdraw money, and he wants to do it from his bank account.
And then the steps explain exactly what happens. Given is the preconditions, When is actions and Then is results.
In the example below we see “And” too, which it just used for adding more to each of them, so they’re not bound to a specific step.



```
Scenario: Eric wants to withdraw money from his bank account at an ATM
    Given Eric has a valid Credit or Debit card
    And his account balance is $100
    When he inserts his card
    And withdraws $45
    Then the ATM should return $45
    And his account balance is $55
```

Pairwise testing



All Combinations for Three Variables of Three Levels Each

	A	B	C
1	Red	Red	Red
2	Red	Red	Green
3	Red	Red	Blue
4	Red	Green	Red
5	Red	Green	Green
6	Red	Green	Blue
7	Red	Blue	Red
8	Red	Blue	Green
9	Red	Blue	Blue
10	Blue	Red	Red
11	Blue	Red	Green
12	Blue	Red	Blue
13	Blue	Green	Red
14	Blue	Green	Green
15	Blue	Green	Blue
16	Blue	Blue	Red
17	Blue	Blue	Green
18	Blue	Blue	Blue
19	Green	Red	Red
20	Green	Red	Green
21	Green	Red	Blue
22	Green	Green	Red
23	Green	Green	Green
24	Green	Green	Blue
25	Green	Blue	Red
26	Green	Blue	Green
27	Green	Blue	Blue

Q2 & Q4

In pairwise testing, we're testing all the possible combinations of given inputs, so that we can quickly blackbox test all the outputs. In the picture above we see 3 colours. There are 27 combinations 3^3, which becomes 3 * 3 = 9, 9 * 3 = 27.
This would take a lot of time if it had to be done manually. We therefore have fantastic tools like "AllPairs" and such, which will do this for us. It is however limited to certain amounts, as this will quickly grow into massive numbers. If it's for example just 5 colors, it will become 15625 different combinations. And will take up a lot of resources.