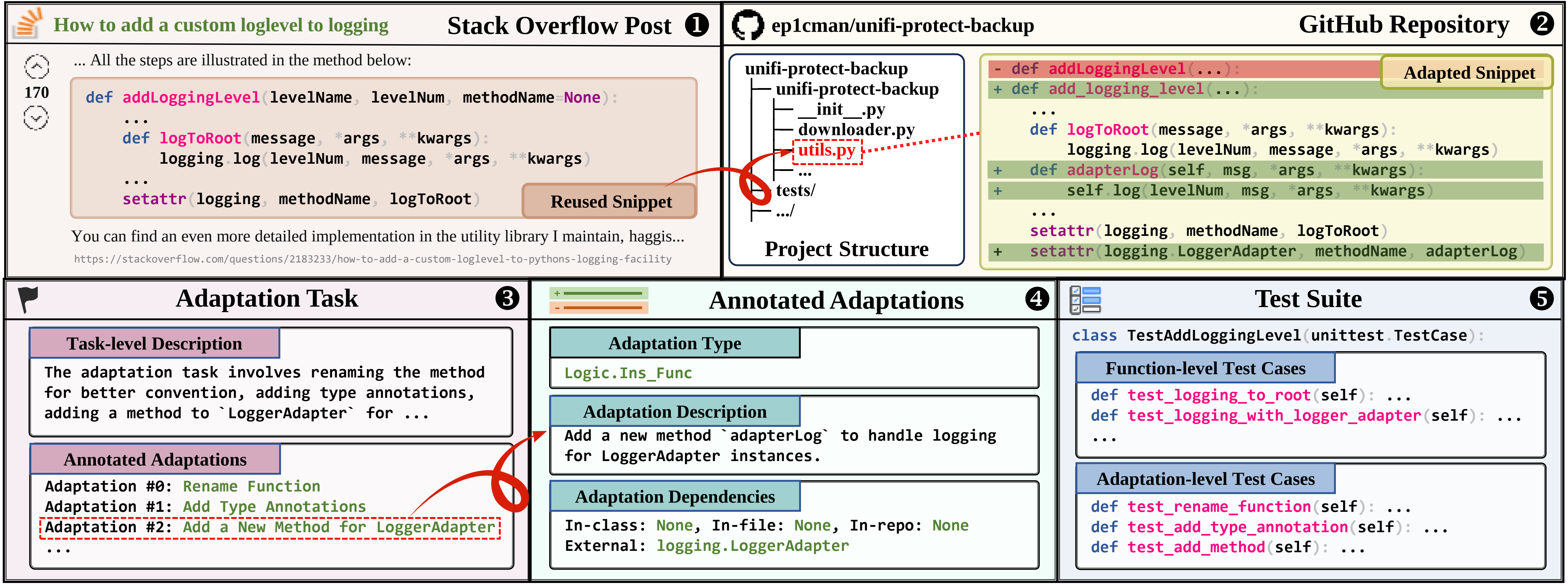

How to add a custom loglevel to logging

Stack Overflow Post ❶

170

... All the steps are illustrated in the method below:

```
def addLoggingLevel(levelName, levelNum, methodName=None):
    ...
    def logToRoot(message, *args, **kwargs):
        logging.log(levelNum, message, *args, **kwargs)
    ...
    setattr(logging, methodName, logToRoot)
```

Reused Snippet

You can find an even more detailed implementation in the utility library I maintain, haggis...

https://stackoverflow.com/questions/2183233/how-to-add-a-custom-loglevel-to-pythons-logging-facility

ep1cman/unifi-protect-backup

GitHub Repository ❷

```
unifi-protect-backup
 ├─ unifi-protect-backup
 │   ├─ __init__.py
 │   ├─ downloader.py
 │   ├─ utils.py
 │   └─ ...
 ├─ tests/
 └─ .../
```

Project Structure

Adapted Snippet

```
- def addLoggingLevel(...):
+ def add_logging_level(...):
      ...
      def logToRoot(message, *args, **kwargs):
          logging.log(levelNum, message, *args, **kwargs)
+     def adapterLog(self, msg, *args, **kwargs):
+         self.log(levelNum, msg, *args, **kwargs)
      ...
      setattr(logging, methodName, logToRoot)
+     setattr(logging.LoggerAdapter, methodName, adapterLog)
```

Adaptation Task ❸

Task-level Description

The adaptation task involves renaming the method for better convention, adding type annotations, adding a method to `LoggerAdapter` for ...

Annotated Adaptations

Adaptation #0: Rename Function
Adaptation #1: Add Type Annotations
Adaptation #2: Add a New Method for LoggerAdapter
...

Annotated Adaptations ❹

Adaptation Type

Logic.Ins_Func

Adaptation Description

Add a new method `adapterLog` to handle logging for LoggerAdapter instances.

Adaptation Dependencies

In-class: None, In-file: None, In-repo: None
External: logging.LoggerAdapter

Test Suite ❺

```
class TestAddLoggingLevel(unittest.TestCase):
```

Function-level Test Cases

```
def test_logging_to_root(self): ...
def test_logging_with_logger_adapter(self): ...
...
```

Adaptation-level Test Cases

```
def test_rename_function(self): ...
def test_add_type_annotation(self): ...
def test_add_method(self): ...
```