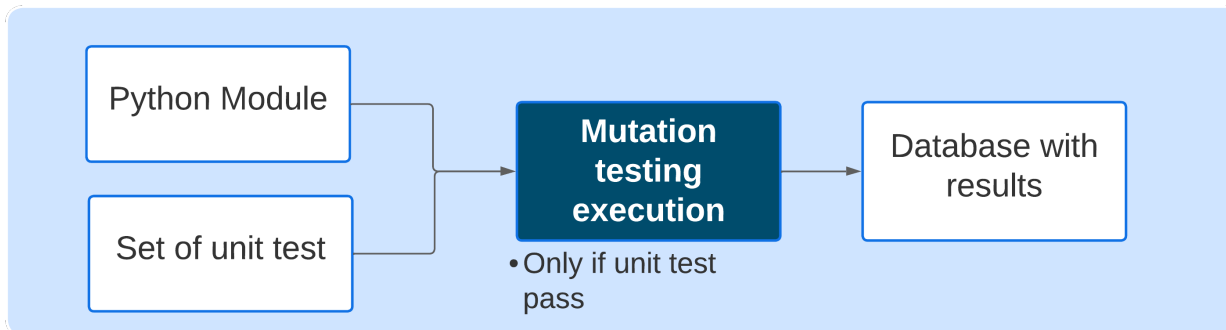# Motivation

Mutation testing is a technique that is frequently discarded by companies when it comes to performing augmented testing. This is due to the high number of mutations that can be generated, resulting in a high computational requirement, which is not frequently available to most companies.

MTP proposes a smart unit testing approach that allows to perform an exhaustive and optimized set of unit test cases based on applying IA to the mutation testing technique.
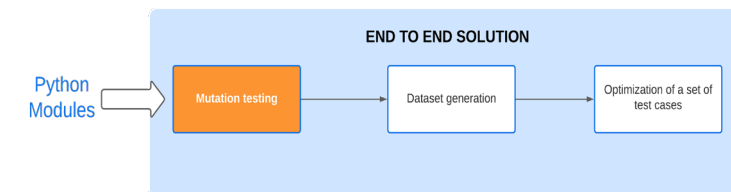
Proposed solution

END TO END SOLUTION

Python Modules → Mutation testing → Dataset generation → Optimization of a set of test cases
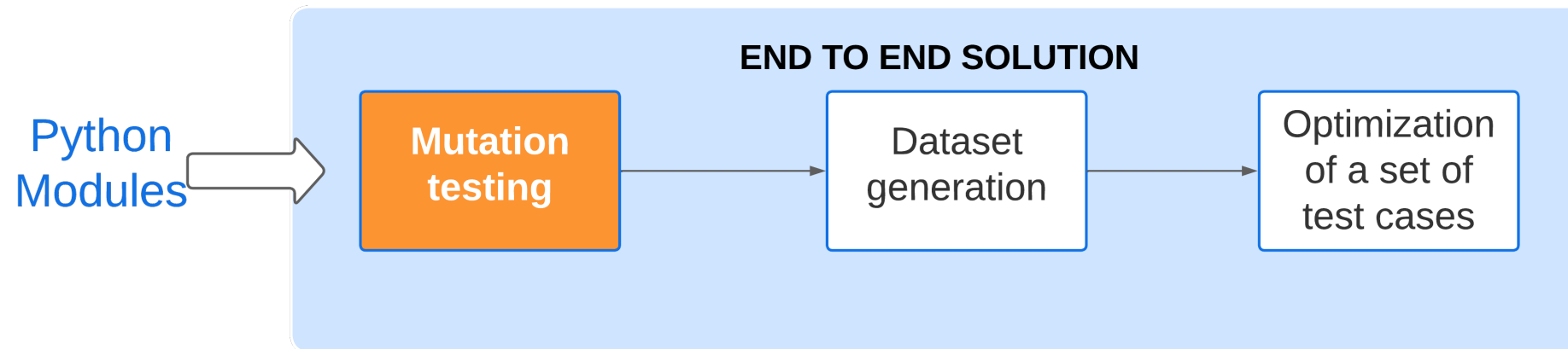
Proposed solution（Mutation testing）

Python Module

Set of unit test

Mutation testing execution

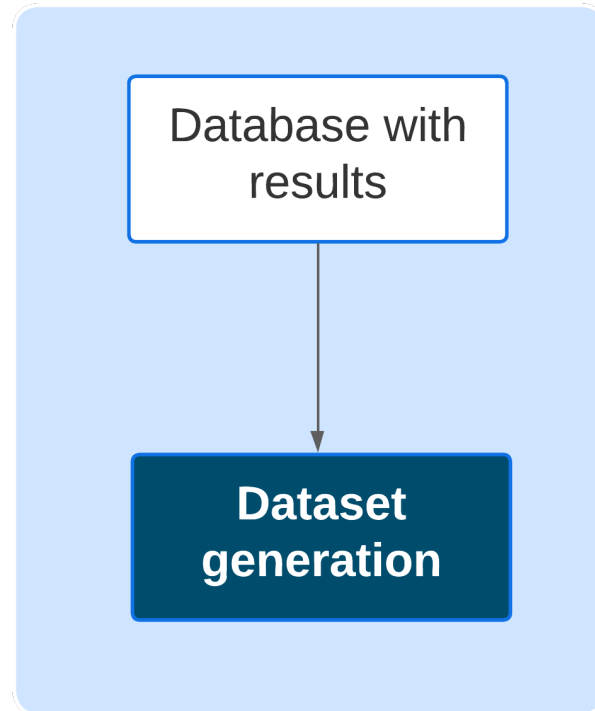- Only if unit test pass

Database with results

In the first step of our solution, we provide the module under test and its base set of unit tests, on which the mutation tests will be executed.

In the first instance, the unit tests will be executed. If they pass successfully, mutations will be generated and executed, ultimately producing a database containing the results.
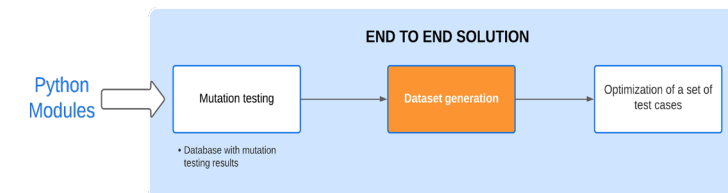
END TO END SOLUTION

Python Modules

Mutation testing

Dataset generation

Optimization of a set of test cases

Proposed solution（dataset generation）

Database with results

↓

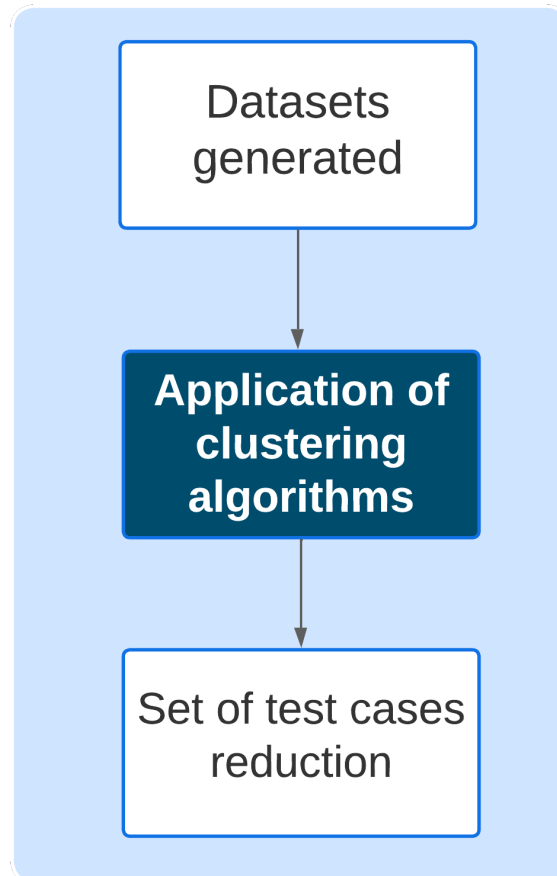Dataset generation

With the results of the mutation tests we performed a preprocessing to generate a dataset of each function under test.

This dataset is primarily structured with each test case represented as columns and each result as rows.
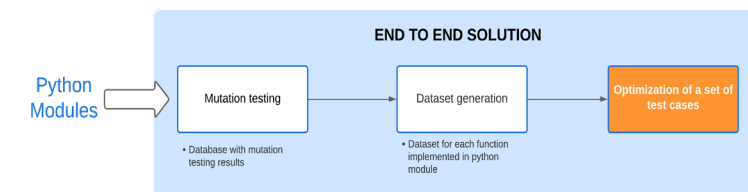
END TO END SOLUTION

Python Modules → Mutation testing → Dataset generation → Optimization of a set of test cases

• Database with mutation testing results

Proposed solution（optimization）

```
┌─────────────────────┐
│      Datasets       │
│     generated       │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│   Application of     │
│    clustering        │
│    algorithms        │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│   Set of test cases │
│      reduction       │
└─────────────────────┘
```

Each generated dataset represents a set of test cases with the results obtained from the mutation tests.

For each one we will apply clustering algorithms to group the mutations by identifying the patterns they follow.

Finally, we will select one mutation from each cluster forming with them the optimized set of mutations.

END TO END SOLUTION

Python Modules → Mutation testing → Dataset generation → Optimization of a set of test cases

- Database with mutation testing results
- Dataset for each function implemented in python module

Future works

❑ Research to optimize the number of operators required for mutation testing.

❑ Research in techniques based on genetic algorithms for optimization.

❑ Development of further experiments to validate our prototype.

Conclusion

- ❑ We have demonstrated that the number of mutations required can be significantly reduced.

- ❑ Our solution can help in:

    - ❑ Performing optimized mutation testing on traditional software systems.
    - ❑ Performance of optimized mutation tests on systems with embedded artificial intelligence.

Any further questions?

jonatan.dominguez@mtp.es

lredondo@mtp.es

innovacion@mtp.es